Smart Card Security

Volume I (of II)

Asier Goikoetxea Yanci

A themed portfolio submitted to

The Universities of

Glasgow

Edinburgh

Strathclyde

Heriot Watt

For the Degree of

Doctor of Engineering in System Level Integration

Sponsored by:

Abstract

Smart Card devices are commonly used on many secure applications where there is a need to identify the card holder in order to provide a personalised service. The value of access to locked data and services makes Smart Cards a desirable attack target for hackers of all sorts. The range of attacks a Smart Card and its environment can be subjected to ranges from social engineering to exploiting hardware and software bugs and features.

This research has focused on several hardware related attacks and potential threats. Namely, power glitch attack, power analysis, laser attack, the potential effect on security of memory power consumption reduction techniques and using a re-configurable instruction set as method to harden opcode interpretation.

A *semi*-**automated simulation environment** to test designs against glitch attacks and power analysis has been developed. This simulation environment can be easily integrated within Atmel's design flow to bring assurance of their designs' behaviour and permeability to such attacks at an early development stage. Previous power analysis simulation work focused on testing the implementation of part of the cryptographic algorithm. This work focuses on targeting the whole algorithm, allowing the test of a wider range of countermeasures.

A common glitch detection approach is monitoring the power supply for abnormal voltage values and fluctuations. This approach can fail to detect some fast glitches. The alternative approach used in this research monitors the effects of a glitch on a mono-stable circuit sensitive to fault injection by glitch attacks. This work has resulted in a **patented glitch detector** that improves the overall glitch detection range.

The use of **radiation countermeasures as laser countermeasures and potential sensors** has been investigated too. Radiation and laser attacks have similar effects on silicon devices. Whilst several countermeasures against radiation have been developed over the years, almost no explicit mention of laser countermeasures was found. This research has demonstrated the suitability of using some radiation countermeasures as laser countermeasures.

Memory partitioning is a static and dynamic power consumption reduction technique successfully used in various devices. The nature of Smart Card devices restricts the applicability of some aspects of this power reduction technique. This research line has resulted in the **proposal of a memory partitioning approach suitable to Smart Cards**.

Table of Contents

List of Figures and Tables

List of Accompanying Material

The following material is included in the Volume II of this portfolio

Technical Reports:

SimEnvTech1:	Glitch Attack and Power Analysis Simulation Environment
SimEnvTech2:	Counter Simulation Results
LaserTech1:	Tartalo test-chip 01OKA
GlitchTech1:	Glitch Detector Report
LowLeakageTech1:	SRAM Memory Partitioning for Leakage Reduction

Publications:

GlitchPub1: Goikoetxea Yanci, A.; Pickles, S.; Arslan, T., "Detecting Voltage Glitch Attacks on Secure Devices," *ECSIS Symposium on Bio-inspired Learning and Intelligent Systems for Security (BLISS '08)*, pp.75-80, 4-6 Aug. 2008.

GlitchPub2: Goikoetxea Yanci, A.; Pickles, S.; Arslan, T., "Characterization of a Voltage Glitch Attack Detector for Secure Devices," *ECSIS Symposium on Bio-inspired Learning and Intelligent Systems for Security (BLISS '09)*, pp.91-96, 20-21 Aug. 2009.

GlitchPub3: Goikoetxea Yanci, Asier, “Detecting Voltage Glitches”, *International Patent Publication Number*, WO 2008/033712 A2.

Acknowledgements

I am very grateful to the Engineering and Physical Sciences Research Council (EPSRC) and my industrial sponsor, Atmel Smart Card ICs, for their funding. Their financial support has made this research possible. I also want to thank Atmel for providing me with the opportunity to carry a research with them and for them.

Thank you to my academic supervisors Professor Tughrul Arslan and Dr. Ahmet Erdogan for their advice and feedback in many aspects of the research and for encouraging me to submit a number of publications. An especial Thanks goes to Professor Tughrul Arslan for all the time he has dedicated to this research.

I am also grateful to my industrial supervisors Russell Hobson and Steve Pickles for their guidance on various technical subjects, countless discussions and undivided attention. Thanks also to Steve for proof reading all the reports and interfacing with the Security Group to solve any confidentially issues regarding this publication. I do not forget either the help of many Atmel employees that helped me at different stages on this research. Thank you.

I owe a big thank you to the Institute for System Level Integration for their assistance throughout the years. Specially for allowing me to join the MSc, for their efforts on transferring my MSc credits toward the EngD programme and for help on different topics related to the EngD. Professor Steve Beaumont, Wendy Glendenning, Sandie Buchannan and Siân Williams have made all this possible.

Last, but not least, I want to thank my friends and family for too many reasons to mention them here. Special thanks to my parents for making me who I am.

Asier Goikoetxea Yanci
January 2011

Author’s Declaration

I declare that the research presented in this portfolio is entirely my own contribution, unless otherwise stated. This research has been carried out with the assistance of my academic and industrial supervisors.

Andrew Burnside (EngD student at Atmel) provided me with his Matlab scripts to carry a difference of means (DOM) differential power analysis (DPA), which I adapted to for the simulation environment covered in section 3.1.

David Dougan and Louis Frew provided me with the operational amplifier and voltage regulators used in the process of testing the proposed glitch detector, section 3.2.

Jalib Ahmed provided me with the RTL and program code of the Cratis device, which I adapted and used in the silicon glitch attack test environment used in section 3.2.3.3 and section 4.4.2 and covered in detail in the technical report GlitchTech1. John Connor provided me with the source code of his own front-end glitch attack control application (VGlitch), which I adapted to the test needs of the mentioned silicon glitch attack test environment.

Asier Goikoetxea Yanci
January 2011

Acronyms

3DES	Triple-DES
AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
AMS	Analog Mixed Signal
APDU	Application Protocol Data Unit
API	Application Protocol Interface
ASIC	Application Specific Integrated Circuit
ATM	Automated Teller Machine
CC	Common Criteria
CMOS	Complementary Metal-Oxide Semiconductor
CPA	Correlation Power Analysis
CPU	Central Processing Unit
DES	Data Encryption Standard
DICE	Dual Interlocked storage Cell
DOM	Difference of Means
DPA	Differential Power Analysis
DRV	Data Retention Voltage
DUT	Device Under Test
DVS	Dynamic Voltage Scaling
EDA	Electronic Design Automation
EEPROM	Electronic Erasable Programmable Read Only Memory
ELT	Enclosed Layout Transistor
EMA	Electromagnetic Analysis
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GAPASE	Glitch Attack and Power Analysis Simulation Environment
GND	Ground
GSM	Global System for Mobile Communications
HDL	Hardware Description Language
IC	Integrated Circuit
IDB	Instruction Decoder Block
IDE	Integrated Development Environment
IP	Intellectual Property
ISA	Instruction Set Architecture
ISO	International Standards Organisation
LUT	Look-Up Table
MOS	Metal-Oxide Semiconductor
NMOS	Negative Metal-Oxide Semiconductor

NVM	Non-Volatile Memory
OS	Operating System
POR	Power on Reset
PCB	Printed Circuit Board
PIN	Personal Identification Number
PMOS	Positive Metal-Oxide Semiconductor
PSL	Post-Simulation Layer
RC	Resistor Capacitor
ReDISA	Re-configurable Dual Instruction Set Architecture
RH	Radiation Hardened
RHBD	Radiation Hardened by Design
RHP	Radiation Hardened Process
RIDB	Re-configurable Instruction Decoder Block
RISP	Re-configurable Instruction Set Processors
RMM	Reuse Methodology Manual
ROM	Read Only Memory
RS	Reset-Set
RSA	Rivest, Shamir & Adleman
RTL	Register Transfer Level
SCD-A	Smart Card Device A
SCOS	Smart Card Operating System
SEE	Single-Event Effect
SERT	Single-Event Resistant Topology
SET	Single-Event Transient
SEU	Single-Event Upset
SIDB	Standard Instruction Decoder Block
SIM	Subscriber Identification Module
SNM	Static Noise Margin
SMM	Security Methodology Manual
SoC	System on Chip
SPA	Single Power Analysis
SPI	Serial Peripheral Interface
SPICE	Simulation Program with Integration Circuit Emphasis
SPM	Scratch-pad Memory
SRAM	Static Random Access Memory
SSOL	Simulation Specific Objects Layer
TTM	Time-to-Market
USB	Universal Serial Bus
VR	Voltage Regulator
WDDL	Wave Dynamic Differential Logic

1 Introduction to Smart Cards and portfolio structure

Smart Cards have their origin in the plastic cards commonly used for the last few decades to identify customers and provide them with personalised services. A few examples of these cards include national insurance cards, library membership cards or video club cards. Initially, these cards might have included just the customer name and identification number, and, hence, were easy to duplicate and to commit fraud.

Certain services required some security features that would make card duplication more difficult and authenticate the rightful owner of the card. This was achieved with the addition of a magnetic stripe, which, among other data, stored the card identification number and the card's expiry date. These cards were also accompanied with a personal identification number (PIN), the purpose of which was to authenticate the user. A common example of the plastic card with a magnetic stripe is a credit or debit card.

Despite the added security feature, these cards could still be duplicated. In fact, financial institutions lost £505 million in 2004 due to fraud [1]. These figures demonstrate that the security measures in place were hardly a deterrent for those committing fraud. Furthermore, the use of the PIN number already included in the magnetic stripe was unevenly adopted or enforced across different countries or services. For example in the UK, the PIN number of a debit card was required only when using it in an automated teller machine (ATM). When using a debit card in a shop, the customer would identify or authenticate himself with a hand written signature on the receipt, which should match to the one on the reverse of the card.

On the other hand, Spain had a different approach to authenticating a card owner in a shop. The shop tender was required by law to request the card holder provide a valid proof of identification, such as the national identity card. Furthermore, when using debit cards, the customer should also enter the PIN number in order to complete a purchase transaction. Credit cards did not need the PIN number entered to complete the transaction.

Smart Cards are the next step in the evolution from the original plastic cards, and they add yet another degree of security. Smart Cards are plastic cards with an embedded integrated circuit (IC) and are used in a wider range of applications, such as e-money cards (e.g. for transport), subscriber identification module (SIM) cards, e-banking (e.g. credit cards) or government (e.g. identity cards).

Smart Cards can be classified into two different categories depending on the components or intellectual properties (IP) they are made of. These are memory Smart Card and microcontroller Smart Card. A memory Smart Card is based on a non-volatile memory (NVM) which is externally accessed to store and/or read information. These Smart Cards are usually accessed to through a Finite State Machine (FSM), just as shown in the block diagram in Figure 1-1. A microcontroller Smart Card, on the other hand, is a System on Chip (SoC) device with a microcontroller, program and data memories, and a series of other IPs. A block diagram of microcontroller Smart Cards is shown in Figure 1-2. Smart Cards are shipped to the end user with an embedded operating system (OS) and applications tailored to the target application needs.

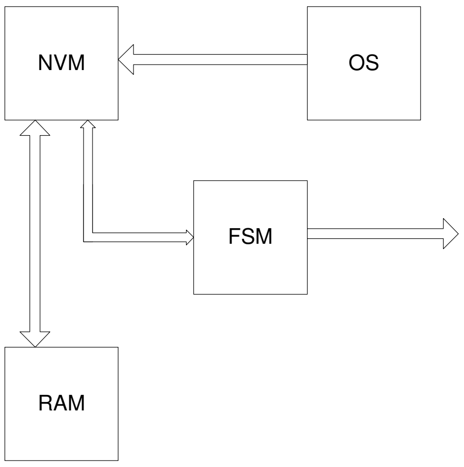


Figure 1-1 Block diagram of the typical memory card

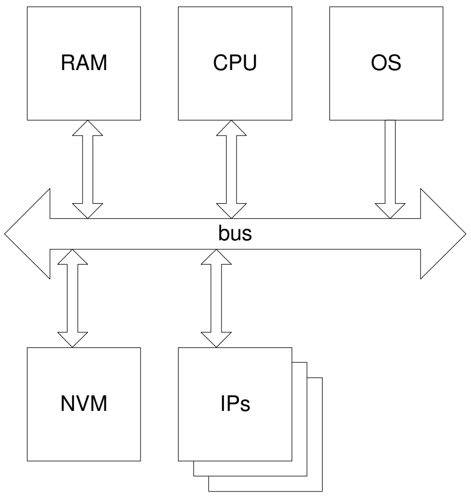


Figure 1-2 Block diagram of the typical microcontroller card

The key benefit of Smart Cards goes beyond the pure user authentication feature of magnetic stripes. Firstly, the security features embedded into the IC make it harder to be duplicated and, hence, to achieve fraud [1]. Secondly, Smart Cards can also store data and restrict the access to this data. One common, everyday use example is the phonebook function embedded into SIM cards. The size of these phonebooks is limited by the available NVM within the Smart Card. The access to this phonebook is granted after entering the right PIN number after powering up the SIM card. This is, unless the user has disabled the PIN number feature.

Another example where data is stored into a Smart Card is the satellite or cable TV receiver decoder or set-top-box. Here, the Smart Card holds information on which TV channels the customer has subscribed to and is allowed to watch. In this case, however, the access to this information is restricted to just the service provider and the set-top-box. The end customer does not have rightful means or needs of gaining access to this information.

Most of the environments and applications where microcontroller Smart Cards are used, involve the need for secure and/or secretive data transactions. For this purpose, Smart Cards are designed with built-in security measures that range from hardware to software. With an increasing number of applications using Smart Cards, however, there is also an increased desire to break the security brought by these devices, as it could grant access to otherwise restricted services. In turn, this has resulted in a continuous need to improve the security of the whole environment and application.

This portfolio does not intend to be a hacker's manual, hence, it will not discuss how the security might be broken nor reference any place or report where this is discussed or explained. However, just a sample of the motivation behind could help the reader understand the drive of hackers to break the security measures, and the need of the Smart Card industry to enhance the security level. Consider the set-top-box case above, where the end user has no access to the information. If the end user manages to gain access to the information stored in the Smart Card or manages to fool the set-top-box, they could potentially watch more TV channels than originally subscribed to.

Focusing on the Smart Card and its environment, it can be subjected to threats and attacks of different natures. The main attacks and threats can target:

- the end user, through social engineering, behaviour analysis or even personal threats;
- the communication channels between the Smart Card device and Smart Card reader (commonly known as reader terminal), by eavesdropping, behavioural analysis and fault injection;
- software or OS, by exploiting any bug or faults; and
- the Smart Card itself, with fault injections such as power glitch attacks, laser attacks or side-channel attacks such as power analysis.

Atmel, as one of the leading Smart Card manufacturer companies, has a great interest in keeping its product portfolio secure against current and forthcoming threats. The core business of the Atmel Smart Card division centres on designing and developing microcontroller Smart Card devices. Therefore, this research focuses on security issues, threats and attacks that target the microcontroller Smart Card IC or its internal hardware. Attacks related to their internal software and their environment are out of the scope of this research. For the rest of the research, microcontroller Smart Card devices will be referred to as Smart Card devices. The next section provides an overview of the structure of this portfolio, the work carried out in this research and the main achievements.

1.1 Portfolio structure

This portfolio is divided into two volumes. This document, Volume I, covers the various research carried out in relation to Smart Card security. The contents of this volume are discussed next.

Chapter 2 of this volume sets the context of this research by giving an introduction to the Smart Card device and the manufacturing industry. This chapter also links all the different research topics carried out in this work under the 'Smart Card Security' umbrella, and lists the resulting contributions. Due to the distinctive nature of the different research areas covered in this work, Chapters 3 to 5, the literature review of each research area is covered individually within the chapter related to each research topic.

Chapter 3 Fault Tolerant Design covers three research lines focused on improving the Smart Card's fault tolerance to the three main attack techniques. These are power glitch attack, power analysis attack and laser attack. Power glitch and laser attacks are active attacks that can be used to inject faults into the Smart Card or alter the program execution flow, which might provide valuable behavioural information to an experienced hacker. Power analysis attack, on the other hand, is a passive attack technique used gain knowledge on the processed data by analysing the Smart Card's power consumption waveforms.

The research carried out in Chapter 3 has resulted in a glitch attack and power analysis simulation environment to enable testing designs against these two common attack techniques at an early development stage. The second research line has produced a new glitch detector capable of detecting fast glitches that are missed by current detectors. The third and final research line on fault tolerance has focused on investigating the applicability of radiation hardening techniques to Smart Cards.

Chapter 4 Low Power Design for Smart Cards looks at the applicability of several SRAM leakage reduction techniques on a Smart Card device. The introduction of new technologies and the adoption of new design techniques can potentially impact on the security level of a Smart Card device. One of the two main research lines covered in this chapter is aimed at determining the impact on SRAM robustness a few SRAM leakage reduction design techniques might have when the SRAM is subjected to glitch attacks. The second main research line covered in this chapter proposes a memory partitioning approach focused on the needs of a Smart Card. For this technique to be applicable to Smart Cards and minimising the impact on security, the memory scrambler will need to be adapted to the chosen memory partitioning approach.

Chapter 5 Re-Configurable Instruction Set CPU for Smart Cards covers a re-configurable CPU proposal where the CPU's instruction set (IS) can be partially modified on-the-fly. As previously stated, power analysis attack can be used to discover the processed data. It can also be used to find out instructions being executed. Furthermore, using a CPU with a publicly known IS has a higher risk of program opcode interpretation should it be made public. This chapter discusses several approaches, advantages and disadvantages of de-standardising the IS to harden the power analysis and opcode interpretation.

Chapter 6 finishes this volume, Volume I, discussing general conclusions on Smart Card security from a higher level perspective than the research covered in this work. Regarding future work, this chapter summarises the main lines of action proposed in chapters 3 to 5.

Volume II of this portfolio includes technical reports, papers and patents presented as a result of the research carried out and information on the work carried out to support this research. For more information on its contents, please refer to the Volume II document.

2 Smart Card Industry, device background and Portfolio contribution

This chapter provides an overview of the Smart Card industry, the device and its threats. It also presents the contributions of this research towards the Smart Card Security field.

2.1 Smart Card Industry

From its design to its usage, there are five main parties involved with Smart Cards: a) manufacturers; b) evaluators; c) vendors; d) customers (e.g. banks); and, e) the end user (i.e. the people that use a given service). Manufacturing companies, where we can find semiconductor companies like Atmel, focus on designing and developing hardware devices according to the requirements of the Smart Cards' target applications. These requirements include not only feature specifications but also security requirements, which are defined by the Common Criteria (CC) framework, international standard organisation (ISO) 15408.

Each new product is evaluated by an external evaluator, such as the German Federal Office for Information Security or the French Secretariat General for National Defence (French Network and Information Security Agency), to determine if they actually meet the application's security requirements. Meeting these requirements does not mean the device is unbreakable, instead it provides a degree of certainty that the device has been designed, tested and documented following a certain methodology. These evaluations are very costly in terms of time and money, and any issue highlighted at this stage, will have a big impact on the cost and delivery dates of the device. Hence, manufacturer companies need a means of testing their products during and after production by themselves in order to maximise the chances of passing the evaluation process.

The Smart Card vendors, such as Gemalto, Giesecke & Devrient and Oberthur Technologies, focus on developing the software side of the Smart Cards (OS and applications) and the packaging according to their customers' requirements. These customers use Smart Card devices to identify their customers (the end user) and provide unique and personalised services, e.g pay per view TV. More often than not, these same customers offer Smart Card devices to the end user free of charge, which impacts the manufacturers' selling prices.

Unlike other industries, where a company might protect its IP against industrial theft, the Smart Card industry protects itself further by dividing the knowledge into the different parties involved. In other words, no one party involved knows everything about a given Smart Card product. In fact, due to their high dependency on security, this knowledge division often happens even at company level.

As a result, the Smart Card industry is very secretive and little information is published by the manufacturers themselves regarding their research on threats and or countermeasures, as otherwise it could be potentially used by attackers to target their own devices. Hence, most of the publications related to Smart Card security have been published by third party research institutions.

2.2 The Device

Smart Cards are regulated by the ISO-7816 standard, which defines their physical, electrical, communication protocols and many more aspects. There are, however, some internal aspects not regulated by this standard, such as the actual design, what kind of CPU to use or what kind of countermeasure should be instantiated for a given threat. These are left to the manufacturer company to choose from and could mean the difference between gaining or losing potential customers and impact their competitive advantage.

Smart Cards communicate with the external world via Smart Card readers (a.k.a. reader terminals), which also have the function of powering the Smart Card. Two approaches have been standardised to power and communicate with a Smart Card: contact and contact-less. Contact Smart Cards are powered and communicate via physical connectors. These Smart Cards are packaged with pins and physically connected to the reader terminal to power the Smart Card, drive control signals (such as clock and reset), and perform data transactions. Figure 2-1 shows the contact Smart Card module's pin-out. Contact-less Smart Cards, on the other hand, are powered via an electromagnetic fields and communicate by load modulation into field.

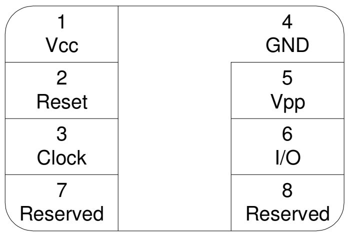


Figure 2-1 Smart Card module pinout

In contact Smart Cards, the power is supplied between the Vcc and GND pins. The ISO 7816 part 3 determines the allowed external supply voltage levels, which currently are 5V, 3V and 1.8V. However, the technology node Smart Cards are developed to operate at lower voltage levels, hence, they require a built-in DC/DC voltage converter. The input of this DC/DC converter is fed with the external supply voltage, Vcc, and its output provides the Smart Card's internal supply voltage, Vdd. For the rest of this portfolio, Vcc refers to the supply voltage provided externally to the Smart Card, that is, the DC/DC converter's input pin. On the other hand, Vdd refers to the Smart Card's internal supply voltage, which is provided by the DC/DC converter's output; in other words, the voltage used by the Smart Card to operate internally. Also, for the rest of this portfolio, the DC/DC voltage converter will be referred to as a voltage regulator. Figure 2-2 shows a diagram of how the voltage regulator links the power rails.

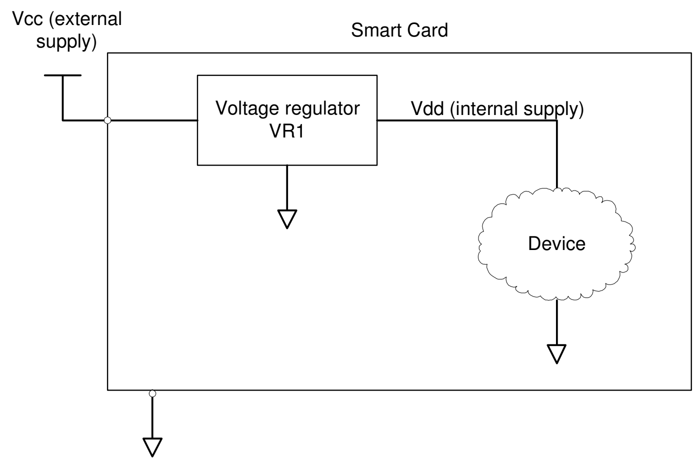


Figure 2-2 Voltage regulator in a Smart Card device

The Reset pin in Figure 2-1 is used to apply a hard reset to the Smart Card device. The Clock pin is used to provide the Smart Card with a clock signal. Although originally this signal was used to clock both the communication channel and the internal system, modern devices have an internal clock generator, so that the system and communication channel use independent clock signals. This is a security enhancement as any attack on the external clock pin may affect the communication but not necessarily the internal system.

The I/O pin (also known as I/O 0) is a bi-directional half-duplex communication port used to send and receive data to and from the Smart Card reader and is synchronised with the Clock signal. Optionally, devices could have an additional I/O (usually referred to as I/O 1). More modern devices also include a serial peripheral interface (SPI) and USB ports. These additional ports can only be accessible by using a package different to that shown in Figure 2-1.

Regarding the device internals, the bare minimum IPs embedded into a Smart Card include: a voltage regulator (covered above); a CPU; program memory; data memory; an I/O interface (covered above) and some security sensors. In addition to these IPs, a Smart Card can also include: a hardware instantiation of cryptographic algorithms and or cryptographic accelerators; more sensors; firewalls that control the memory access.

CPUs used in Smart Cards have traditionally been 8- or 16-bit ones, although more recently 32-bit ones are being used too. The CPUs are either secure versions of commercially available ones (e.g, AVR, C51 or ARM) or custom designed ones. The main benefits of using versions of commercially available CPUs are that: a) software developers can have prior knowledge of the CPU's assembly language; and b) third party tools might be already available to develop code for that particular CPU even before the physical device exist. In short, its impact on the time-to-market (TTM) is minimised. However, these devices might be more sensitive to code leakage (i.e. someone getting access to the code), as it would be easier to understand it. Custom CPUs, on the other hand, can be safer against code leakage, however, the closed nature of its Instruction Set Architecture (ISA) would reduce the software developers' exposure to it and limit the amount of available development tools, having a potentially greater impact on the TTM and overall cost.

The program memory space is usually occupied with a ROM and an NVM such as EEPROM or Flash. The ROM memory might hold functions (libraries) provided by the Smart Card vendor together with the manufacturer's code. The NVM on the other hand usually holds the Smart Card's OS and any application developed by the vendor.

The data memory is occupied with both SRAM and NVM memories, where SRAM is used to store any temporal data related to Smart Card operations, such as buffers for data transactions and temporal storage for encryption algorithms. NVM is used to store application and personalisation data, such as the personal identification number (PIN) number, bio-metric data or a phone book.

Some Smart Cards also include hardware instantiations of cryptographic algorithms and or cryptographic hardware accelerators. There are two kinds of cryptographic algorithms, symmetric and asymmetric. Symmetric algorithms use the same key to encrypt and decrypt a message, this is, the sender (encrypting device) and receiver (decrypting device) share the same key. For the communication to be secure with such algorithms, the key must be kept secret, as leaking it can result in an observer gaining access to the encrypted message. The Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are the typical symmetric cryptographic algorithms used in Smart Cards. These algorithms can be fully instantiated either in hardware or in software.

Asymmetric algorithms use different keys to encrypt and decrypt a message, where the encryption key can be public and the decryption key is private. In this scheme, both devices, i.e. Smart Card and reader terminal, only share their public keys. To communicate, the sender encrypts the message with the receiver's public key. Upon receiving the encrypted message, the receiver decrypts it with its private key. For the communication to be secure with such algorithms, all four keys should be kept secret. The private key only known by the device that owns it (i.e. either the Smart Card or the reader) and the public key shared only between the devices that take part in the communication. Asymmetric algorithms, such as Rivest, Shamir & Adleman (RSA), can also be fully instantiated in hardware or in software. Hardware accelerators may also be instantiated to speed up the software implementations.

An attacker gaining access to a public key does not compromise the integrity of any encrypted message, as the public key cannot be used to decrypt a message. The attacker could, however, use the public key to inject corrupt messages in the communication channel, even if it only can target the device owning that public key. In other words, the attacker could encrypt messages in one direction. Such attacks could be used to perform behavioural analysis.

An attacker gaining access to a private key, on the other hand, compromises the messages being sent to that private key owner device (either the Smart Card or the reader terminal), as the attacker could decrypt those messages. Hence, leaking a private key has worse implications than leaking a public one. However, in some applications, different Smart Cards could have different private keys, meaning that gaining access to the private key of one Smart Card device does not necessarily affect other devices nor it necessarily compromises the application.

Other features embedded into Smart Cards are attack countermeasures and sensors. Sensors are used to monitor the Smart Card's environment parameters for alterations that could affect the device's performance and raise an alarm if any of these parameters are out of the specification values. Typical sensors monitor for variables such as temperature, clock frequency and glitch and voltage variations on the power rails. Sensors are also used to detect light (such as laser beams) applied to the Smart Card die in order to inject faults into the device. Typical countermeasures include: adding a metal mesh on the die's top layer to avoid micro-probing the device; and design techniques that improve cryptographic algorithm implementations against power analysis (more about this in section 2.3.2).

In addition to all these hardware features, Smart Cards also include the software they need to operate, which, as previously stated, includes functions or libraries provided by the Smart Card manufacturer and the OS and applications provided by the vendors. In fact, the Smart Card OS can be developed by the vendor or a third party company. For single application devices, the vendor usually has its own OS. For the case of multi-application Smart Cards, using the OS developed by third party companies such as JavaCard, MULTOS or STARCOS is more common.

Single application devices tend to be built around 8-bit or 16-bit CPUs, whereas multi application devices can make a better use of 32-bit CPUs. Both OS need to be secure, where the application and OS memory spaces are kept separated. In the case of a multi application OS, however, this memory space differentiation expands to applications, where an application cannot access other applications' data, as otherwise, an application developed by a hacker could potentially access other applications' data or even the OS data. In some cases, as with multi-application Smart Cards, applications can be added or removed even after the device has been deployed and while it is being used by the end user.

2.3 Threats

Despite all the security measures embedded into a Smart Card device, they are still targeted with a range of attack techniques to get valuable data. This section explains those threats directly related to the work carried out in this research.

2.3.1 Glitch Attacks

Electronic devices are designed to be powered at a certain, constant voltage level that is characterised by the technology node at which the device is built. By powering the device outside its specification values, the device could fail to operate as expected or suffer permanent damaged. This is especially true when the device is subject to abnormal voltage levels for a prolonged period of time.

Electrical noise, such as glitches, are sudden temporary changes of the voltage level in the power rails of a device. Due to the short and temporary nature of this phenomenon, electronic devices can withstand a certain level of electrical noise; a typical specification is +/- 10% of Vcc. However, electrical noise or glitches above certain level can inject errors, temporary malfunctions or damage a device permanently.

Glitch attack is a technique used by attackers to disrupt a device's normal operation in order to analyse the device's behaviour or gain access to otherwise secret data. These attacks are applied on the target device's power rails, Vcc and/or GND in the case of Smart Cards. An alternative use could be targeting the device's I/O or clock signal. A typical outcome of a successful glitch attack could be skipping the execution of an instruction.

Despite the ISO standard defining the valid supply voltage levels, Smart Card manufacturers have no control over the environment where their devices are used or how they are powered. In other words, Smart Cards are exposed to the environment and potential abuse from an attacker which, does not need to follow the ISO standard nor the device's powering specifications. Hence, they need to make their devices robust against such attacks by design.

As already mentioned, Smart Cards include a voltage regulator that converts the standard supply voltage level into that supported by the technology node it is built in. In addition to that, the built-in voltage regulator also provides an inherent protection against certain glitches and electrical noise levels, as it filters out some of the undesired fluctuations present at its input, providing a cleaner supply voltage at its output. This filtering feature, as it will be shown in section 3.2.3.3 Test-chip Test and Results, is highly design dependant. Despite this protection, some glitches are still capable of injecting faults into the Smart Card device. Figure 2-3 shows the typical impact pattern of positive glitches[1] applied to the Vcc of a Smart Card.

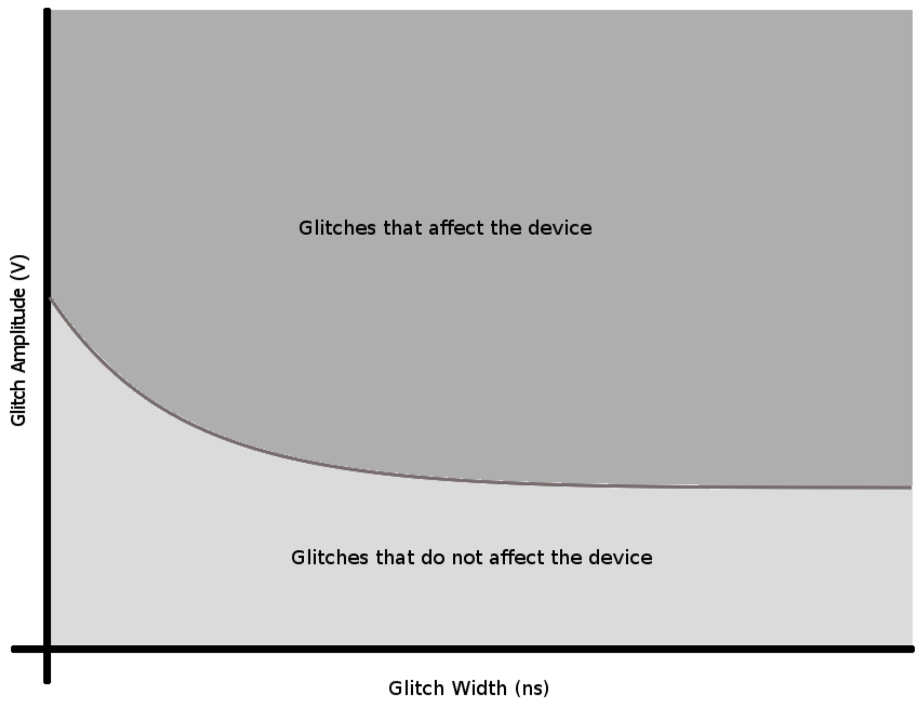


Figure 2-3 A Smart Card's typical response to glitch attacks

As a countermeasure to this threat, Smart Cards include glitch detection circuits located at the input and output of the voltage regulator. The aim of these glitch detectors is to detect voltage levels and glitches that violate the supply specifications and that can pose a threat to the Smart Cards' correct operation whilst avoiding the detection of permissible noise.

The detection range of a typical detector should not only focus on the region that affects the device, but it should also overlap it with the region that does not affect the device. In other words, a detector should aim at detecting those glitches that affect the device and those that are close to affecting it. Figure 2-4 shows a typical detection range of glitch detectors to positive glitches on the Vcc of a Smart Card device.

Two research lines of this work have focused on glitch attacks. Firstly, a simulation environment has been developed to test devices or parts of them against glitch attacks,

[1] Several kind of glitches can be applied. These are explained further in the Section 3.1.2.2 of this Volume.

section 3.1. Secondly, a glitch detector capable of detecting certain fast glitches has been developed and patented, section 3.2.

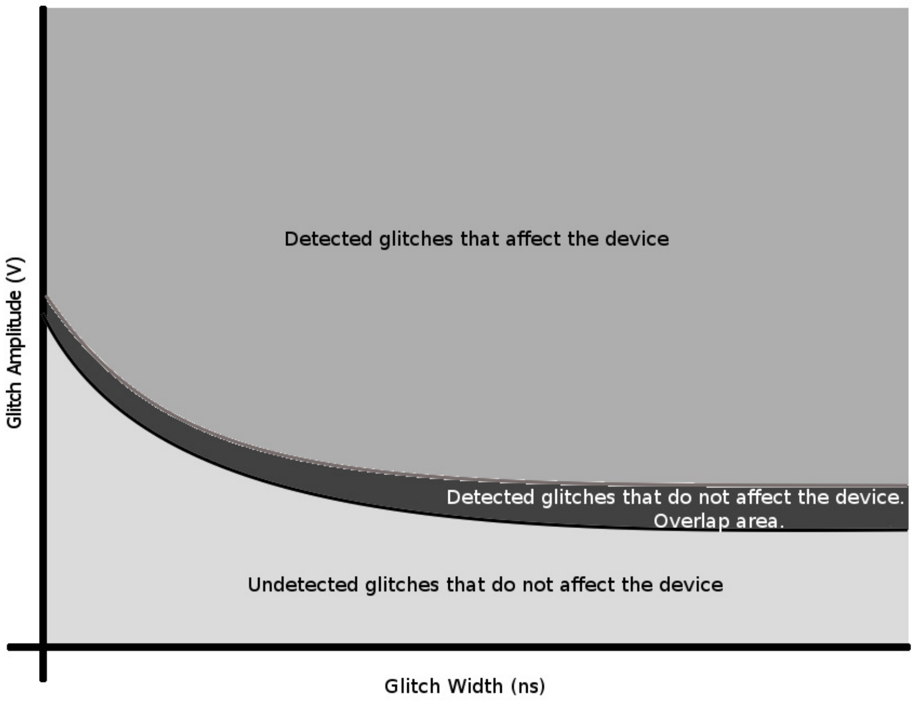


Figure 2-4 Typical aimed detection range

2.3.2 Power Analysis

As previously stated, DES and AES are encryption algorithms commonly used in Smart Cards. The mathematical models of these standards are strong and unbreakable, that is unless brute force is used. The DES algorithm, with its 56-bit key length (2^{56} or over 72 quadrillion key combinations), is generally seen as a weak cipher, as it is relatively easy to break by a brute force attack [2]. A very common DES enhancement is triple-DES (3DES), where encryption is achieved by a consecutive chain of encryption-decryption-encryption, using different keys for each operation, as shown in Figure 2-5. As a result, the 3DES has an effective key length of 168 bits. The 3DES is regarded as safe against brute force attacks. In the case of AES, no successful brute force attack has been published to date for any of its three possible key lengths, 128, 192 or 256 bits.

However, whilst the mathematical model of these algorithms might be secure and robust, its hardware and software implementations (which ultimately depend on hardware) can introduce a degree of weakness. Complementary Metal-Oxide Semiconductor (CMOS) devices consume static and dynamic power. Static power is produced by the leakage current present when a device is powered. This leakage current depends on the technology and the number of transistors in the design, where the device activity has little or no impact. Dynamic power, on the other hand, is produced by the dynamic current consumed by the device. Dynamic current occurs when the device is active and is produced as a result of transistor switching in the device, more switching meaning higher currents. Since the switching of transistors in a design (e.g. a combinational block) depends on its current status and the data being processed, it can be said that there is a link between the consumed dynamic power and the data being processed. In other words, processed data can be leaked.

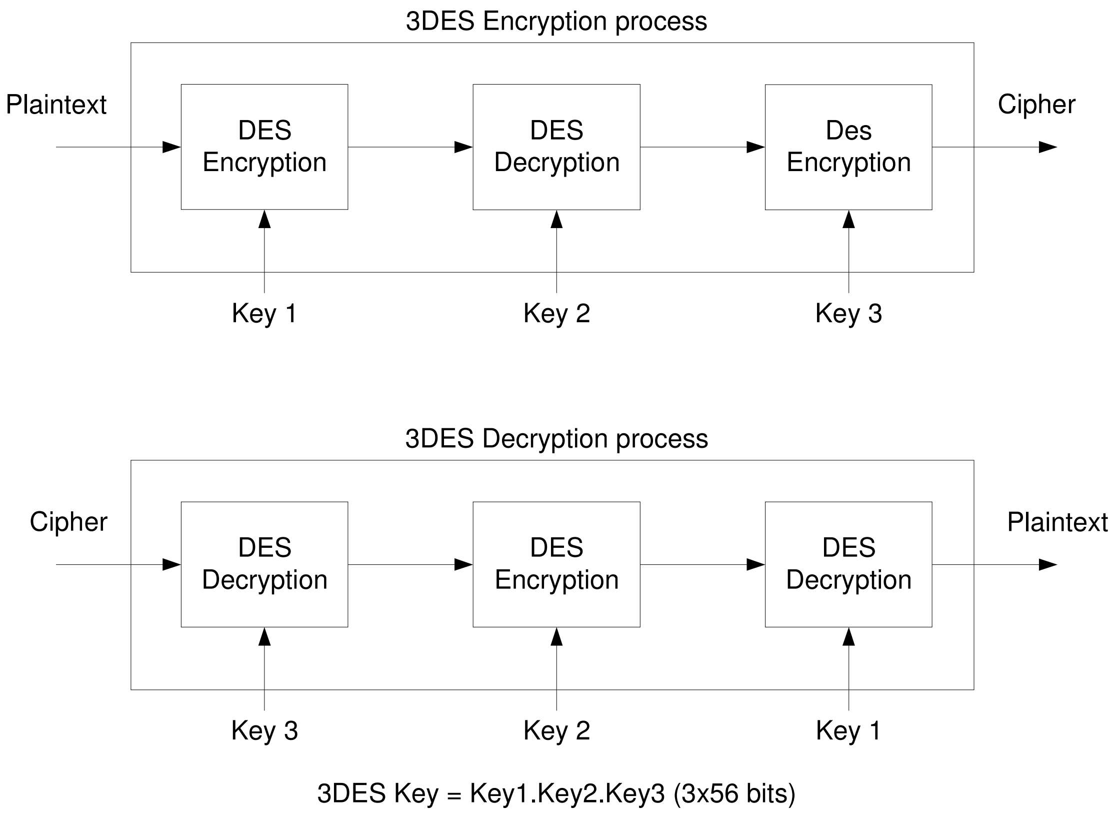


Figure 2-5 3DES encryption and decryption processes

Power analysis techniques such as simple power analysis (SPA) and differential power analysis (DPA) exploit this link between data and current consumption to guess or estimate the operation being performed by a device or even the data it is manipulating. SPA techniques, which directly monitor and analyse the power consumption of a device, can be used to identify the instruction executed by a CPU [3] or even the operand of an operation [4]. DPA techniques, which perform a statistical analysis on a series of power traces, have been used to successfully guess the secret key of cryptographic algorithms such as DES and AES [5, 6].

Several countermeasures have been proposed against SPA and DPA. One research line of this work focused on designing a re-configurable instruction decoder block (RIDB) to protect the CPU from SPA and code leakage, Chapter 5. Another research line has focused on developing a simulation environment to test DES modules against DPA, Section 3.1. This simulation environment has been integrated with the glitch attack simulation environment.

2.3.3 Laser Attacks

Lasers are used in the aerospace industry to test electronic devices at terrestrial level for fault injection events that they might face at high altitudes and/or outer space due to radiation [7]. Despite the fact that a laser beam cannot inject the full range of faults that radiation particles can, its capability to inject faults known as single-event upset (SEU) is of great use to an attacker, and using it correctly could result in obtaining valuable information.

The cost of the setup to inject faults with a laser, can vary as much as the range of laser devices that can be used for this purpose. At the lower end of the scale, a photoflash was successfully used to flip the value of an SRAM bit on a PIC microcontroller [8]. This attack was carried out by focusing the light spot on the SRAM bit-cell it was targeting. This attack could be used to change the value of a register indicating the state of an operation, an FSM or a loop counter. In theory, this same attack can be applied to Smart Cards, however, and due to the address and bit scrambling of the memories that takes place in a Smart Card device, the effect of a similar attack could be unpredictable and also hard to repeat.

An alternative use of the laser is to scan the whole die to identify the location of specific functions such as the arithmetic logic unit (ALU) of the CPU, the CPU registers, the cryptographic block or memories. After that, an attacker could focus on the area of interest to attack it.

The main drawback of the laser is the laser beam's penetration level. Laser beams are only capable of injecting faults on exposed areas of silicon, as the laser beam can be stopped by physical obstacles such as packaging and metal layers. So, in order to be able to carry a laser attack, the target device needs to be de-packaged first and no metal tracks or layers should cover the target area.

The designs' complexity level and the amount of layers used on modern Smart Cards makes it difficult to achieve successful front-side attacks (through the metal layers), as metal tracks often hide the silicon below. This task is made even more difficult if the metal mesh on the top layer to stop from micro-probing is taken into account. However, these difficulties can be overcome with back-side attacks, where the laser is applied to the substrate, which could have been previously milled. Hence, the threat imposed by the laser is still present.

This research line focused on the design of countermeasures to harden devices against light attacks. This research was cut short due to a shift of interest by the sponsor company. However, a test-chip with some countermeasures was produced and tested. More about this research topic can be found in Section 3.3 of this Volume and, at a deeper level in the Technical report LaserTech1 of Volume 2.

2.3.4 Program code leakage

Smart Cards typically use commercial CPUs or secured versions of commercially available CPUs. Since commercial CPUs are better known than specific or non-commercial ones, developers might already be familiar with them and more development tools might be available. This could help with the code development. However, this also makes them sensitive to program code leakage, as an attacker gaining access to the opcode or assembler could easily understand it. Furthermore, different devices with the same CPU might have similar instruction power signatures, which make the Smart Card more sensitive to SPA like the ones used in [3] and [4]. Again, RIDB and other CPU de-standardisation techniques focused on minimising the impact of this threat.

2.3.5 Security challenges introduced by new technology nodes

Newer technology nodes result in higher static power drain due to an increase of leakage current. Memories are the single highest source of leakage current on Smart Card devices due to their high transistor density. Several leakage reduction techniques have been developed to reduce this leakage current, which mainly focus on reducing the SRAM's supply voltage [9-11]. A lower supply voltage means a lower static noise margin (SNM), which can impact on the memory's robustness and sensitivity to soft errors [12].

Smart Card devices are products designed for secure applications; reducing the power consumption could not be done at the expense of security. This research line's focus was on investigating the security impact of different low power techniques, some of which were developed by Atmel's Memory Group based in Rousset (France), Chapter 4.

2.3.6 General comments on security and attacks

The above mentioned attacks and threads can be more or less effective on their own depending on the countermeasures instantiated into the Smart Card devices. However, combining the above attacks or threads, [3, 13], can provide a hacker with new and more complex tools to circumvent security measures and/or inject faults into the Smart Card.

A common corollary in the computing industry says that ‘the most secure device is the one that cannot be accessed’. In reality, however, such device would be of little, if any, use, as real world devices are designed to be used and, hence, accessed. From the minute a device can be accessed, implementation faults, side leaks or human errors could result in exposing such a device to attackers or unintended use of the device.

Security companies can and do invest millions of pounds securing their products and systems. However it does not mean that those security measures are unbreakable. In some cases, the security can be broken with a heavy financial investment and/or with time and dedication, but on other occasions, thinking out of the box can provide surprising results too and bypass secure technology with a small investment [14].

In the Smart Card Industry context, a device is considered secure when it meets the Common Criteria rules related to its target application. Although this does not guarantee absolute security, it does provide a degree of assurance that a certain methodology has been followed during the device’s development and that the device meets the security requirements for the target application. Also, devices are often considered secure when the security measures cannot be (are not expected to be) broken within the device’s life time, or the cost of doing so outweighs the benefits obtained by breaking the security measures.

With regards to attacks, one could think that the aim of an attack is to obtain some tangible data. For example, that as a result of a glitch attack, the attacked device would provide the attacker with the value of a given register, or hand the attacker full access to the device or dumping its code. Although possible, this is often not the case.

A more common effect of an attack can be interrupting a process or transaction being performed by the card, and the handling of this interruption might be providing valuable information to the attacker. Hence, detected attacks should not abort the process, but only provide a wrong result, that should be handled by an error handling protocol put in place for that purpose.

2.4 Portfolio contribution

Smart Cards can be challenged in several ways and therefore device security needs to be tackled from several fronts. The change of focus by the sponsor company during this EngD has resulted in a portfolio of different research areas targeted at improving the security of Smart Card devices. These research areas have resulted in the following contributions:

- Development of a **simulation tool** to test designs against glitch attacks and power analysis attack during a design's development. This tool could easily be integrated within Atmel's design flow to bring assurance of their designs' behaviour and permeability to such attacks. Power analysis attack can be either simulated or tested in a physical device. While physical analysis was out of the scope of this work, no simulation environment reported being able to perform different attacks.
- A **new glitch detector**. By changing the detection approach, the design proposed here achieved detecting glitches that current detectors do not.
- The use of **radiation countermeasures as laser countermeasures and potential sensors**. Radiation and laser attacks have similar effects on silicon devices. Whilst several countermeasures against radiation have been developed over the years, almost no explicit mention of laser countermeasures was found. This research has demonstrated the suitability of using some radiation countermeasures as laser countermeasures.
- **Memory partitioning applied to Smart Cards**. Many techniques to reduce leakage have been proposed during the last few years. However, no work was found on how these techniques can be used on Smart Cards, where security is priority number one. This research line has resulted in an implementation proposal of memory partitioning focused towards Smart Cards.

3 Fault Tolerant Design

This chapter covers the work carried out in relation to fault tolerant design. Three fronts or research lines were covered in this topic: a) the development of a glitch attack and power analysis simulation environment (GAPASE); b) the design of a glitch detector; and c) the test of radiation countermeasures when targeted with laser.

Since these three research lines are highly independent from each other, the work carried out on each research line will be covered independently in three different, self-contained, sections. These sections include: an introduction to the research; the literature review related to that specific research line; an explanation of the work carried out and the results; the conclusions drawn in relation to that particular research; and future lines of actions.

3.1 Simulation Environment

Devices are designed according to specifications that define their behaviour and operational range. A device must meet the behavioural specifications for as long as it is within its operational range. When a device is subjected to operation conditions outside its specifications, however, its behaviour is no longer guaranteed and might result in temporary fault injections or even in permanent damage.

Electronic Design Automation (EDA) tool vendors provide tools that allow developers to simulate different aspects of their designs at different stages in the development process. These simulations can verify the design’s correct behaviour and check for timing violations, power consumption and area. All these simulations provide developers with a degree of confidence that their design will work in silicon as expected. However, for the most cases, these simulations are performed within the device’s operational range.

Smart Cards can be attacked by forcing them to work out of their operational range for a short period of time: a) by raising the supply voltage over the absolute maximum; b) by over clocking the design; c) by subjecting it to extreme temperatures; d) by subjecting it to a source of radiation; or by applying glitches in the power source and/or signals [3, 15]. White paper [15] is included in the Appendix A.

Devices can be tested after their fabrication for these operation conditions. In fact, they are tested by external evaluators when certifying their security grade. The issue with this approach, however, is that these tests occur too late in the design flow, as fixing any security flaws detected at this stage will incur in high costs, as new lithography masks might be needed. In other words, it is not cost effective.

Hence, Smart Card manufacturers and security product manufacturers in general need tools capable of simulating a device's behaviour under abnormal operation scenarios. The simulation environment developed in this research line has focused on glitch attacks and power analysis.

By integrating GAPASE within the Atmel's design flow, designers are enabled to test their designs and to know how they perform when attacked. The following sections cover the literature review; the simulation environment developed in this research line; a discussion on its usability and the conclusions of this research line. Finally, some possible future developments are proposed.

Volume II includes the following reports related to and supporting this research line:

- SimEnvTech1 (Glitch Attack and Power Analysis Simulation Environment);
- SimEnvTech2 (Counter Simulation Results);
- LaserTech1(Tartalo test-chip 01OKA);
- GlitchTech1(Glitch Detector Report)

3.1.1 Literature Review for Fault injection and side-channel

There are several factors and features that need to be taken into account and decided when designing a simulation environment. Some of these factors include the simulation level, the simulation model, the kind of injected faults and the amount of circuitry to simulate.

In relation to the simulation level, the typical debate is high level simulations vs. low level simulations, where the usual trade off is speed vs. accuracy. High level approach was taken in the glitch attack simulation environment in [16]. Developed around SystemC, this simulation environment allows simulating the propagation of a fault injected by a glitch attack. Since this simulation environment runs high level simulations (i.e. behavioural level), it is capable of both simulating complex circuits and faster than with lower level simulation tools.

Simulating a glitch attack in SystemC has, however, other kind of limitations, such as the need for a behaviour model of a glitch attack and or presuming where the fault is injected. However, the recent publication of the analog and mixed-signal (AMS) extension of SystemC (SystemC AMS) [17] could help modelling a circuit's behaviour when subjected to glitch attacks and, hence, ease or minimise these drawbacks and or limitations.

Low level simulation tools such as Simulation Program with Integration Circuit Emphasis (SPICE) or Nanosim, on the other hand, might be slower to simulate, but they are more adequate to simulate analog circuits and, hence, the response of a circuit to glitch attacks. As a result, these simulation tools could indicate more accurately where a fault is injected after a glitch attack and how the circuit under test responds to such fault. Ultimately, this information could be used to enhance the security of the affected region.

Equally, the power analysis simulation environment could be subjected to the same debate, high level simulation vs. low level. Since power consumption is the key to guess the encryption key of DES and AES cryptographic modules, the natural assumption could be to use a simulation tool that provides an estimated power consumption waveform, be it simulated at gate level or transistor level.

Work [18], however, demonstrated the feasibility of register transfer level (RTL) simulations to carry out a successful power analysis to AES cryptographic modules. The key to their success was the definition of the power consumption model. This was the hamming weight of the target substitution box's (SBOX) output, which is the building block of DES and AES cryptographic algorithm. Such power consumption model produces a single value instead of a waveform per encryption.

This simulation level and approach has an immediate impact on power analysis speed as, on one hand, RTL level simulation will always be faster than lower level ones. On the other hand, producing a single value instead of a waveform reduces the amount of time the simulation tool spends in writing to output files. Furthermore, and as a direct result of having a small power trace data, the power analysis script will run faster than if the output was a waveform.

The simulation environment in [18] needed just 400 plaintexts to guess the key of one SBOX of an AES instantiation without countermeasures. Despite this number might be above the theoretical minimum (256), the time required to encrypt the additional plaintexts could be assumed to be short enough to justify this simulation environment.

The main drawback of this simulation environment, however, is the potential limitation to test various countermeasures. These are the reasons: a) high level simulation ignores layout and parasitic information; and b) this simulation environment only simulates the required minimum part of a cryptographic module for DPA simulations [18, 19].

By simulating the required minimum logic, this simulation environment focuses on one SBOX at a time and can only test countermeasures related to the simulated logic. Testing other countermeasures, such as dummy cycles, could still be possible with this simulation environment by emulating these cycles in the stimulus generation step. However, this simulation approach excludes itself from testing certain countermeasures such as those that aim at protecting the algorithm.

This high level simulation environment could also fail to be of any use to test so called constant power consumption countermeasures such as dual-rail [20] and wave dynamic differential logic (WDDL) [21] unless new power consumption models are created.

At the other end of the simulation tools' scope is work [22], which was based on SPICE. Like for [18], this simulation environment only simulated the required minimum logic to carry a DPA, inheriting the same drawbacks mentioned before in regards to its ability to test certain countermeasures.

This simulation environment needed 100 plaintexts to guess the key of the target SBOX of a DES module without countermeasures, which was designed at the transistor level – all equally sized and by hand – and based on domino logic. This kind of SBOX design provides a clear and clean power signature, which facilitates the power analysis, and is contrary to the implementation approach followed by almost all digital designs, which consists of designing the IP in RTL using a hardware description language (HDL) and then synthesising it to a gate level netlist. The designs resulting from this later approach produce a higher number of gate switching, that is, a less clear signature.

Another, more recent, simulation environment is that covered in [23]. This simulation environment also relies on SPICE as a simulation tool with some promising results, as it can successfully guess the right key associated to a DES or AES SBOX implementation with no countermeasures within just 2 minutes.

No simulation environment was found to cover both attack methods or threats, perhaps due to the distinctive simulation tools used for each purpose. The simulation environment developed in this research line, however, does just that. It targets the simulation of both threats using the same low level simulation tool, Nanosim. Nanosim is a SPICE like simulation tool that trades accuracy for speed.

Unlike with [18], [22] and [23], the simulation environment developed here targets the whole DES module, which enables the possibility of testing a wider range of countermeasures. On the other hand, despite the higher accuracy of the generated power trace means the need for less plaintexts, the simulation speed of the simulation environment developed here might not get close to that of the simulation environment [18], especially considering that the GAPASE simulates the whole DES and [18] only focused on the minimum required logic.

3.1.2 Glitch Attack and Power Analysis Simulation Environment

This research line has resulted in a simulation environment capable of testing designs against glitch attacks and power analysis. Briefly, it could be said that GAPASE is actually a wrapper around a simulation tool (currently Nanosim). GAPASE reads a configuration file and, from it, it generates the required input files for the simulation. Once the simulation is completed, the output data can be post-processed depending on the targeted simulation. A block diagram of the current GAPASE environment is shown in the Figure 3-1. In addition to the configuration file, GAPASE needs the design netlist and stimulus files as inputs from the user before launching the simulation. It also needs to know the location of SPICE devices' model library. The path to these files is indicated in the configuration file.

The following sub-sections cover the current features of GAPASE and its implementation.

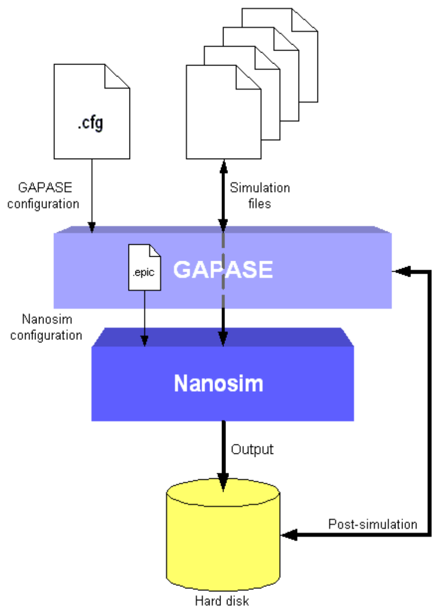


Figure 3-1 GAPASE block diagram

3.1.2.1 Features

This is a list of the features of the current version of GAPASE:

- GAPASE has been designed in a **modular architecture**, providing flexibility and enabling future extensions.
- It currently supports **Nanosim** as the only **simulation tool**. Only SPICE netlists of the target designs can be simulated. Back-annotated netlists with parasitic information can also be loaded.

- The simulation can be detached from the console to allow the simulation to run in the background, i.e. even if the user logs out.
- Simulations are **configurable with up to 51 parameters**, divided in three groups: parameters applicable to all simulations; parameters applicable to glitch attack simulations, and parameters applicable to power analysis simulation. The full parameter list is available in Appendix B.
- GAPASE targets three kinds of simulations:
    1. **Normal simulations**, which are effectively simulations without any disturbance. These simulations only use general configuration parameters.
    2. Simulations of a circuits' response to **glitch attacks**. These simulations use general and glitch specific configuration parameters. Glitches and noise can be applied to any input source of the design under test.
    3. **Power analysis** simulations on DES cryptographic modules. Power analysis simulations use general and power analysis related configuration parameters. The GAPASE environment is used to generate power consumption traces that are later analysed in Matlab on a post-simulation step.

3.1.2.2 Implementation

At the first instance, the requirement was to develop a simulation environment capable of running a glitch attack simulation. Since the associated tasks for such simulation were not too complex, this initial simulation environment was achieved with some Bash scripts. The later requirement to add the power analysis feature, however, resulted in the need for a major rework. This was carried out by porting the whole simulation environment to Perl and adopting a modular approach by using Object Oriented Perl programming style –currently totalling about 2,800 lines of code. Figure 3-2 shows a layer diagram of the latest implementation.

In its current implementation, GAPASE can be divided in three layers: 1) the core layer at the bottom; 2) the simulation specific objects layer (SSOL) in the middle; and 3) the post-simulation layer (PSL) at the top. The core layer carries out some generic operations, such as parsing the configuration file, checking and generating general parameters and files. On top of that, it also provides an application protocol interface (API), so that the layers above can send messages to a log file and access simulation environment variables. This layer also provides the simulation_type class, which defines the methods to be inherited by the objects in the SSOL.

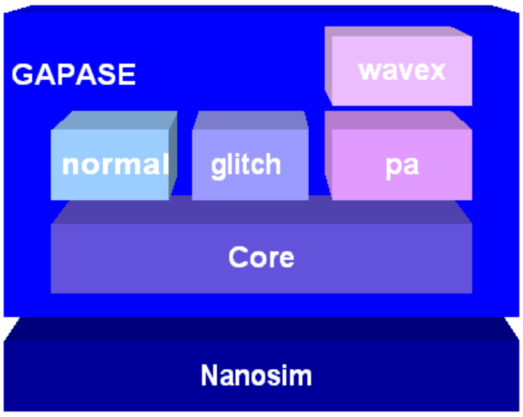


Figure 3-2 GAPASE's layer division

Each simulation type is instantiated as an object that inherits the methods in the simulation_type class. Some of the inherited methods are modified by the target simulation object. The function of these objects is to check the parameters specific to the target simulation, to create support files for this task and to generate the commands that would launch the simulation.

Post-simulation layer currently implements just one module. When Nanosim completes the simulation, this module is called by the power analysis object to extract the power consumption waveform into individual power trace waveforms. These power traces can later be used by Matlab scripts to perform DPA by either a difference-of-means or correlation power analysis.

The process of running a simulation is as shown in the diagram in Figure 3-3. The user would first edit the GAPASE configuration file, which defines the simulation conditions as well as pointing to the location of the required input files. After this, GAPASE could be launched and the whole process started. Initially, GAPASE would load all parameters and check the generic ones. Next, simulation specific parameters will be checked, and if any error is found in any parameter, the simulation is aborted. The user can check the errors in the log file. If, on the other hand, no errors are found, GAPASE will create first all generic input files followed by simulation specific ones. At this point Nanosim simulation will be launched and no further action will be taken by GAPASE until the simulation is finished. After the simulation, GAPASE will gain control over the environment. If the target attack was power analysis, the post-simulation functions would be executed, wavex in this case. Otherwise, GAPASE will terminate and the user will be able to check the results. The section 4 of the report SimEnvTech1 covers this flow in more detail, including a description of the configurable parameters shown in the Appendix B.

Regarding the GAPASE's internal operation, it powers the circuit under test by applying the supply voltage value specified in the configuration file (VOLTAGE parameter) to the circuit's power rail. The default power rail (VOLTAGE_NODE parameter) is Vdd!. This is the device's internal supply rail, once rectified by the voltage regulator. Setting VOLTAGE_NODE to Vdd! allows a faster simulation by avoiding the simulation of the voltage regulator. However, GAPASE also allows simulating the voltage regulator if required.

The simulation environment allows applying glitches to both power rails, Vdd and GND, of the circuit under test. A built-in glitch generator function allows the user applying any glitch definable with up to four points. The first and last points of the glitch would usually represent Vdd! or GND! just before and after the glitch. Examples of the glitches that can be applied with the built-in glitch generator function are shown in Figure 3-4. For more complex glitches, repetitive glitches and or noisy supplies, GAPASE allows defining power supply waveform vectors. Noise and glitches can also be applied to any input signal through waveform vectors.

The power analysis is divided in two parts; power trace generation and power trace analysis. GAPASE is responsible of generating the power traces, which are later analysed in Matlab. Currently, the power analysis is only enabled for the DES cryptographic algorithm, and the whole DES design is simulated regardless of the targeted SBOX.

Despite DES having 64-bit data and key registers, the CPU and the DES module communicate via an 8-bit wide interface. The implication of this interface is that in order to run an encryption, 16 clock cycles are needed in order to setup the plaintext and the key. GAPASE avoids this non-productive simulation by directly forcing the input or output of data and key registers. To further save on simulation time, GAPASE can also limit the number of rounds to be simulated, as not all rounds are used to perform DPA. Typically, only the first two rounds are simulated. After these rounds have been simulated, the DES block is forced to a reset status before the next plaintext encryption process is started.

Finally, only SBOX1 can be targeted on the current implementation and, consequently, only the 64 plaintexts that exercise this SBOX should be used during the simulation. The processing of all 64 plaintexts is based on the Atmel's security group approach. This approach consists on feeding all plaintexts in a sequential order, from the smallest to the highest hexadecimal value. Each plaintext is followed by a reset status. This set of plaintexts can be simulated several times one after the other.

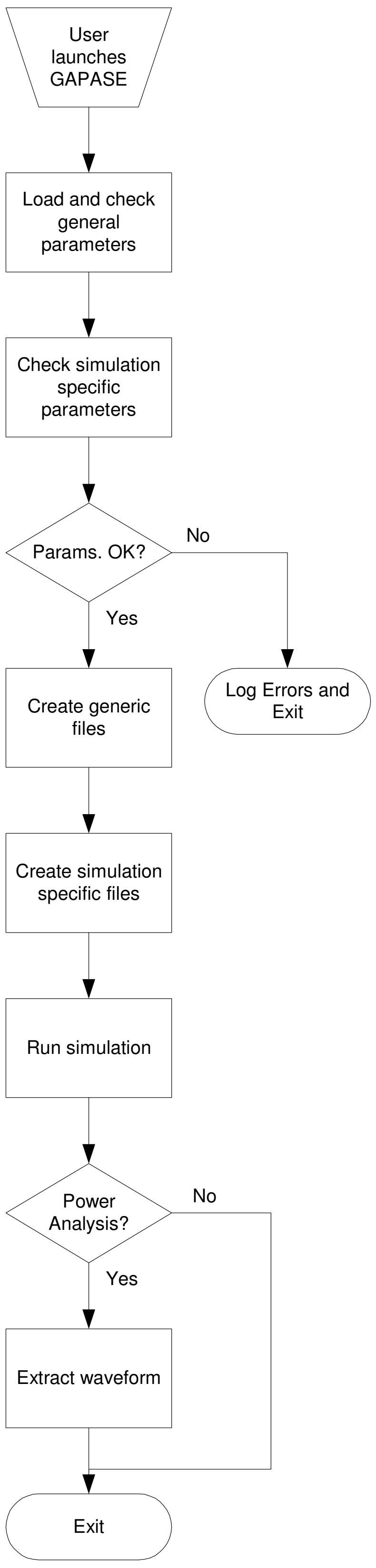


Figure 3-3 Simulation flow diagram

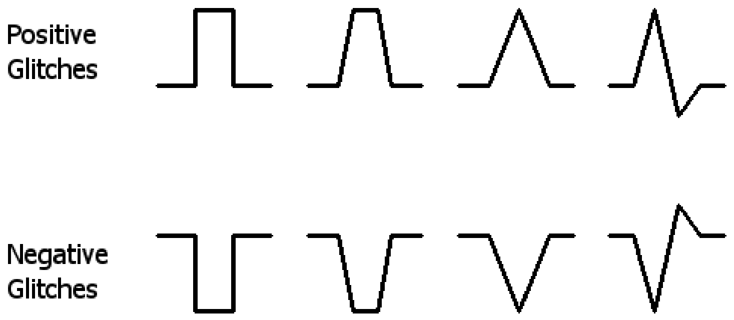


Figure 3-4 Examples of glitches that can be defined with GAPASE

After the simulation is completed, the post-simulation module is launched so that the recently generated simulation-long power trace can be divided into individual power traces corresponding to each ciphered plain text. These power traces are then analysed in Matlab. Current Matlab scripts allow carrying out two type analyses on this data, difference-of-means (DOM) [5] and correlation power analysis (CPA) [24]. Matlab scripts to perform DOM are based on the scripts developed by Andrew Burnside for the Security Group, whereas CPA scripts were developed exclusively for this simulation environment. These Matlab scripts can be found in the Appendix C of the report SimEnvTech1.

3.1.3 Discussion

The initial requirement of a glitch attack simulation environment, where the focus was on finding out how a glitch attack can affect a circuit or a design, implied the use of simulation tools capable of simulating the analogue behaviour of digital designs. This meant moving away from RTL or pure logic level simulators where the information they work with is either a logic 0 or a logic 1.

Two SPICE based simulation tools were available at Atmel, HSPICE and Nanosim. Nanosim trades accuracy for simulation speed; in fact, an HSPICE simulation of a glitch that takes over eight hours to complete, it could be completed in around half the time when simulated in Nanosim. Since the simulation speed was a priority, Nanosim was chosen over HSPICE as a simulation tool. Still, due to the input stimulus compatibility between both simulation tools, GAPASE could be easily adapted to simulate on HSPICE when higher accuracy is required.

Creating a glitch attack simulation environment around a SPICE based simulation tool is not very hard. Proof of it was the fact that the initial version was developed as a Bash script. The main requirements for the script were being able to define: a) the target design; b) the SPICE device model; and c) an easy way to define glitches.

Rather, the limitation of the glitch simulation environment is the SPICE device model itself. The behaviour of passive and active elements of a given technology node, e.g. transistors and resistors, are modelled so that they can be simulated using tools such as SPICE. These elements could be modelled to a wide range of operation parameters, such as temperature ranges from -40 to 125 or Vds of a transistor from 0V to 20V, even when its normal operation value is around 1.6V. However, since the use of these elements in such a wide range is unrealistic and the modelling very time consuming, in reality, they are accurately modelled for a short operation range only.

The impact of a limited model is that, out of range simulations might not be accurate enough and as a result, unrealistic glitches might be needed to inject a fault. This was the case when simulating test glitches for low and medium simulation accuracy levels in the report SimEnvTech2. On these cases, Vdd! should be raised as high as 6.4V in order to be able to inject a fault. However, on real silicon and SPICE simulated behaviour of a voltage regulator, on the other hand, Vdd! would rarely rise above 3.5V or 4V, see Figure 3-15 and Figure 3-16. Section 5.1 of the report SimEnvTech1 covers the work made on validating the glitch attack feature of GAPASE, which includes the design of the test-chip covered in LaserTech1.

Another aspect to consider when setting up a simulation environment is the resistivity in the power rails, which results in an uneven distribution of the power supply fluctuations. This feature could create circumstances where nearby circuitry is powered at voltage levels different enough to inject faults. This is a feature not simulated by GAPASE, among other reasons, because taking into account the resistivity in the power supply would increase the complexity the target design's mathematical model and it would take far longer to simulate.

Later simulations fixed the unrealistic glitch issue by setting the following simulation parameters: set_sim_eou sim=4 model=4 net=4. Subsequent simulations produced results comparable to real glitch attacks.

Regarding the DPA feature of the simulation environment, several power analysis attacks were performed with GAPASE on a DES module with no countermeasures, commonly known as a vanilla implementation, and on a secure DES module with a countermeasure. Simulations were run with different simulation resolution settings (time and current amplitude) and encryption key values.

Table 3-1 collects the main power analysis simulation results, which are expanded in more detail in the section 5.2 of the report SimEnvTech1. GAPASE has shown the feasibility of correctly guessing the key bits associated to the SBOX1 of a vanilla DES cryptographic module. The electrical noise free environment provided by GAPASE allows guessing the key bits associated to the SBOX1 using only the theoretical 64 plaintexts.

When performing DPA on a silicon implementation of a vanilla DES module, one or two order of magnitude more plaintexts might be needed to guess the key bits associated to the SBOX1. The need for additional plaintexts is to filter out the electrical noise introduced by: a) measurement instruments; b) electronic components; and c) other IPs in use within the target device, such as a CPU. This difference in the plaintexts requirement to crack the key was also shown in the RTL based DPA simulation in [18], where 10 times more plaintexts were needed in silicon than in simulation to crack the key.

Table 3-1 Power analysis simulation results

DES module	Key	Number of plaintexts	Samples per trace (s/t)	Simulation time	Used CPUs	Right key guess
Vanilla DES	2E3A44BB4248D774	64	30000	13h12m	1	✓
			75	26m	1	✗
			3000	6h12m	1	✓
			75	16m	1	✗
	940AEBA0604CF479	64	3000	4h58m	1	✓
	83373F5D2CF55BC8	64	3000	5h29m	1	✗
Secure DES	2E3A44BB4248D774	64	3000	14h10m	1	✗
		1024	3000	2d2h53m	4	✗
	0000000040008010	1000	3000	2d2h32m	4	✗

The discrepancy between the required amounts of plaintexts to crack the key can only reflect one thing. While the typical behavioural simulation is run to model a circuit's behaviour as accurate and realistically as possible, DPA simulation results should not be taken as absolute ones, but as an index or reference of how secure a particular DES implementation is.

Any DES implementation that takes more than 64 plaintexts to compromise its key will, inherently, be more secure than the vanilla implementation. However, the important information is not only whether a given implementation is more secure or not than the vanilla one, but by how much and how does it translate into silicon implementations.

For instance, given a DES implementation, say DES Secure1, that takes in simulation 200 times more plaintexts than the vanilla implementation to crack the key, it could be considered 200 times more secure than the vanilla implementation. DES Secure1 could have a security index of 200. The difficulty here is on translating this simulated security index into a silicon security index, which depends on the noise sources of the silicon DPA setup and the technology node of the silicon die.

One approach of estimating the index translation function would be applying the minimum mean square error to a series of DPA results in several DES implementations in silicon and simulation for different key values. Subsequent simulations could use the index translator to estimate the real security index of a given DES implementation. Furthermore, if silicon DPA data is generated, this data could be used to improve the index translator.

Regarding the simulations run with GAPASE and which results are shown in Table 3-1 above, two out of three encryption keys were correctly guesses with the vanilla DES implementation with just 64 plaintexts, whereas simulations on the secure DES implementation failed to guess the encryption key. The encryption key that was not correctly guessed in vanilla DES simulation is 83373F5D2CF55BC8. Perhaps, this failure could be explained by the fact that, for this encryption key, all the key bits to be used in the SBOX1 were logic zero, causing a particular case were the analysis failed. This case needs to be studied further in order to determine whether more plaintexts are needed for these cases, higher simulation accuracy or the power analysis algorithm needs being improved. The real impact of resetting the DES module after each encryption should also be analysed.

A total of 2024 random plaintext encryptions with a secure DES were simulated without guessing the right key. At first instance, this could indicate that the countermeasure used in this DES implementation results in a security index of over 31 (2024/64). More plaintexts could be used with the aim of guessing the key, say another 3000 or 8000, totalling 5000 and 10000 respectively. There are two possible outcomes: a) eventually guessing the right key and, hence, determining a more accurate security index of the target DES module; or b) still not guessing the right key. This last case is necessary but not enough to prove that a design is immune against power analysis, as it only proves that the correct key was not guessed within a given practical amount of time and plaintexts. Spending any more time on guessing the key is no longer practical.

On the GAPASE case, the practicality is measured in the combination of simulation accuracy and the time required to run the power analysis. So much so, that it can be a key factor when determining its feasibility as a production power analysis tool.

Encrypting 64 plaintexts with a silicon implementation of a DES module running at 50MHz takes under a second, and running enough encryptions to guess the SBOX1 key bits of a vanilla DES would take few minutes. Running the same operation on a DES module within a Smart Card takes longer due to the communication overhead.

Simulating the DES module in GAPASE does not have the Smart Card communication overhead. Still, running successful power analysis attacks simulating the first two rounds of 64 plaintext encryptions with GAPASE took around 5 hours. The accuracy set for these simulations was 0.001ns simulation time resolution, and 3000 sample points per trace (s/t). Lower accuracy simulations did not yield successful power analyses. A 5-hour simulation can be common for certain analogue simulations, but for a vanilla DES module it could be considered excessive.

Adding countermeasures implies the need to more plaintexts to begin with, which increases the simulation time. Furthermore, the increasing circuit complexity of some countermeasures could result in increasing the simulation time per round. This is, having a double impact on the overall simulation time.

This is the case of the secure DES, where 2 days and 5 hours were employed to simulate 1000 plaintexts divided in 4 CPUs. This is, if all plaintexts where simulated on a single CPU, it would have taken about 11 days to complete the power analysis and not guessing the right key bits of the SBOX1. While 5 hour or even 2 day long simulations could be manageable, especially if as a result the key is correctly guessed, running additional 3000 or 8000 plaintexts would take around 6 or 16 days respectively using four CPUs. Taking this amount of time to run power analysis, which is what DES modules with countermeasures might need, is no longer practical. The limiting factor here is the simulation tool, Nanosim, and or its configuration.

The power traces used on real power analysis might be less accurate and with less sample points per power trace than those used here, yet the key can be successfully guessed from them. One approach to speed up the power analysis could be reducing the simulation accuracy and increasing the number of plaintexts or power traces. In fact, the recent work [25] has achieved successful power analysis on Nanosim with resolutions of 1uA and 10ps. The resolutions used in GAPASE simulations are of 1pA and 10ps. With the simulated DES modules consuming in the order of few mA, peaking around 12 to 20mA, reducing the current resolution to 1uA could still produce current consumption values of up to 4 digits. In principle, high enough to carry a successful power analysis.

Reducing the current consumption simulation resolution by a factor of 10^6 would no doubt speed up the simulation process. The work [25] does not provide any information on simulation times, hence, unless new simulations are run, this performance improvement cannot be quantified. Anyway, for this performance improvement to be meaningful, it should reduce the simulation time of generating 1000 power traces with the secure DES module with countermeasures from 2 days to just few hours.

Regarding the simulation tool itself, the Nanosim versions up to Z-2007.03 are not capable of multi-threading the simulation. In other words, a simulation would use only one CPU or core on a multi-CPU platform or a multi-core CPU, which could be considered as an important limitation or drawback when multi-cores are common even on modern low end consumer rated desktops. This multi-thread limitation is the result of the mathematical model of the circuitry to be simulated that is generated by Nanosim and could still affect newer Nanosim version.

The only work around to this limitation and to increase overall simulation time is, dividing a long simulation into smaller ones and launching several simulations concurrently as different processes. This was done when performing power analysis to the DES module with countermeasure.

Finally, Nanosim is an RTL to transistor level simulation tool, allowing the co-simulation of RTL and SPICE netlist. Simulating part of the DES module as a SPICE netlist (e.g. SBOX1) and the rest of the design in RTL could dramatically shorten the simulation time, however, this feature requires an additional license. Such license was not available when this research was carried out, hence, GAPASE does not take advantage of this feature.

The current power analysis approach comes with its share of drawbacks, namely simulation speed or performance, which could limit its use as a production tool. Other power analysis simulation alternatives could be considered, for instance field-programmable gate array (FPGA) emulation. This approach would allow instantiating the target DES module into an FPGA and using the same power analysis setup as the one used for production devices. An indirect benefit of this approach is that any setup improvement or development of new analysis techniques can be easily and equally applied to development and production devices.

However, FPGA and application specific integrated circuit (ASIC) instantiation will yield different power traces. This might not have major implications, other than considering FPGA analysis results as a security index or security degree. Furthermore, FPGA emulation might not be able to test certain countermeasures, certainly not layout based ones.

Coming back to the simulation results, the work done in [18] seems promising too. Similar techniques could be used, perhaps, with tools such as PrimePower, which is capable of generating dynamic and static power consumption waveforms of simulations at gate and RTL levels. PrimePower would result in faster simulation than Nanosim, although loosing on some accuracy. Work [18] showed that this needs not to be an issue though.

Furthermore, PrimePower can simulate the design in RTL and as a gate level netlist. Generating the gate level netlist is a common task for digital designers, and avoids the extra step of generating the SPICE netlist needed by GAPASE, making the workflow simpler. Also, it would allow using simple RTL test benches or even the ones used during the development process as a source of stimulus to generate the power traces for the power analysis. Using PrimePower as a simulation tool would also allow embedding glitch attack and power analysis tests within the actual design cycle and test flow.

3.1.4 Conclusion of the simulation environment

Simulation tools and environments are a requirement for IC manufacturers, as they can help them determining the quality and fidelity of their products. GAPASE allows Atmel designers to test their designs against glitch attacks and power analysis. The glitch attack feature is functional. The power analysis feature needs further improvements in the performance side.

The down side of high level simulation tools, such are RTL, is that they not be capable of testing certain countermeasures, such as those that focus on the layout, e.g. WDDL and differential routing [21], and perhaps dual-rail or similar. Nanosim, on the other hand, is more suited for this kind of countermeasures.

3.1.5 Future work

This simulation environment can be considered to be in continuous development, hence, there is still room for development and improvement, especially on the power analysis side. The following, lines of work are proposed here, divided in what is need and/or desirable at immediate, short and long terms.

Immediate future, the consolidation of the simulation tool. The following points should be accomplished:

- Determine whether the encryption key of a vanilla DES module can still be correctly guessed by reducing the resolution of the current consumption and, if so, determine the obtained simulation performance to test vanilla DES modules and modules with countermeasures. The results of these simulations could indicate the feasibility of Nanosim as a simulation tool for power analysis.
- If applicable, test the performance of co-simulating RTL level and SPICE level netlists in Nanosim.
- Test other alternative tools such as FPGA emulation or PrimePower.
- The simulation environment's performance should be benchmarked against the results from a commercial silicon power analysis tool. The new secure DES block being designed Atmel could be used for this purpose.

These are the suggestions for the **short term** future:

- Enable running power analysis on AES cryptographic blocks. This would make the whole simulation environment more useful to Atmel for their current and future needs.

- Integrate the current Matlab based post-simulation analysis into the GAPASE simulation environment. This would allow running the whole analysis on a single computer, rather than forcing to run the analysis only on computers loaded with Matlab.
- On the post-simulation side, the simulation environment should also generate output data readable by the RisCure™ software, so that readily available post-simulation analysis tools could be used, allowing running other analyses more complex than difference of means and correlation power analysis without having to develop these for GAPASE.

Finally, for the **long term**, following points should be studied:

- Updating the simulation environment to adopt other attacks, such as laser or electromagnetic analysis (EMA). By keeping the simulation environment up-to-date on attack techniques, Atmel would benefit from extending this tool's life.
- Target of additional cryptographic blocks, such as RSA. By adding further targets, Atmel could test other IPs used in their chips against these attacks.
- Designing a graphical front-end. Although this has a lower priority, this feature could make the simulation environment easier to use.

Figure 3-5 shows a block diagram of how GAPASE could be formatted in the long term future.

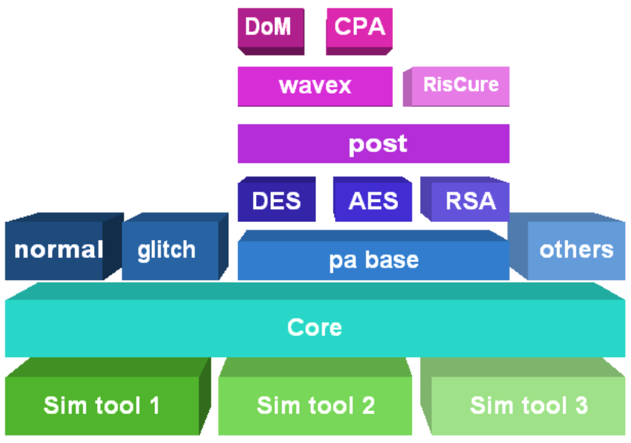


Figure 3-5 Example of future GAPASE

3.2 Glitch Detector

As covered in Section 2.3.1 Glitch Attacks of this Volume, glitch attacks aim at fault injection by means of fast alterations of the supply's voltage level. The effect of these attacks can be minimised by designing a robust built-in voltage regulator, however, this is often not enough or possible, due to the negative impact on cost in terms of design effort, area or price of the final product. Instead, glitch detector circuits are used to monitor the power rails' voltage level to make sure the supply voltage is within the allowed operation range. Glitch detectors are, hence, a fundamental part of the Smart Card design, as in the event of a glitch detection, measures can be taken to correct or avoid the fault [26].

Smart Cards have a built-in voltage regulator, whose output powers the device and, by design, cannot be accessed by an attacker. Hence, glitch attacks on Smart Cards focus on injecting voltage level alterations on the device's Vcc and GND pins. Glitches applied to Vcc or GND can still propagate to the internal Vdd power rail and into the system. This implies two requirements [27]:

1. Voltage regulators design can have a great impact on how glitches are propagated into the system.
2. Detectors have to monitor Vcc and GND power rails as well as Vdd.

There are two glitch detection approaches: a) direct; and b) indirect. The direct approach monitors the power rails to detect supply voltage anomalies as they are happening. The indirect approach detects glitches by monitoring the status of a circuit sensitive to power supply voltage fluctuations. This is, a circuit prone to error injection by fluctuations in the power supply voltage.

Since the width of a glitch (the time it can be present on the power rails) ranges from hundreds of nanoseconds to just few nanoseconds, the direct detectors' response time is key to detect fast glitches. So much so, that glitch detectors with shorter response time, not only can detect fast glitches that those with longer response time cannot, but also, they detect glitches quicker. However, the detection of fast glitches can be challenging with this approach, as making the detector too fast or sensitive could result in detecting legitimate supply noise as a glitch. This leads to false detections.

Unlike with the direct approach, where the supply voltage level anomaly is present only for the duration of the glitch, with the indirect approach, the effect of a glitch on a sensitive circuit can be present for longer that the glitch itself. So, by playing with this circuit's sensitivity, it is possible to detect fast glitches with longer response time monitors whilst minimising false detections. In other words, indirect approach detectors can achieve safer detection of fast glitches but with slower detection times than direct approach detectors. The glitch detector proposed in this research falls into the indirect detector category.

The following sections cover the literature review; the proposed glitch detector; simulations and test carried out on this detector; a discussion on its usability and the conclusions of this research line. Finally, some possible future developments are proposed.

Volume II includes the following reports related to and supporting this research line:

- GlitchTech1(Glitch Detector Report);
- GlitchPub1(Paper: Detecting Voltage Glitch Attacks on Secure Devices);
- GlitchPub2(Paper: Characterization of a Voltage Glitch Attack Detector for Secure Devices);
- GlitchPub3(Patent WO 2008/033712 A2: Detecting Voltage Glitches)

3.2.1 Literature Review

Glitch detectors are key components to protect devices, such as Smart Cards, against glitch attacks. Publishing any kind of information in regards to these circuits can compromise the device's robustness or security. Hence the lack of relevant publications on this topic, especially about current glitch detector circuit designs. Fortunately, two glitch detector patents were found [28, 29], both falling into the indirect category.

The detector in patent [28] detects glitches by monitoring fault injections into single-bit registers and the read operation of a SRAM. These components are usually designed to operate at the silicon technology node's nominal voltage level (e.g. 1.8V for 0.18um technology node, 1.6V for 0.13um, 0.9V for 90nm). Hence, by the components used in this design, it is safe to assume that this detector has been designed to detect glitches at the output of a voltage regulator. This design is capable of detecting positive and negative glitches in Vdd and positive glitches in GND. This is a fast detector, as the fault injection can be detected within/after a clock cycle. The drawbacks with this detector, however, are the fact that it requires a clock signal and the registers' sensibility to glitch attacks.

Simulation results in the technical report SimEnvTech2 demonstrated that glitches could have an instantaneous impact on combinational logic depending on the data being operated at the time the glitch is applied. Storage units, such as registers, however, were only affected when updating data, especially if the data was already corrupted in the preceding combinational logic. The design in [28] shows the registers being fed directly from the power rails. For these registers to latch the wrong value, two factors need to be involved. One is that the voltage difference between the original and the abnormal supply levels need to be high enough to change in logic values. In other words, Vdd should fall low enough to be interpreted as a logic zero or setting the register, at which point the power on reset (POR) could kick in too. The second factor is that the glitch has to happen close to the positive clock edge of the registers' clock signal, which takes us to the other drawback of this detector.

Glitches are asynchronous supply voltage alterations. Despite an attacker trying to apply them in reference to certain time or clock signal (external or internal), the truth is that they are an uncontrolled phenomenon. So, for an error injected into a combinational logic to be latched into a register, the error needs to be propagated throughout the combinational logic. The propagation time depends on where the error has been injected. If it has been injected at the end of the data generation path, for this error to be registered, the glitch needs to happen close to the clock's positive edge. Design [28] could detect these glitches. However, if the error is injected at the very beginning of the data generation path and propagated through to its output, then, for this error to be registered, the glitch should be applied earlier on. Design [28] would miss this glitch all together. This issue could be minimised by using a faster clock frequency for the detector, but it would increase the area and complexity of the device and, hence, its cost.

The detector in patent [29] is more suited for the asynchronous nature of glitches, as it detects glitches by continuously comparing the response to glitches of three different resistor-capacitor (RC) filters. With this detector, the glitches would be detected regardless of their likeliness to inject faults. By its components, this design is capable of detecting glitches at both sides of the voltage regulator (Vcc or Vdd) as well as glitches in the ground rail. This detector is capable of detecting positive and negative glitches in both power rails.

Two drawbacks can be identified with this design too. Firstly, due to the RC circuits' design, they will drain current continuously. This continuous current drainage is particularly undesired for battery powered applications due to the limited current budget. The drainage can be reduced by increasing the resistors’ values; however, doing so would impact negatively on the area and slow down the capacitor's charge speed, potentially missing some fast glitches. Secondly, is its dependency on the correct instantiation of resistors and capacitors, as process variations can result in a mismatch of the real value and, hence, not behaving as expected. Despite the drawbacks, this detector seems to be relatively fast at detecting glitches too, although probably not as fast as detector [28].

3.2.2 Design

The glitch detector proposed in this research is a mono-stable circuit which is always set to a given state. This state will only change as a result of a fault injection, in this case by a glitch attack. The mono-stable circuit designed in this work is based on the comparison between a reference voltage and a function of this reference voltage, as it is shown in the block diagram in Figure 3-6. The function block is characterised as a non-linear transfer function, where the transient response to a pulse differs from the reference voltage's transient.

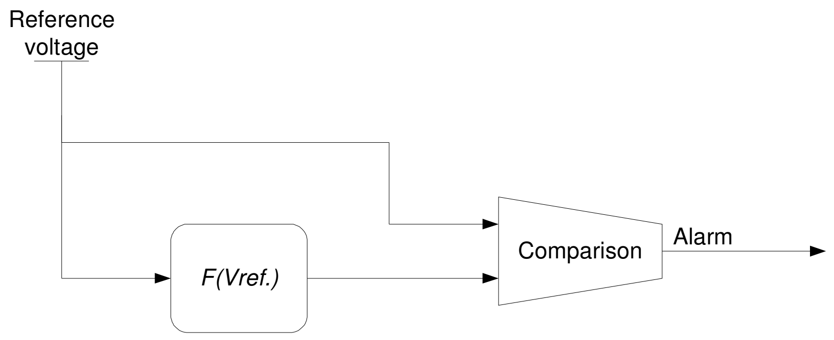


Figure 3-6 Diagram block of the glitch detector

The actual design of the glitch detector developed in this work is shown in Figure 3-7. Three parts compose it: a) a modified inverter, which is the non-linear transfer function; b) an operational amplifier (OpAmp) setup as a comparator; and c) an RS latch to register any detected glitch. This detector was designed to detect glitches at the output of the voltage regulator, hence, the reference voltage is the voltage regulator's output. Despite being designed to detect glitches at the voltage regulator's output, and as it will be covered in the discussion section, this design could also be used to detect glitches at the voltage regulator's input.

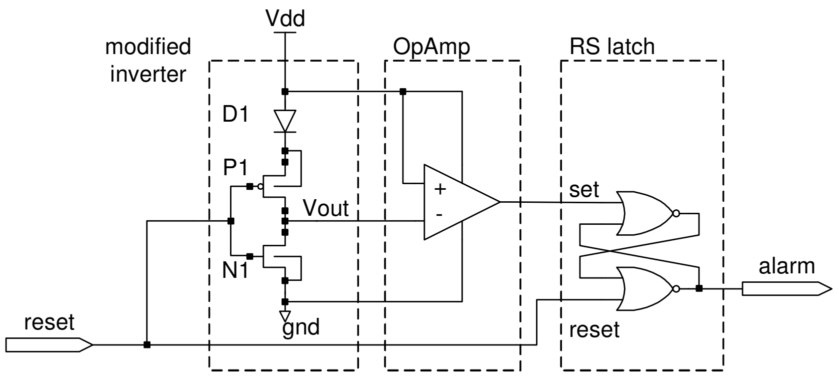


Figure 3-7 Proposed glitch detector

In normal supply conditions and when the circuit is disabled (the *reset* input is logic one), Vout is a logic zero. It results in the OpAmp's output being set to a logic zero and resetting the RS latch to a logic zero. When the circuit is enabled (the *reset* input is a logic zero), the modified inverter sub-circuit provides a constant voltage at its output, which is lower than Vdd due to the voltage drop in the diode D1. As Vout is lower than Vdd, the OpAmp's output is a logic zero. As a result, the value stored in the RS latch is unchanged.

When the detector is enabled, the inverter circuit can be substituted by its equivalent circuit in Figure 3-8. Here the inverter's transistor P1, which is conducting, is substituted by its equivalent R_{onP} (in the order of ohms). The inverter's transistor N1, which is not conducting, is substituted by its equivalent R_{offN} (in the order of mega ohms) in parallel with C_N, the parasitic capacitance between its source and drain. In addition to the inverter sub-circuit, Figure 3-8 also shows the equivalent load seen by the inverter's output, the OpAmp's input, represented by the $Z_{in.a.o}$ impedance and which is in the order of mega ohms.

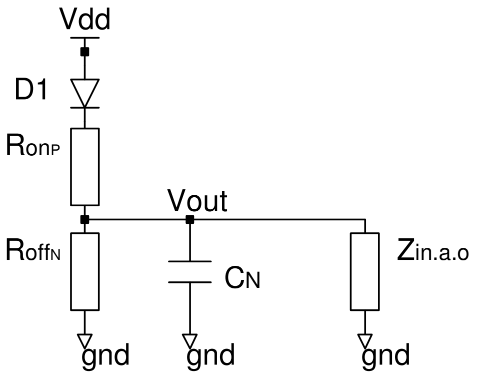


Figure 3-8 The inverter's equivalent circuit and load

The proposed glitch detector's response to a glitch attack when the detector is enabled is shown in Figure 3-9 and can be described as follows. When Vdd is stable, Vout is stable too and set to Vdd minus the voltage drop in the diode D1, which is around the diode's threshold voltage (V_t). If, as a result of a glitch, Vdd rises, the parasitic capacitor C_N is charged further, raising the voltage level at Vout. When Vdd falls, C_N starts to leak its charge through R_{offN} and $Z_{in.a.o}$, hence, lowering Vout. If Vdd falls at a higher rate than C_N is discharged, the diode D1 will, eventually, become inversely polarized. As a result, Vout will become higher than Vdd and stay higher even after the glitch's effects on Vdd have disappeared. This phenomenon can be detected by the comparator, which sets its output to a logic one, setting the RS latch to a logic one and triggering the glitch detection alarm. Upon a glitch detection, the CPU can run the appropriate routine or procedure related to glitches on the supply power. A Smart Card's typical response to a glitch detection is resetting the whole device.

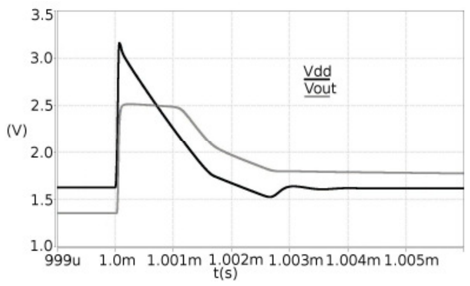


Figure 3-9 The modified inverter's response to a glitch

After detecting a glitch, the glitch detector would have to be reset in order to detect more glitches. This is achieved by setting the *reset* input to a logic one and back to logic zero. This process will discharge the capacitor C_N and reset the RS latch. If no action is taken after detecting a glitch, C_N will gradually discharge due to the leakage current, gradually reducing Vout to its previous level, i.e. below Vdd. However, the RS latch would continuously indicate the detection of a glitch, and no further glitch would be detected.

The diode D1 plays an important role in this design, as without it, the parasitic capacitor Cn could also be discharged through the power supply. This additional leakage source would result in faster discharge rates and, hence, reduce the chances of glitch detections, as it would be harder for Vout to become higher than Vdd and, when becoming higher, it would stay that way for shorter periods of time. Ultimately, this design would require a faster comparator.

This glitch detector is patented under the International Publication Number WO 2008/033712 A2. This patent is included in the Volume II of this portfolio, GlitchPub3.

3.2.3 Test and Results

A series of simulations and silicon tests were conducted, to determine the performance of this design and whether it is capable of detecting a series of fast glitches that currently go undetected by the glitch detectors used by Atmel. Two different versions of the proposed design were simulated and four versions of this design were instantiated and tested in the test-chip 01VGA. Following subsections cover the simulation and silicon test and results.

3.2.3.1 Simulation Test and Results

In order to test this glitch detector's performance, a simulation environment was setup, which emulated the conditions in which this detector would operate. The setup in particular, shown in Figure 3-10, consisted of a voltage regulator powering the glitch detectors under test and a variable resistive load and a fixed capacitive load. The voltage regulator used in this setup was designed for low security applications such as global system for mobile communications (GSM). The variable load, connected to Vdd, was designed to emulate the typical loads the voltage regulator can be subject to during normal Smart Card operation. The load values were: high load (80 Ohms); medium load (1K6 Ohms); and low load (3K2 Ohms). The capacitive load was of 2.2nF. The glitch detectors were connected to the voltage regulator's output, Vdd.

Two versions of the proposed glitch detector were designed and simulated; they differed in the transistor type they were built with. The first glitch detector, Design A, had its diode and RS latch made of low leakage transistors and the inverter and OpAmp made of high voltage transistors. The second glitch detector, Design B, was made out entirely of low leakage transistors. The section 3 of the GlitchTech1 covers the transistor sizes of the modified inverter of both designs.

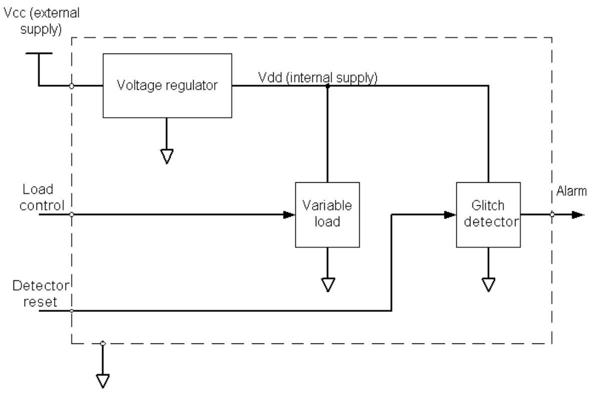


Figure 3-10 Diagram of the glitch detector test setup

Low leakage transistors are designed with a higher threshold voltage (V_t) than their normal counterparts. Increasing the V_t helps reducing a transistor's leakage current when it is in the cut off region, an increasing issue with deep-submicron technologies. However, it also reduces the transistor performance when comparing to transistors with lower V_t. The diode was designed with a low leakage transistor in both designs to minimise the leakage through the diode and, hence, to isolate Vout as much as possible from Vdd. High voltage transistors should allow the OpAmp to operate within its specified voltage range even when the effects of a glitch is present.

The above setup was simulated with HSPICE where the designs were tested for the following power supply scenarios:

- clean external Vcc supply, 3v and 5v;
- noisy external Vcc supply, 3v and 5v with a +/-10% ripple noise at 100Hz (the standard for this application states that +/-10% ripple at 50Hz is a valid supply source)[2];
- two fast positive glitches; one that went from 2.7v to 7v and back to 2.7v in 100ns (Glitch A), and the other that went from 3v to 15v and back to 3v in 10ns (Glitch B)

Each of these power supply scenarios were repeated with three different load values (i.e. high, medium and low) and two different SPICE models of transistors. Only extreme SPICE models were simulated, i.e. worst-case and best-case. Worst-case model had the following

[2] Knowledge gained at Atmel

parameters: `mos_wcs, rlow, clow, temp=125`. Best-case model had the following paramenters: `mos_bcs`, `rhigh`, `chigh`, `temp=-40`.

In total, 12 different noisy scenarios and 12 different glitch scenarios were simulated for each detector design version. For the clean power supply cases, only medium load was used, as repeating the test with different load values would not provide further valuable data. Hence, only 4 different clean power scenarios were tested per design.

None of the simulations with a clean source supply, or the simulations with a noisy supply source triggered the glitch detector. Out of the 12 glitch scenarios each design was subjected to, Design A managed to detect the glitch in 11 scenarios with an average detection time of 1,073ns. Design B managed to detect the glitch in 9 scenarios with an average detection time of 7,657ns. The time results of both designs for each glitch scenario simulations are collected in Table 3-2. The report GlitchPub1 shows more simulations results.

Table 3-2 Glitch simulation results

Design version	**Load**	**SPICE model**	**Applied glitch**	
			Glitch A	**Glitch B**
A	high load	worst-case	not detected	741ns
		best-case	571ns	611ns
	mid load	worst-case	1,375ns	1,281ns
		best-case	1,015ns	990ns
	low load	worst-case	1,508ns	1,394ns
		best-case	1,252ns	1,062ns
B	high load	worst-case	629ns	8,273ns
		best-case	not detected	not detected
	mid load	worst-case	2,093ns	1,947ns
		best-case	1,359ns	25,522ns
	low load	worst-case	2,163ns	2,170ns
		best-case	not detected	24,755ns

3.2.3.2 Simulation Discussion

When comparing glitch detectors, the comparison needs to focus on two axis, detection range and detection time. In general terms, these simulation results demonstrate that this glitch detector is capable of detecting glitches that detectors used with the instantiated voltage regulator are not capable of. The detection time ranges from 571ns to 25,522ns, depending on the simulation scenario. Not so immediate in computing terms as, in the worst case scenario, an AVR CPU running at 50MHz could execute up to 50 instructions per microsecond.

Comparing the simulated designs, Design A excels over Design B in terms of amount of successful detections and detection times. This is primarily due to the different composition of each design. Low V_t transistors have a lower performance, and that is reflected in Design B's amount of detections and detection times. Furthermore, during a glitch, low V_t transistors might be subject to voltage levels outside their specifications, and this can temporally affect the circuit's ability to detect such glitch. When it comes to high voltage devices, however, the voltage levels they are subject to during the glitch are likely to be within or not that far of their operational specifications, hence, having a lower impact.

The transistor choice can explain one design taking longer than the other to detect or detecting more or less glitches. However, the detection time divergences between Glitch A and Glitch B needs further explanation. Analysing the evolution of Vdd and Vout in both designs during and after a glitch, it can be noticed that the voltage difference between Vout and Vdd is lower in the case of the Design B. This difference is particularly small when applying the Glitch B, Figure 3-11 and Figure 3-12. The smaller the voltage difference between Vout and Vdd, the longer the OpAmp takes to trigger the alarm and the more chances of missing that particular glitch. This indicates that Glitch B sits in the detection range limit of Design B. Furthermore, it seems that the closer a glitch is from a detector's detection range limit, the longer it takes to be detected.

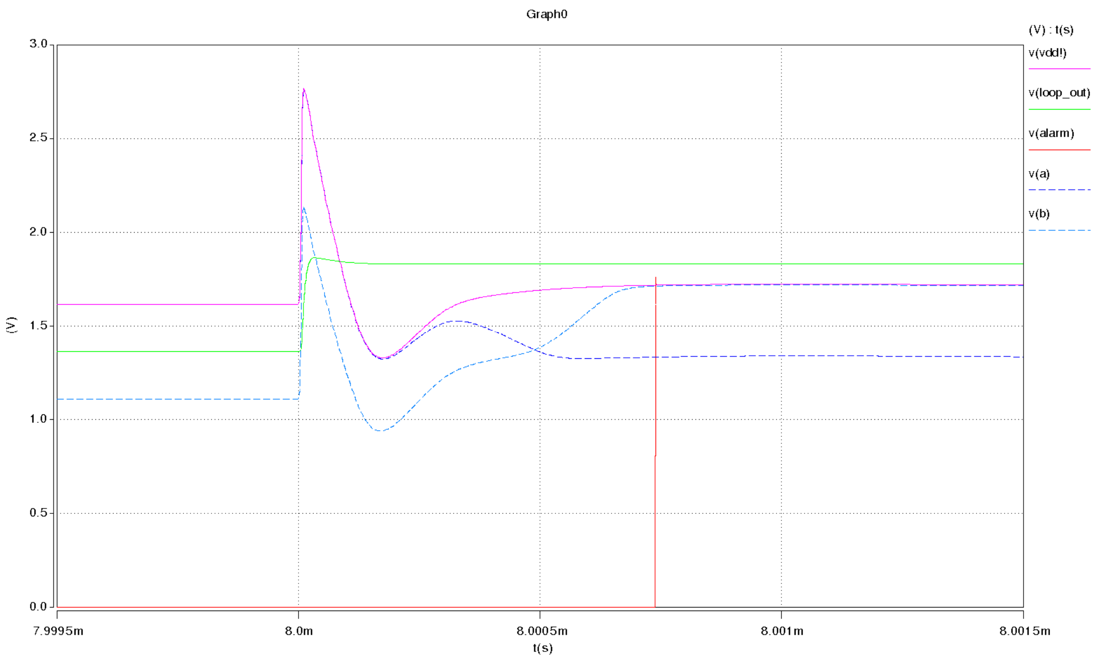


Figure 3-11 Design A's response to Glitch B with high load and worst-case SPICE models

Design A, again, performs better than Design B in these simulations. It is also capable of detecting glitches that detectors used with the instantiated voltage regulator cannot. However, its average detection time, 1,073ns, is longer than the direct approach detectors' typical detection time, in the order of tens of nanoseconds. This is, Design A takes one to two orders of magnitude longer to trigger the alarm signal. For obvious reasons, it is better to detect a glitch than not detecting it at all, but in terms of detection time, what is the detection time boundary after which, the detection can be considered too late? This is, when the fault injection can be considered irreversible and raising the alarm would not provide any security benefit.

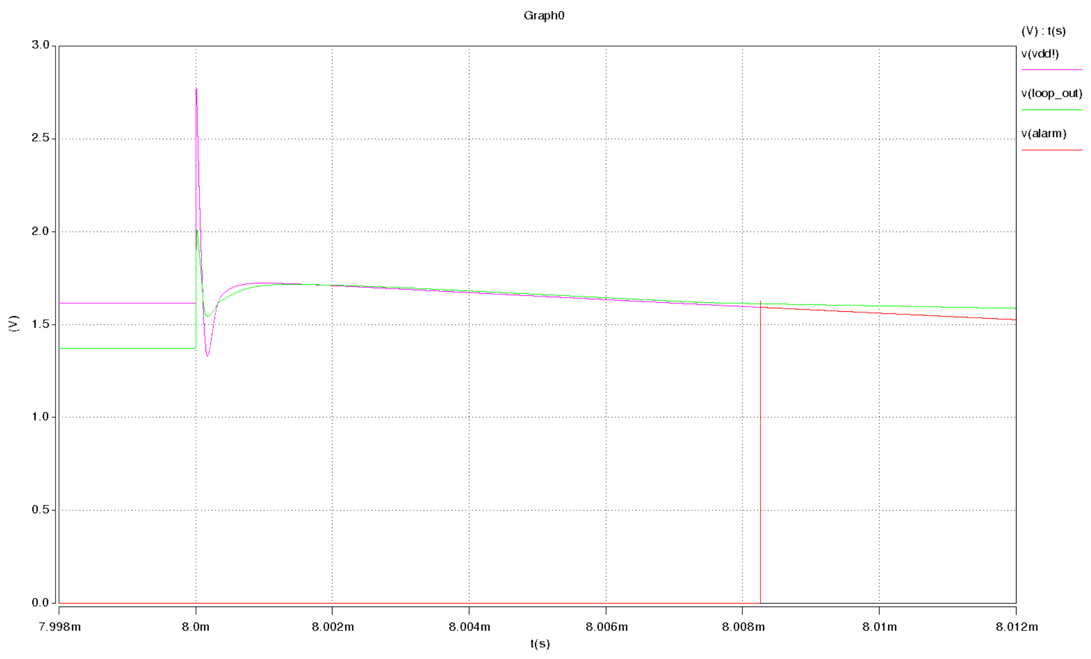


Figure 3-12 Design B's response to Glitch B with high load and worst-case SPICE models

Looking at a Smart Card from a high level perspective, the following two aspects can be identified. Firstly, a Smart Card is a device that carries out transactions, where it first receives a command, it executes that command, and then transmits the result back. Secondly, a Smart Card has both, volatile memory (i.e. SRAM) and non-volatile memory (e.g. EEPROM), where volatile memory is used to store temporal data and the NVM is used to store Smart Card software applications and application and user data.

Command transfer and result transfer are made in a serial mode at a maximum clock speed of 5MHz, according to the ISO standard. In this context, a transaction will require in the order of milliseconds to seconds to be completed. If a glitch is applied aiming to disturb a transaction, with an average detection of just over 1 microseconds, the glitch could be detected before the transactions has been completed and act in consequence, such as invalidating the transaction by producing an erroneous response and or resetting the Smart Card.

It be could argued that the execution of transaction commands is not atomic, as it takes several CPU instructions to execute the transaction command. This is correct and, in this context, a glitch attack might result in diverting the program flow and/or corrupting data stored in the registers and/or SRAMs, which takes us to the second aspect mentioned above, the memory. An attack that corrupts the SRAMs and registers is an attack that the device can recover itself from by resetting itself, as it would result in an initialisation of certain registers and the SRAM memory to a known status. If an attack resulted in corrupting the NVM, however, the matter would be different, as it would be virtually impossible for the Smart Card to know which data has been affected and what its previous value was. However, with NVM writing times in the order of 2 milliseconds, with the proposed detectors, the Smart Card could still reset itself in time to stop the write operation and avoid corrupting the NVM.

Another interesting feature that can be noticed in Table 3-2 is the load's impact on the detection time. Overall, there is a trend of longer detection times for lower loads, which is more noticeable with Design A. To understand this feature, let us focus on the evolution of Vdd and Vout when the voltage regulator is under a glitch attack, Figure 3-13. The simulations have shown that the time required to detect a glitch can be divided in two parts: a) *t1*, the time needed for Vout to become higher than Vdd; and b) *t2*, the time between Vout becoming higher than Vdd and the raise of the alarm signal. When subjecting the voltage regulator to different loads, the time *t1* suffered significant variations, whereas time *t2* did not. Higher loads resulted in shorter *t1* times than lower loads. In synthesis, a voltage regulator subjected to higher loads, outputs a higher current. This means that when subjected to a glitch attack, its output tends to correct itself faster (Vdd raises and falls faster), hence, Vout becoming higher than Vdd earlier on and allowing the glitch detection sooner than with low loads.

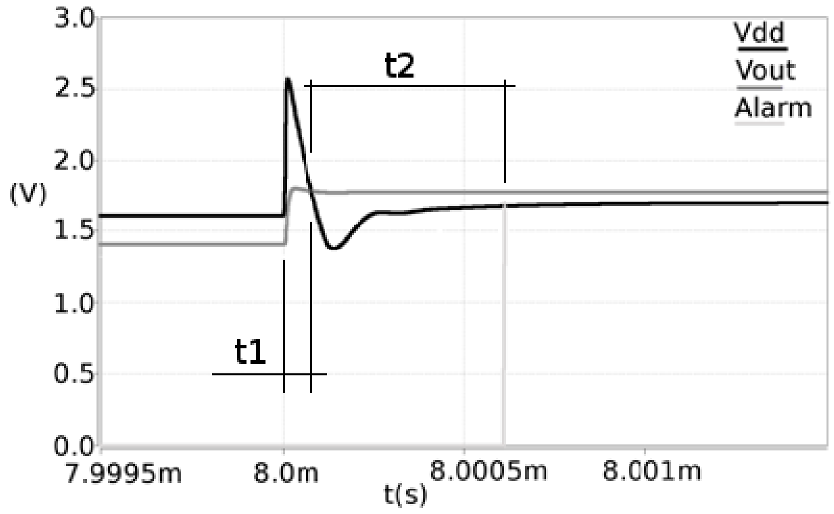


Figure 3-13 Simulation of glitch attack detection under a high load

Looking at the bigger picture, the detector proposed in this research is capable of detecting glitches that current detectors cannot. However, it could only be used as a complement to current detectors due to its prolonged detection time. This way, in the range where both detectors can detect glitches, current detectors would trigger the alarm immediately. Whereas the proposed design would be left in charge of detecting glitches that current detector cannot.

3.2.3.3 Test-chip Test and Results

Two instances similar to the glitch detector simulation setup were instantiated into a test-chip to characterise the proposed detector's detection range and time for different voltage regulator and load scenarios. Each instance included a different voltage regulator, capacitive and resistive loads and four different versions of the proposed glitch detector. The obtained characterisation data was compared against the detection range and time of the glitch detectors embedded in the instantiated voltage regulators. The aim of this comparison is to determine the detection range improvement provided by the proposed detector. A block diagram of the test-chip is shown in Figure 3-14, where GD_0, GD_1, GD_2 and GD_3 represent the different glitch detector versions and VR1 and VR2 represent the different voltage regulators.

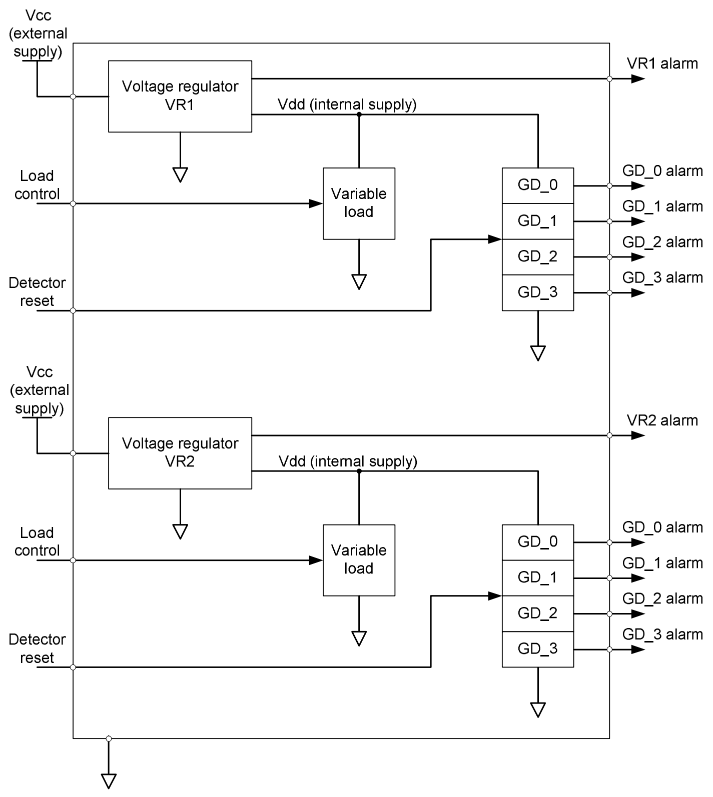


Figure 3-14 Block diagram of the test-chip

The instantiated versions differ in the following: GD_0 was designed using low leakage transistors for the diode D1 and the RS latch and high voltage transistors for the inverter and the OpAmp, same as the simulated Design A; GD_1 was designed using low leakage transistors throughout, same as the simulated Design B; GD_2 was designed using high voltage transistors throughout; and GD_3 was designed using high voltage transistors throughout too but with a different OpAmp design.

The voltage regulators instantiated in this test-chip differ in their target product and, hence, their performance. VR1 is the same voltage regulator as the one used in the previous simulation setup, which was designed for products with low security requirements. VR2, in the other hand, was designed for high security products. These voltage regulators were instantiated with their corresponding glitch detectors, so that they could be characterised too and compared against the proposed detector. Since the simulation results already determined that the proposed detector could only be used in conjunction with existing ones, and the aim is to determine the detection range improvement provided by the proposed detector, only the voltage regulator's Vcc, Vdd and GND glitch detector's combined detection signal was monitored. A side effect of this comparison approach is not being able to establish a direct comparison between the proposed designs against those built-in ones monitoring Vdd.

These voltage regulators have different responses to a glitch attack, as shown in Figure 3-15 and Figure 3-16, where the amber waveform shows the glitch applied to Vcc and the pink waveform shows the impact on the voltage regulators' output, Vdd. As in the simulation setup, each voltage regulator also powered a fixed capacitive load of 2.2 nF and an independently controllable variable resistive load, which could be set to: high load (80 Ohms); medium load (1K6 Ohms); and low load (3K2 Ohms).

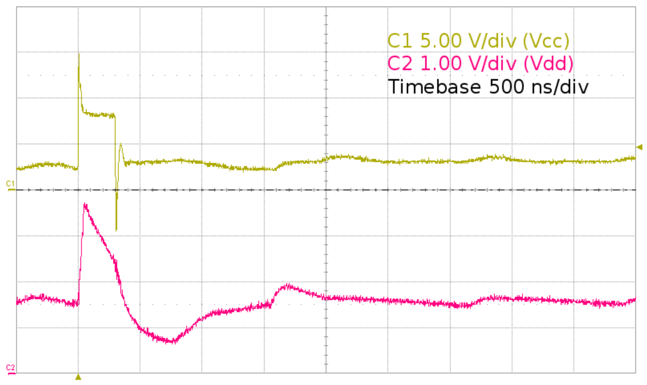


Figure 3-15 VR1's response to a 17V and 300ns glitch under high load, Vcc = 3V

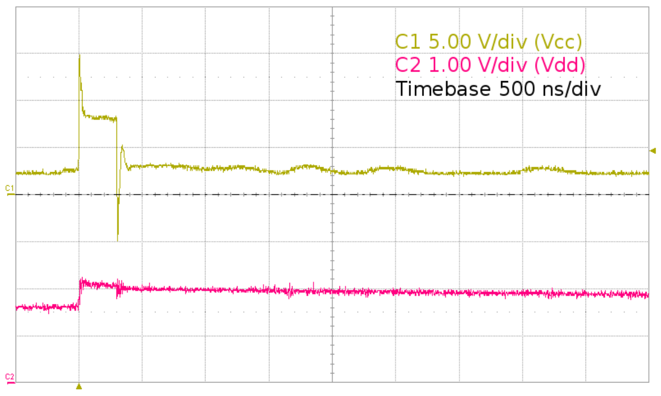


Figure 3-16 VR2's response to a 17V and 300ns glitch under high load, Vcc = 3V

Each glitch detector version was characterized for detection range and detection delay using both voltage regulators, four resistive load combinations (high, medium, low and variable) and two base voltage levels, 3V and 5V. Furthermore, these tests were repeated in four different test-chips. Figure 3-17 shows a block diagram of the test environment. The characterisation was run by applying positive glitches of different amplitudes and widths on Vcc with the HP81110A pulse generator, which higher pulse is limited 20V. In total, 416 different glitches were applied for the 3V base supply, and 364 glitches for the 5V base supply. The test environment and test board are covered in detail in the section 4 of the report GlitchTech1.

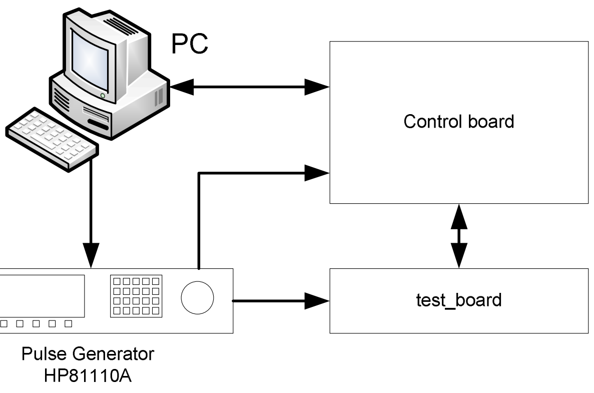


Figure 3-17 Diagram of the test environment

Figure 3-18 shows the detection characterisation of the proposed glitch detectors when powered with the VR1 for high and low loads, as well as the combined characterisation of the glitch detectors embedded into the VR1 and how it compares to GD_3. All these figures focus on glitch widths between 10ns and 150ns as this is where the most useful information lies. The proposed detectors monitoring the VR2's output hardly detected any glitch at all. Hence no valid characterization or comparison could be done for the VR2 case.

Finally, Table 3-3 presents the maximum, minimum and average detection time for each detector when VR1 is subject to high and low loads. The longest detection times are associated to glitches in the detectors' detection limit, whereas short detection times are associated to glitches far from the detectors' detection limit.

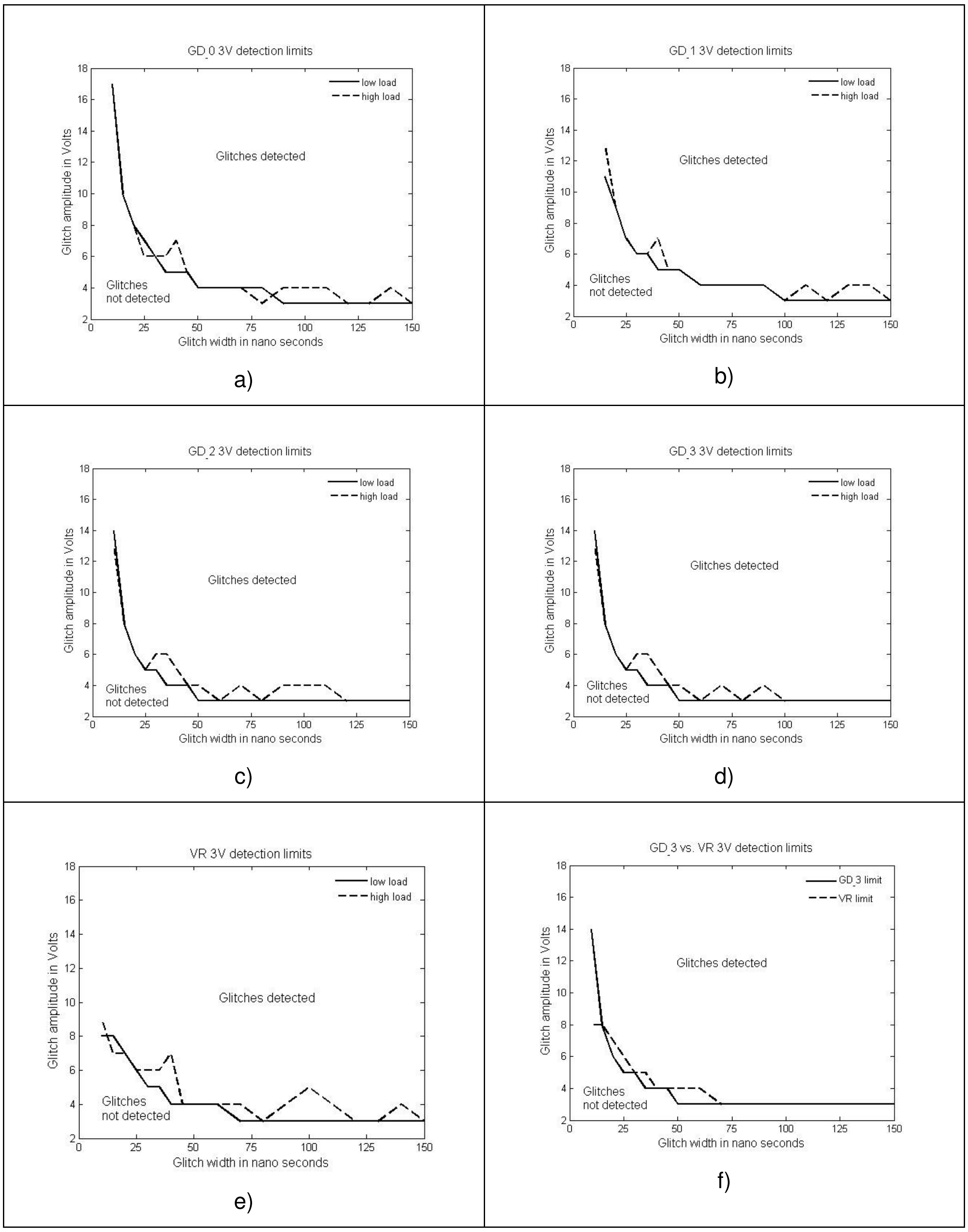


Figure 3-18 Glitch detection range for different detectors when Vcc = 3V: a) GD_0 3V; b) GD_1 3V; c) GD_2 3V; d) GD_3 3V; e) VR1 3V; and f) GD_3 vs VR1 3V

Table 3-3 Overall detection times in nanoseconds

Detector	Load	Detection time		
		Min	average	max
GD_0	High	870	1200	1640
	Low	910	1310	2460
GD_1	High	1100	1400	2005
	Low	1250	1600	2020
GD_2	High	310	420	1250
	Low	320	470	1380
GD_3	High	310	440	1300
	Low	310	460	1370
VR1	High	40	55	400
	Low	50	60	440

The technical report GlitchPub2 also covers the characterisation work of the glitch detectors and the results.

3.2.3.4 Test-chip Discussion

Several deductions can be made from these results. Firstly, these results corroborate the simulation results. The best detectors' maximum detection time is around 1,300ns, and in general it takes between 3.10 and 8.5 times longer to trigger the alarm than the current detectors. However, this is well within the typical Smart Card transaction time as well as the typical NVM write time. Hence, the statement made in the simulation analysis still holds valid: the proposed detector could be used only in conjunction with current detectors due to their slower detection time.

Also, it is clear that the load and voltage regulators play a key part on the detectors' detection range and time, although the load impact is not as severe as initially indicated from the simulation results. The voltage regulator, in the other hand, had a massive impact on the detector's performance, as detectors coupled with the VR2 hardly detected any glitch. The explanation can be found in the voltage regulator's response to a glitch.

The VR1's output behaviour when a glitch is present in Vcc can be very dramatic. In Figure 3-15, the voltage regulator's output raises almost immediately from 1.8V to about 3.5V and then it drops it back very quickly too. On the other hand, the VR2's output behaviour is much more controlled. In Figure 3-16, the VR2's output raises to just over 2V but then it drops it gradually back to 1.8V. This VR2's behaviour does not allow the modified inverter's output, Vout, to become higher than the nominal Vdd, 1.8V. Without this first step, the glitch detector is not capable of detecting a glitch. Furthermore, even if a glitch could raise Vout over 1.8V, Vdd's gradual drop is too slow to allow it becoming lower than Vout. Hence, dissipating at once any chance of detecting a glitch on Vdd with this detector when is paired with the VR2.

Focusing on the VR1 scenario's results, detectors GD_0 and GD_1 performed worse than GD_2 and GD_3 when it comes to detection range. Their timing performance was the poorest too. This is directly related with the detectors' design decisions. GD_0 and GD_1 used low leakage transistors, which are characterized by their low speed.

Detectors GD_2 and GD_3 showed a similar behaviour both in terms of detection range and time, where GD_3 has a slightly better detection range, whereas, at times, GD_2 is a slightly faster detector. Since in this work, expanding the detection range is a priority, detector GD_3 is considered marginally better than detector GD_2. The detection range improvement provided by the GD_3 over the VR1's combined detection range in shown in Figure 3-18.f), where the dashed line represents the VR1's combined detection range and the continuous line is GD_3's detection range. This improvement is repeated across most of the tests carried out in this work.

Looking at the effect the load has in the detection range and time, it can be noticed that when VR1 is subject to lower loads, all detection ranges are expanded at the cost of speed. This same phenomenon can be seen too in the VR1's built-in detectors. Section 3.2.3.2 Simulation Discussion explains why a higher load results in faster detection times, and why it takes longer to detect a glitch when it sits in a detector's detection range limit. The phenomenon highlighted here works as a combination of the two previous ones.

When it comes to detecting glitches, at the detection range limit, the evolution of Vdd becomes crucial. Vdd not raising high enough or falling before Vout has risen above the nominal Vdd (1.8V) could easily mean missing the glitch, and this is what actually happens. As previously explained, Vdd shows a faster recovery when subjected to higher loads and, at the detection range limit, this is translated by not letting Vout becoming higher than Vdd. Low load, on the other hand, allows more severe changes on Vdd after a glitch, which, at the detection range limit is translated as still detecting the glitch.

Finally, characterising the combined detection range and time of the detectors built into VR1 and VR2, makes it impossible to know each individual detector's performance both, at the voltage regulators' input and output. Whilst the individual comparison was not the objective of this work, having that information could have helped with discerning each detector's contribution to the combined detection range and time. Furthermore, it could have helped on determining the performance improvement introduced by the proposed detector when comparing it to glitch detectors at the VR1's and VR2's output.

3.2.4 Conclusion of the glitch detector

Detectors currently used in Smart Cards primarily focus on detecting glitches in real time by directly monitoring the power rails. Detecting fast glitches with this approach is challenging, as legitimate noise could be mistaken for a glitch, triggering false detections. Instead, the glitch detector proposed in this work, and for which a patent has been granted (see GlitchPub3 in Volume 2), allows the safe detection of those fast glitches by monitoring the glitches' effect on a sensitive circuit.

Test results have demonstrated the usability of the proposed detector and how it improves the overall detection range and, hence, the security. However, and due to the high detection delay comparing with direct approach detectors, the currently proposed detector could only be used in conjunction with current detectors. This is, as a supplement to the other detectors.

There has been a limitation on the comparison between the proposed glitch detector and the current ones. This limitation was produced by the combined monitoring of the built-in detectors' behaviour, which did not allow us to make any further analysis on how the instantiated versions compare against the individual built-in detectors, especially the internal ones.

The instantiation of different versions of the proposed glitch detector has proved that small changes in the design can result in faster glitch detection and bigger detection range. There might still be room for improvement. By making the comparator even faster and more sensitive, the detection range could be expanded and the detection time reduced, whilst still avoiding detecting noise as a glitch.

Furthermore, it has been corroborated that the same glitch might affect a design differently depending on the environment parameters. The voltage regulator was the single factor with the highest impact on their detection capabilities, to the point where in one of the cases, no detections were registered. Other two factors affecting the detection capabilities included: a) the detector's design; and b) the voltage regulator load, although this last factor had a lower impact than the previous two.

Finally, throughout all this work, the glitch detector has been designed to detect glitches at the voltage regulator's output. However, the design principle could still apply if designed to detect glitches applied in Vcc, and it would be interesting to see how it performs at detection range and time, especially for the case of the VR2.

3.2.5 Future work

As with many other detectors and analog circuits, this detector might need tweaking for each new technology node. Hence, the design will need to be constantly reviewed. In addition to this iterative improvement, three lines of action could be suggested: a) adapting the glitch detector to detect glitches in Vcc; b) characterise the voltage regulators' internal and external detectors separately and compare them to the Vcc and Vdd glitch detector versions; and c) testing the current detector with negative glitches on the ground and redesigning it to detect negative glitches in Vcc.

3.3 Laser Attacks

Silicon devices' susceptibility to lasers is well known by the aerospace community, since they used a laser beam for the first time in 1965 [7] to simulate the effects of ionising radiation on semiconductors. Nowadays the laser technology is more affordable, making laser attacks more likely. In fact, in [8], a successful attack on a SRAM was achieved with a camera flash, which shows the potential threat posed by this attack technique.

Unlike consumer devices, aerospace devices are developed to withstand these kind of threats. This research line analysed the applicability of a few radiation hardening countermeasures to Smart Card devices and developed the understanding of laser attack effects. The following section reviews the state of art of different laser attacks techniques and radiation countermeasures. The next section covers the instantiated countermeasures and the tests carried out. This is followed by the conclusions and future work.

The following sections cover the literature review; the countermeasures tested in this work; and the conclusions of this research line. Finally, some possible future developments are proposed.

Volume II includes the following reports related to and supporting this research line:

- LaserTech1(Tartalo test-chip 01OKA)

3.3.1 Literature Review

The impact of energy particles (radiation) on microelectronic circuits produces a phenomena commonly referred to as single-event effect (SEE), which can cause temporal or permanent changes on microelectronic devices [30, 31]. A full list of SEEs can be found in Appendix C. Lasers have a similar behaviour but with some key differences: a) test repeatability; b) needs direct access to silicon; and c) only induces a subset of the radiation SEEs. The actual differences between radiation and laser hits are collected in Table 3-4.

A particle's hit size and energy are determined by the particle itself; as different particle types might have different sizes or energy levels. A particle hit is also random in the temporal and spatial domains. Even if particle accelerator chambers used for radiation tests, such as [30], allow defining a variable target area and radiation level, the actual hit cannot be controlled.

Table 3-4 Radiation vs. laser effects on silicon

Feature/Property	Radiation	Laser
Hit location	random	controlled
Hit time	random	controlled
Hit duration	in the order of pico seconds	controlled
Hit size	particle's size	laser spot size (controlled)
Hit energy	particle's energy	controlled
Barriers	Penetrates the whole chip (package, metal layer and die)	Needs direct access to the silicon. Package and metal layers act as barriers
Induced upsets (SEE)	SEU, SEL, SEB, SEGR, SETD, SET	SEU, SEL, SET, SETD

Laser beams, on the other hand, are far more controllable, as the beam can be directed to a specific area and the hit time controlled and even synchronised with a particular event in the device under test (DUT). The laser beam's spot size and energy level can also be controlled, and enables targeting anything from an individual transistor to the whole device.

Although in theory a laser beam could be on for an undetermined amount of time, in practice, short pulses are used. Applying a laser beam for a long time might render the targeted area unusable for as long as the laser is present, and depending on the laser energy level it could even damage it permanently. After all the aerospace industry's aim is to emulate the radiation's behaviour, and an attacker aims to inject some sort of temporal fault that produces valuable behavioural information. A short pulse is enough for that.

Controllability of laser beam parameters allows test repeatability, making laser beams desirable for the aerospace industry to tests electronic devices' response to a space like environment. Controllability also allows carrying a series of attacks not possible with radiation, such as synchronising the laser beam to the DUT's system clock. Another attack option could be using big laser spots to comb the device and identify the location of IPs. Then, a small laser spot could be used to target specific parts of the IP, such as registers. These test scenarios makes laser attacks desirable to an attacker.

One of the laser beam limitations is barriers. The laser needs to target the silicon in order to induce a SEE, and packaging and metal layers act as barriers to laser beams. Furthermore, with the increasing design complexity, the silicon gets buried under several metal layers and successful front-side attacks are harder to achieve [32]. This can be worked around, however, with back-side attacks [33], although substrate milling is often required first to improve the laser penetration.

Comparing to radiation, laser beams can only induce a subset of SEEs, of which SEU and single event transient (SET) are the ones an attacker would aim for. SEUs cause a change of the data stored in registers and memory cells. SETs cause short duration effects that could result in error injection. These errors are temporary in nature.

A lot of effort has been put towards hardening devices against radiation. Radiation countermeasures include those proposed in [34-40], where the use of radiation hard processes (RHP) has been the most commonly used countermeasure. Devices built on RHPs are inherently more robust than commercial ones, however, the use of RHPs is in decline, as they are more expensive and power hungry than commercial CMOS processes and usually lag two generations behind commercial ones [41].

Foundry level countermeasures are out of the scope of this research. However, RHP would be hard to justify in a commercial environment such as that of Smart Cards, were size, power consumption and price are major constraints. More adequate than RHP are the countermeasures based on radiation hardening by design (RHBD), which rely on specific circuit design techniques to increase the radiation resistance of a device [30, 34]. RHBD techniques include: a) redundancy; b) memory immunity [35, 38]; and c) transistor layout [37].

Memory immunity RHBD focuses on sequential logic and storage, whereas redundancy and transistor layout are applicable to both, sequential and combinational. Hardening data storage cells is important to avoid corrupting the information they hold, but with soft errors in combinational logic increasing exponentially with the clock frequency [39, 41], hardening combinational logic is also becoming a need.

There are two main kinds of redundancies [3]: time redundancy and hardware redundancy. Time redundancy is based at computing the same input data at different times and comparing the results. This countermeasure could potentially help detecting laser pulses that last less than the whole cycle of processing and comparing the data. Although [3, 36, 39, 40] cover the hardware instantiation of this countermeasure, it could actually also be instantiated in software for some functions such as cryptographic operations.

One intrinsic feature of this technique is, however, the time delay penalties it incurs on, as the same data needs to be processed at least twice within the same cycle. The imposed delay might be prohibitive on applications or ICs which are already struggling to meet timing specifications or where applying this technique implies reducing the device’s maximum performance.

Hardware redundancy countermeasure is based on instantiating the same block several times and comparing the outputs of these blocks [3]. The time penalty associated to these techniques is not that severe, as all blocks will process the data at the same time; the voting circuit will be the only additional time delay to the design. The main drawback of this approach is, however, the area penalty it incurs on, as the same block needs to be instantiated at least twice for a fault to be detected, and at least threes times for a fault to be corrected.

By using hardware redundancy, a circuitry's or blocks immunity against radiation or laser attacks could be improved greatly, as all redundant blocks would have to be attacked at the same time and inject the same error in order to work around this countermeasure. Redundancy blocks can be located apart from each other in order to improve the immunity further.

There is, though, a question on how well protected the voting circuit will be against laser attacks. The voting system improves a block's immunity to laser attacks, but unless the voting circuit's immunity is matched to the block's one, an attacker could potentially target the voting circuit to inject a fault, bypassing then the voting technique itself. And, if the voting circuit can be designed in such a way, other than redundancy, that hardens it against laser attacks, would it be possible to design the block following the same approach in first instance and avoiding the use of redundancy?

Given a radiation resistant voting circuit, hardware redundancy could be used in Smart Cards to enhance the immunity of certain critical blocks or functions, such as FSMs, the CPU's ALU or the crypto engine. The area and power consumption penalty resulting from the use of hardware redundancy would suggest against extensive use of this technique though.

Radiation hardened cells like dual interlocked storage cell (DICE) [38] and SERT SEU immune register [42] improve the cell's stability and robustness against radiation effects by duplicating the feedback paths. A particle hitting one transistor in a DICE structure most likely will not inject any upset, and the stored value will remain unchanged [38]. This hit, however, could result in two nodes or transistors driving a net or wire simultaneously and with different voltages for as long as the hit effect is present, which would increased power consumption. Also multiple simultaneous particle hits might result in fault injection in the memory cell.

Low power SERT SEU structure improves on the DICE's disadvantages. By design, a single hit could not result in two nodes driving a wire concurrently, and it can handle up to two simultaneous particle hits. The downside of SERT SEUs is their size, as the amount of transistors needed for a standard latch cell is 4, the transistors for a DICE latch cell is 8 and the number of transistors for a SERT SEU latch can be as high as 12.

In any case, it might only be suitable for specific registers or latches. Instantiating a SRAM with any of these two countermeasures would dramatically increase the area overhead, hence it should be avoided. Instead, error detection and correction circuitries or policies such as parity and Hamming code should be added to further increase the device's robustness against radiation or laser attacks.

Finally, radiation induces edge current leakage on affected transistors, which in turn can inject a fault into adjacent devices [30]. Enclosed layout transistors (ELT), also known as edge-less transistors, mitigate the radiation effects by designing transistors with no edges, as shown in Figure 3-19. The main drawback of these transistors is their size, as they can be three times bigger than standard transistor layout. In addition to the size, these transistors also have a higher parasitic capacitance, which makes them slower and more dynamic power hungry.

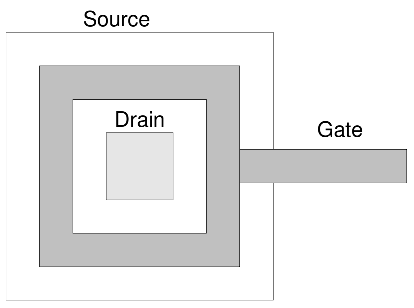


Figure 3-19 Enclosed layout transistor

3.3.2 This Work

A test-chip (01OKA) was designed with some of the countermeasures described above instantiated into simple 4-bit adders, as the one shown in Figure 3-20. These countermeasures included:

- **Layout alterations:** A laser attack success depends not only on the timing, i.e. when a specific transistor is illuminated, but also on the transistors that can be targeted individually or as a group, which might be affected by its layout. Three register layouts shown in Figure 3-21 were instantiated to determine how the layout can influence on the effects of a laser attack.
- **Enclosed layout transistor:** ELT transistors were designed and instantiated to test their performance against light attacks.
- **Low power radiation hardened register:** SERT registers were designed to test their response to laser attacks.
- **Radiation mitigation on combinational logic:** The radiation countermeasure presented in [36] was instantiated at the output of all adders, as shown in Figure 3-22.
- **Parity error detection:** This simple error detection technique was used to calculate the addition's parity and storing it in a register.

- **Covering the silicon with a metal layer:** This technique was used to completely cover a test circuitry with a metal layer. This metal plate was grounded to avoid it getting charged with static electricity.

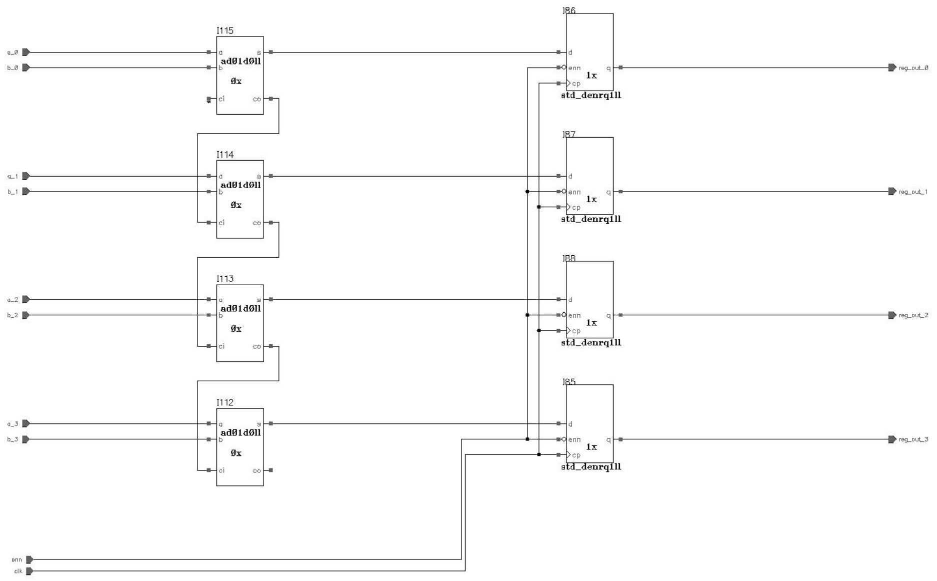


Figure 3-20 Basic light attack test circuit

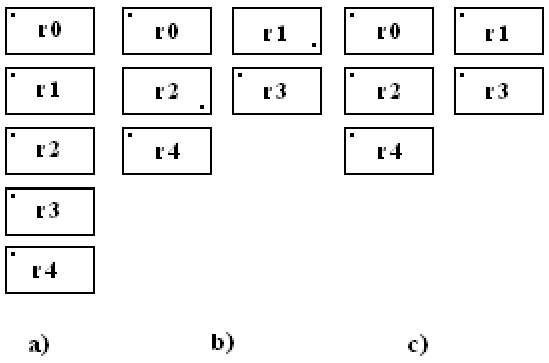


Figure 3-21 Different layouts used in the test-chip

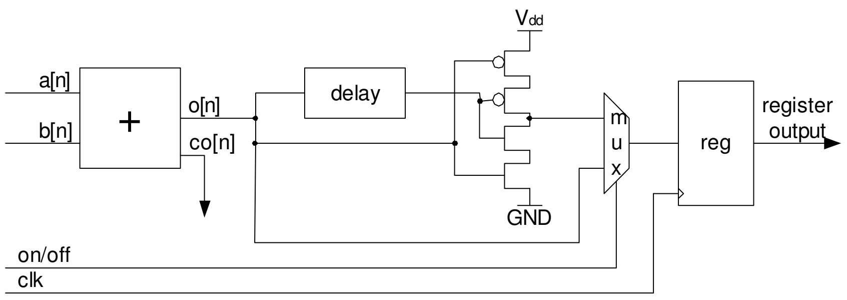


Figure 3-22 Implemented combinational block SEU mitigation logic

Six adders were designed in total, each of them targeting a particular set of countermeasures. The relationship between the adder design and the instantiated countermeasures is shown in Table 3-5. All designs except for ELT design were built using normal MOS transistors. The ELT design was fully built of ELT transistors, so their performance in the registers and the combinational blocks could be measured.

Table 3-5 Design and countermeasure relation table

Design name	Layout version			RHBD	ELT	Combinational mitigation	Parity	Metal layer
	a	b	c					
Layout_a design	x	-	-	-	-	x	x	-
Layout_b design	-	x	-	-	-	x	x	-
Layout_c design	-	-	x	-	-	x	x	-
RHBD design	-	-	-	X	-	x	x	-
ELT design	-	-	-	-	x	x	x	-
Metal design	x	-	-	-	-	x	x	x

3.3.3 Tests

The test consisted of targeting the adder circuitries with laser pulses when the registers stored a fixed value and whilst exercising the adder circuitries. The actual test methodology is covered in the report LaserTech1. The test-chip was fabricated with a fibre glass coating that hid the designs, see Figure 3-23, which made it difficult to target individual transistors. Hence, laser pulses were set to target the combinational block of all circuitries, the registers associated to each circuitry in Table 3-5, referred here as direct attack, and the registers of all circuitries at once.

The setup only allowed front-side attacks, which only Metal design and radiation hardened (RH) registers demonstrated to be immune against when they were targeted directly. No other countermeasure withstood direct attacks. ELT was expected to pass the tests, however it failed and was weaker than MOS circuitry. Leaving the reasons for ELT's failure aside, which could be design related or not, the area penalty resulting from using these transistors is too high to justify their use even if they were immune against laser attacks.

As expected, parity error detection only detected a limited amount of error injections and the combinational SEU mitigation circuitry failed to protect the combinational logic against error injections. Combinational SEU mitigation was designed to mitigate single particle impacts, and hence it failed when targeted with a spot size that affected several transistors or the whole countermeasure.

Initially it was expected that a laser attack would almost always set all registers to the same value, say logic 0. This would ease detecting error injections by defining the right parity policy for the adder registers, either even or odd parity. In reality however, the errors injected in registers changed for different stored values, laser energy level and laser target areas.

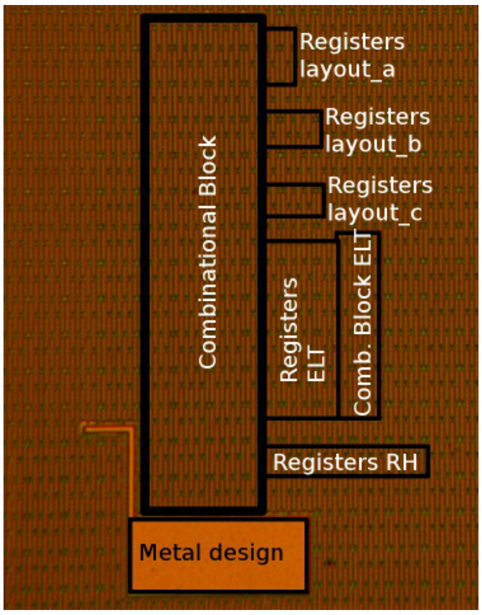


Figure 3-23 Silicon view with circuitry boundaries

The injected error value was more consistent for ELT transistor based registers; especially with high energy laser attacks. The injected error value would almost always be 0x1F when all registers are targeted at once, and 0x10 when only ELT registers were targeted.

An interesting behaviour was noticed when targeting the registers of all circuitries. In this case all registers became affected, including those in the Metal design and the RH registers. In other words, otherwise immune registers were prone to error injections when a high enough number of neighbouring transistors were attacked with a laser. This behaviour could be explained by the laser producing temporary localised shortcuts when a high enough energy level targets a high enough number of P and N transistors.

3.3.4 Conclusions of laser attacks

Only one usable countermeasure was immune to direct laser attacks, SERT SEU immune registers; although even this one would be affected when attacking enough neighbouring transistors. This means that no single countermeasure might be adequate as a universal remedy.

Metal layer countermeasure also succeeded in protecting the design against direct front-side attacks. However, the drawbacks of this technique discard it as a valid option in commercial devices. In one side is the raise of fabrication costs. The primary function of metal layers in a semiconductor device is connecting the different building blocks (e.g. transistors, capacitors, resistors). Placing metal plates to cover parts of an IC will consume valuable routing space, and could force the device to grow in area and/or using an additional metal layer.

Laying down metal layers in a semiconductor device is an expensive process, and so, designs must use the least amount of metal layers. A bigger die minimises the number of devices that can be fabricated in a given wafer. Hence, any of these two growing approaches minimise the benefit margin per device and could potentially force Atmel to increase price per die.

The other important drawback of a metal plate is its lack of protection to back-side laser attacks. These kind of attacks, not tested in this research, consist of milling the die substrate and applying the laser to the die’s back side. Milling the substrate increases the laser penetration depth and enables fault injection through back-side attacks. A metal plate can protect a circuitry from a front-side attack but would fail to protect it from a back-side attack. Hence, back-side attacks would make this countermeasure redundant. Back-side attacks also force to focus on structure and/or architecture based countermeasures such as RHBD.

A design’s sensitivity to neighbouring circuitry should be studied further, as it could provide clues as to how to harden certain critical functions, such as crypto blocks. It could also help defining sensors and design strategies to detect attacks on neighbouring circuitry.

3.3.5 Future work

This research line was cut short by a shift of interest; however, it would be interesting to test other radiation hardening structures similar to the SERT SEU, including custom ones. Alternatively, the commercial viability for using the same RH register tested here could be studied.

Again, an in depth knowledge of a design’s sensitivity to its neighbouring circuitry could be crucial to the success of any particular countermeasure and detection strategy. Finally due to the sensitivity shown by ELT transistors, its use as a laser sensor could be considered or analysed.

4 Low Power Design for Smart Card

With every new technology node, transistors get faster, smaller and more complex designs can be integrated in the same area. With new technologies nodes, the nominal supply voltage level is also reduced, which in turn lowers the dynamic power. However, deep-sub micron technologies are accompanied by other challenges, such as process variations (e.g. random doping) and an increase of the leakage current [43]. The leakage current is the direct contributor to static power consumption, which wastes valuable energy. This is an undesired side-effect, especially on mobile applications and devices with a tight energy budget, such as contact and contactless Smart Cards.

Despite the leakage current being still manageable at 180nm and 150nm technology nodes, smaller technology is already suffering severely from it. As a result, several techniques have been developed to minimise leakage or static power consumption, such as adapting dynamically the threshold voltage [44], power gating [45] or reducing Vdd [46], each with different success rates.

In Smart Cards, as with most SoCs, SRAMs are the main contributors towards leakage current, generating over 80% of the overall leakage current [47]. Hence, this research line was initiated to: a) look at SRAM leakage reduction techniques; and b) test the robustness and possible security issues related to the proposed techniques. This research line was carried out in collaboration with Atmel's Memory Group based in Rousset, France.

Three lines of work were initiated in relation to low power. The first one focused on the impacts and approach to enable memory partitioning, where a memory is divided into different sections or partitions and each powered independently according to their usage and contents. This work resulted in a proposal and design considerations when adapting the Smart Card SRAM to enable memory partitioning.

The second approach consisted of powering the SRAM continuously to a supply level lower than its nominal value, in order to reduce the static power consumption. Initial simulations carried out by the Memory Group suggested against this approach.

The third and final line of work focused on the security impact of a few leakage reduction techniques. This work was divided in two parts, one developing custom SRAM bit cells and another one testing SRAM memories with different leakage reduction techniques developed by the Memory Group when subjected to power attacks commonly used against Smart Card devices. Test setup issues prevented from carrying out the planned tests.

The three research lines are highly related to each other, and so, the next section provides a common literature overview. The next three sections cover the work carried out on each research line, including an introduction to the research line, the obtained results, the conclusions drawn in relation to that particular research; and future lines of actions. In addition to this, the memory partitioning research line expands the literature review.

4.1 Literature Overview

One of the side effects of smaller technology nodes is the increase of leakage current. This leakage is the result of shorter channel distances and geometries and, as shown in [48], the static power consumption is set to offset the dynamic power unless measures are taken to minimise the leakage. This increase of leakage current is especially critical for battery powered applications, as more energy will be wasted, resulting in a shorter battery life. This is added to the fact that the batteries' capacity has increased at a slower rate than the power requirements of portable devices.

Six different mechanisms contribute to the leakage; these are shown in Figure 4-1, where gate tunnelling oxide (a.k.a. gate oxide) and sub-threshold leakage are the major contributors [49]. Gate oxide leakage is the result of thin gate oxides, thus, this leakage mechanism needs to be tackled at foundry level and is out of the scope of this research line. Current proposals focus on using high K dielectrics [50]. The sub-threshold leakage appears when the gate is in weak inversion.

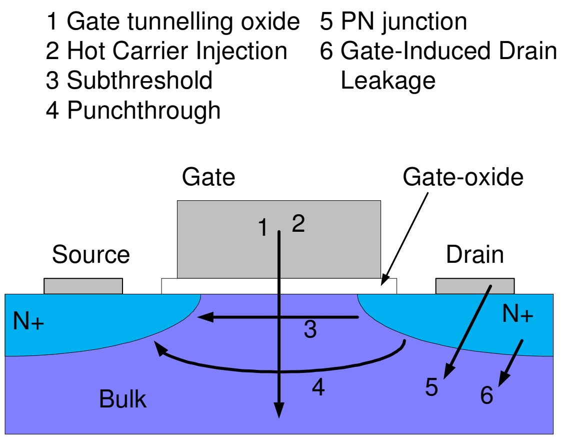


Figure 4-1 Leakage contribution mechanisms [49]

Several techniques have been developed to help reducing leakage current, ranging from low level (e.g. foundry and/or transistor level) to high level (e.g. system and/or software level) [46, 49]. On the low end, one approach is enabling the technology process having transistors with different V_t. This allows using high V_t transistors throughout the design, which are less leaky but slower, and reserving lower V_t transistors for critical paths, which are leakier but faster [51]. This approach, already used by Atmel, enables reducing the leakage of any design, be it a memory or logic.

On a typical device, its resources or functions are used on demand. This implies that some of these resources might sit idle for long periods of time. One approach to reduce the leakage on the unused functions is power gating them, and powering them on or off depending on whether they are needed or not [45]. This is achieved with the addition of sleep transistors, which are placed between the main power rails and each function, block or area's power rails or ring [45]. Power gating is the most effective approach to minimise leakage current, as it can reduce a function's leakage by a 97%, however, it comes with its share of issues. Powering up and down a block or function takes time and, unless it is managed carefully, it can impact on the device's performance.

Sizing the power gates correctly is crucial, as they need to be able to conduct the maximum current needed by the block powered through them. However, designing too big power gates would result in higher area overhead, but more importantly, longer power up and power down times, which reduces the number of times the block or function can be powered down.

When powering down a block or function, its outputs are left floating, which can cause leakage in the circuitry that reads these outputs. This can be fixed easily by driving the outputs to either a logic one or logic zero when the function is powered down, e.g. by using pass-gates. Another issue to consider is that when a circuitry is powered down, loses its current status, it is reset. If preserving the status is desired, the desired data storage units, e.g. registers, should not be powered down. In other words, data storage and combinational logic should have different sources of power. Alternatively, the circuit's current status could be stored to a memory, e.g. main data memory, before powering it down, and restored it once the circuit is powered back up again.

A similar approach to minimise the leakage when a function or block is not active is reducing its supply voltage, Vdd, instead of powering it down. Although this approach does not reduce leakage current as much as power gating, recovering from a low power mode is much quicker, hence reducing the impact on performance [52]. Furthermore, if Vdd is not lowered below the data retention voltage (DRV), the registers or memories within the block or function would not lose the stored data, hence, preserving its status.

However, lowering Vdd implies the need for a variable supply source. This could mean an additional, controllable, voltage regulator, or powering the functions or blocks through power gates with a sophisticated management or control.

This leakage reduction technique is very common when reducing an SRAM's leakage [46, 53-55]. However, reducing the supply voltage not only reduces memory's performance, but also its static noise margin (SNM), which determines the memory's robustness. New SRAM cells have been presented to improve the SNM. In [56] it was achieved with bulk biasing, whereas in [57-59] it was achieve by changing the SRAM bit cell's structure.

4.2 Memory Partitioning

Extensive research has been carried out on SRAM memories aiming to reduce their power consumption and leakage current. Leakage reduction techniques have primarily focused at the foundry level, transistors' design and SRAM architecture [49, 60]. The common factor in these techniques is that they focus at the hardware level, ignoring software's impact on power consumption. Other techniques, such as memory partitioning and power management [61, 62], not only allow saving power via hardware techniques, but also through analyzing the software applications' use of the memory, dividing a monolithic memory into different smaller memories and redistributing the data to minimise power consumption.

Memory partitioning and power management have been primarily used with the on-chip memory of 16-/32-bit CPU based devices, where the main memory is external. This work focuses on the unique challenges involved when applying memory partitioning to Smart Card devices, which are mostly designed around 8-bit CPUs with on-chip memory only.

The following sections cover the literature review of the memory partitioning approach and a description of the main issues when applying this approach to Smart Card devices. This is followed by a description of the simulated test cases to analyse a Smart Card SRAM memory usage and the obtained results. The next section covers the proposed memory partitioning implementation, followed by the discussion section. The last two sections draw some conclusion and suggest a few lines of action for future work.

Volume II includes the following reports related to and supporting this research line:

- LowLeakageTech1(SRAM Memory Partitioning for Leakage Reduction)

4.2.1 Literature Review

Memory partitioning and independently powering individual partitions has extensively been used on Cache and Scratch-pad Memories (SPMs) for reducing both, dynamic and static power [61, 62]. The dynamic power consumed by a circuit is defined by its effective capacitive load, the supply voltage and the frequency the circuit is exercised at, just as defined in (1). Smaller memory arrays have a smaller capacitive load for read/write operations than bigger ones, hence, partitioning or dividing a monolithic memory into smaller size memories results in reducing the dynamic power consumption [61]. The static power consumed by a circuitry, on the other hand, depends on the supply voltage and its leakage current. Hence, by reducing the supply voltage of the non accessed partitions, the static power of these partitions is reduced too [62, 46]. The reduced supply voltage needs to be above the DRV to avoid loss of data [46, 52].

$$P_{dynamic} = C_{eff} \cdot Vdd^2 \cdot F_{clock} \qquad (1)$$

When partitioning a memory, it is important to know: (a) the partitioning approach to be targeted; (b) partition sizes or granularity; and (c) the partitions' powering policy. Programming software tools also play a key role on the effective usage of partitioned memories.

4.2.1.1 Partitioning

Two partitioning approaches are possible: uniform and non-uniform. Uniform consists of dividing the memory into equally sized partitions. Whilst this is the simplest approach, it does not necessarily maximize the energy savings; especially on mono-application systems [61]. Non-uniform partitioning consists of dividing the memory into different size partitions, so that the most used variables can be stored in the smaller partition in order to further reduce the dynamic power. This partitioning approach is better suited for mono-application systems and the cases where there is a prior knowledge of the code to be run. The main drawback of this approach is the need to know the code to define the partition sizes; this is, before designing the device.

4.2.1.2 Partition sizing or granularity

Partitioning a memory increases the area overhead, as the periphery overhead increases with the number of partitions [62]. This penalty is more significant when partitioning smaller memories. Furthermore, additional control logic and power management registers will be needed. Hence, the optimum partition sizes would be determined by those sizes where the difference between energy savings and area cost is maximized.

The energy efficiency of a partitioned memory is determined by its usage, in other words, by the code run by the CPU. When the code is available, optimum [63] or sub-optimum [62] partition sizes can be determined by analyzing the memory usage and grouping the most used variables into one small partition, and the least used variables into the biggest partition. The amount and size of each partition required to maximise power reduction is, hence, application dependant.

For the cases where the code is not available, the number of partitions and their size can be determined by comparing the cost of different partitions and their impact on the dynamic power.

4.2.1.3 Powering policy

Partitioned memories can be defined by their contents and use as: a) being accessed; b) not accessed but with valid data; or c) not accessed and without valid data. An accessed partition contributes towards dynamic and static power, whereas not accessed partitions only contribute towards static power. Based on these definitions, accessed partitions are powered to nominal Vdd so that they can be read and written to without losing on performance; not accessed partitions but with valid data are powered at a lower supply, so that their static power consumption is reduced; and partitions without valid data are powered off to avoid their contribution towards static power [61, 62, 58, 64].

Techniques to low power a partition include dynamic voltage scaling (DVS) and power switching. Since all partitions can be powered independently, this technique requires a more complex voltage regulator and/or supply power distribution mechanism. This technique is mainly targeted at the partitioning level, although it can also be used at individual address level. DVS must be used with care on noise prone environments, as low powered memories are susceptible to soft errors and fault injection [65]. This weakness can be minimized with bit cell architecture modifications such as [57, 58]. An alternative static power reduction technique is to reduce leakage current by forward/reverse biasing. Although this technique could target address level more easily, the decreasing effectiveness of forward biasing with smaller technologies [66] and the need of triple-well for reverse biasing makes them inefficient or too expensive.

Powering off partitions can be achieved through power gating. Two approaches are possible, positive metal-oxide semiconductor (PMOS) or negative metal-oxide semiconductor (NMOS) transistors. Although power gating does not fully eliminate the leakage current, the work carried out in [45] shows that it achieved up to 97% of energy savings.

Switching power states results in a power consumption and a performance penalty [62], which is worst for higher voltage differences between states. These penalty effects are usually mitigated with timing policies that determine the maximum number of cycles between accesses before a partition goes to low power mode.

Despite the timing policy missed accesses will still occur, where accessing a not ready partition is attempted. For these events, the CPU and memory should be provided with a wait/ready signal, which is usually a standard in a CPU.

4.2.1.4 Software techniques

Software tools, compilers especially, also play an important role in saving power, as these can greatly impact on the memory usage [63, 67-69]. Two main approaches have been proposed to manage data in non-uniform SPMs: static and dynamic allocation. A static approach consists of placing the overall most used variables in the smallest partition for the whole execution time [70]. Dynamic approach, on the other hand, consists of moving the contents between the partitions and the main memory in order to maximize the usage of the smallest partition. This is, in order to maximize power saving [69].

A static approach can result in a high enough power saving for mono-application software codes, whereas a dynamic approach is more suited to multi-application codes. In any of the cases, new software tools are required to allow programmers develop code abstracting from the memory layout and in order to exploit the benefits obtained from partitioning the memory. This requirement might also mean that for a particular IC, there are less available software tools than for others. This is particularly important when taking into account studies such as [71], which showed the importance of the available software development tools when deciding which CPU to use on a project.

4.2.2 Considerations when partitioning the memory of a Smart Card

When applying memory partitioning to Smart Cards, there are three main aspects that differ from other cases where this technique has been applied and which can impact on its use or implementation. The first difference is that previous works focused on either cache and/or SPM memories, leaving system the memory aside. In Smart Cards, there is not cache or SPM memory; instead, the leakage reduction approach targets the whole system memory. This fact limits the amount of memory that can be set in lower power mode or even powered off.

Furthermore, the data space of an 8-bit CPU is usually limited to 64Kbytes, although accessing more memory is still possible by using memory pagination. Nevertheless, an 8-bit CPU based Smart Card's internally SRAM memory is usually below 8Kbytes. Since the periphery overhead is higher for smaller memories, partitioning a memory into separated smaller memories might not be cost effective for a Smart Card.

The second difference is that some previous works required prior knowledge of the target application and or applications to be run in the system to optimise the memory partitioning. Due to the security level required with Smart Cards, manufacturers have no access to the OS nor the applications run by their customers. Furthermore, different customers can use the same device in different ways. This diversity makes it difficult to estimate the memory usage and the partitioning formulation for any one product. Also, optimising a particular device for a particular application or customer might not provide other applications or customers with extra benefit.

Instead, studying a Smart Card’s general behaviour and software architecture could help estimating a memory usage pattern and formulating a memory partitioning approach. For example, a Smart Card usually stays idle until it receives a command. It then executes the command and sends back the appropriate response. The Smart Card could then go to an idle mode again if no further commands are received within a certain amount of time. With this behaviour example, a partition could be assigned to act as a communication buffer, and power it on or off to save on static power consumption depending on the Smart Card’s status.

Regarding the software architecture, the OS is likely to allocate certain memory sections dedicated to different OS related functions. If these OS functions can be related certain memory partitions, the static power consumption could be reduced by powering off those partitions related to the unused OS functions. Since the OS and application are likely to locate their data in different memory regions, a similar memory behaviour and or usage analysis could also be applied to the application data.

Finally, Smart Cards use a memory scrambler that interfaces between the CPU and the data memory. The scrambler maps logical memory addresses into random physical ones, where logical memory addresses are the ones the CPU is aiming at and physical memory addresses are the ones the CPU is actually accessing. The relationship between the physical and logical memory can be changed over time, which can make it harder to implement some previously stated techniques. One example would be non-uniform partitioning, as with the current scrambler, two addresses belonging to the same logic partition could end up in different physical partitions.

Although the scrambler is capable of scrambling the whole memory as one unit, a typical scrambler implementation approach is to divide the memory into identical sections and scramble the addresses within each section. Further security can be achieved by scrambling these sections.

These aspects are covered in more detail in the section 3 of the report LowLeakageTech1. The next case study provides a better inside of the memory usage for different operations and how the scrambler can impact on the physical memory usage.

4.2.3 Smart Card memory usage case study

When it comes to studying the memory usage of a Smart Card under normal operation circumstances, ideally, customers' OS and applications should be run. However, as previously stated, this is not possible. Instead, for this case study in-house developed OS and applications were used. The OS, which for confidentially reasons will be referred to as Smart Card OS (SCOS), was designed to run on a Smart Card which, again, for confidentiality reasons will be referred to as Smart Card Device A (SCD-A). The Smart Card in particular is unimportant for this test, as the accessed addresses will depend on the OS and applications, not the Smart Card.

SCD-A has built-in cryptographic capabilities and comes with 6Kbytes of SRAM memory divided into two 3Kbytes memories. Built in 0.18µm technology and powered at 1.8V, each memory had a leakage current of 2.1µA and a dynamic power consumption of 17µW/MHz.

The SCOS was statically linked–with fixed size variables and in fixed locations. The memory needed by the OS and applications ranged from 512 to 3Kbytes, leaving another 3Kbytes of unused memory at all times. Furthermore, taking this device to low power mode would result in losing all the SRAM contents but the first 128 bytes.

At first glance, the memory's leakage current could be approximately halved if one of the two 3Kbytes memory instances could be powered down. Additional power savings could be achieved by dynamically enabling and disabling SRAM partitions according the application's needs. Finally, and considering that setting the device to low power mode results in the loss of all but 128 bytes, further power savings could be achieved if all but the first 128 bytes can be powered down when the Smart Card is set into low power mode.

Despite this case study only uses a portion of the available memory (50%), and one could expect that other OS and applications might have a higher memory usage, this case study is still a good indicator of the potential benefits of this technique.

The section 4 of the report LowLeakageTech1 analyses in detail the memory usage of the SCOS and SCD-A combo running three different applications:

- Application 1; basic application protocol data unit (APDU) commands
- Application 2; Smart Card personalisation

- Application 3; several consecutive random number generations

The impact of a scrambler can be appreciated clearly by comparing Figure 4-2 with Figure 4-3. Figure 4-2 shows the logical memory addresses used by the three applications. Figure 4-3, in the other hand, shows how these logical addresses are spread across the physical memory. In the case of Figure 4-2, the memory distribution is compact and localised, whereas in the case of Figure 4-3, the memory is spread by inserting gaps in un-used memory. This clearly indicates that the scrambler will impact on the memory partitioning policies for Smart Card devices, and that these policies might also have an impact on the scrambler.

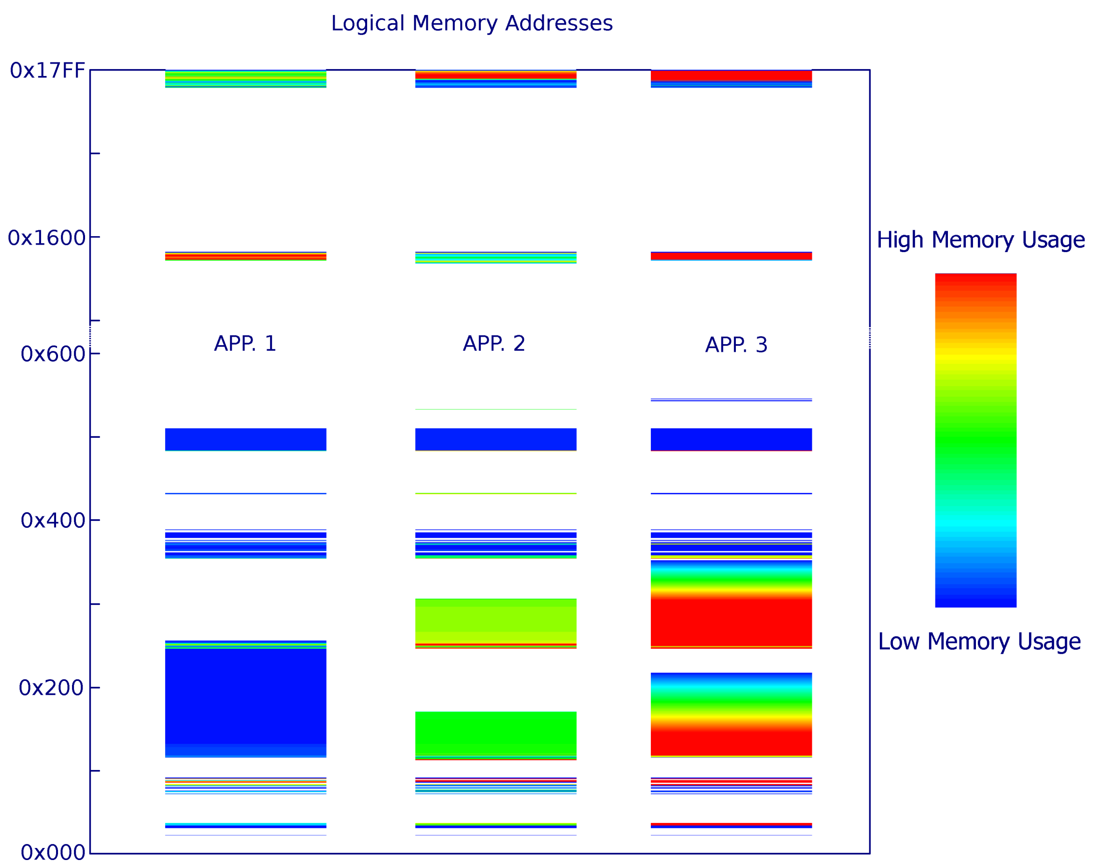


Figure 4-2 Logical memory usage of different applications. Blue indicates low activity, red indicates high activity.

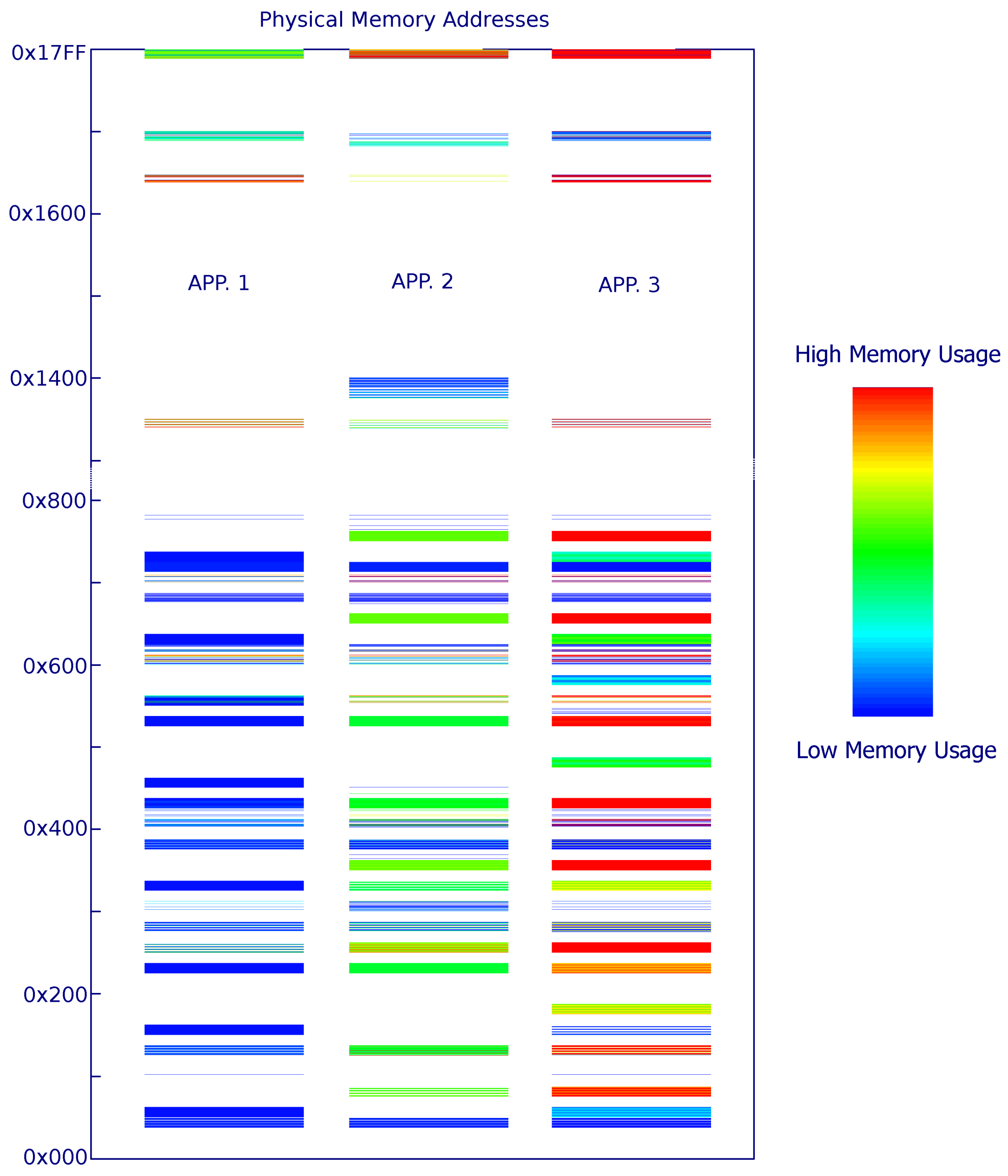


Figure 4-3 Physical memory usage of different applications. Logical addresses are scrambled. Blue indicates low activity, red indicates high activity.

These figures also show how the memory usage both in spatial and frequency terms vary for different applications. Here, the blue colour indicates small amount of accesses. The red colour indicates a high amount of accesses.

4.2.4 A Proposed SRAM Memory Partitioning for Smart Cards

Based on the awareness raised by the previous sections, this section highlights the main aspects of the SRAM memory partitioning approach and powering policy for Smart Card applications proposed in this research line. The aim of which is saving on the static power consumed by the SRAM memory.

The aspects covered here include the partitioning method, the partition size, the powering policy, the memory scrambler and considerations to be taken when coding for such memories.

4.2.4.1 Partitioning

Two partitioning approaches can be suggested. Firstly uniform partitioning. This approach is mandated by potential memory usage differences from different customers and applications. Also, this partitioning approach would ease the memory scrambling and the scrambler's design. The second option could be non-uniform partitioning. Here though, the bigger partition's size should be a multiple of the smaller one (e.g. 256 and 512 byte partitions). Such non-uniform partitioning could be easily managed by the same scrambler used for the uniform partitioning approach.

4.2.4.2 Sizing

Partitioning an already small memory into smaller independent instances has a high area overhead. The alternative here is partitioning only the memory array, where the minimum partition size is determined by the array's natural partition. Natural partitions are considered those sections of the memory array surrounded by power tracks, just as depicted in Figure 4-4.

The amount of partitions and their sizes will depend on the total amount of available memory and the device's target application. Each partition will need a control bit to turn it on and off, which could be located with the CPU's special function registers. The more available partitions, the more registers are needed to control their state; hence, limiting to a maximum of 8 partitions is suggested (one additional special function register).

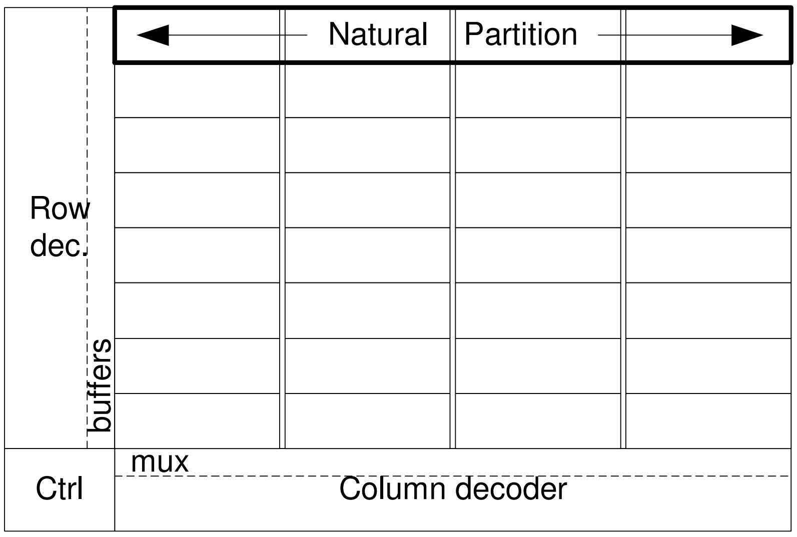


Figure 4-4 Typical SRAM organization showing natural partitions

Another aspect of partitioning the memory array instead of the whole memory is that, for the memory in Figure 4-4, it does not necessarily result in a reduction of the dynamic power, as the pre-charge logic still sees the same capacitive load. Some read/write capacitive load values can be reduced by embedding the column decoder block into the memory array, as shown in Figure 4-5, and isolating the bit-line and bit-line# wires of the north and south partitions. Still, even in this case, the capacitive load might not be reduced to its minimum, as the sub-arrays could be grouped into several partitions. This case is shown in Figure 4-6, where the north sub-array consists of four partitions and the south one consists of two. In this scenario, accessing Partition 1 or Partition 6 would result in similar dynamic power consumption, although less than accessing the same size partitions in the memory unit in Figure 4-4.

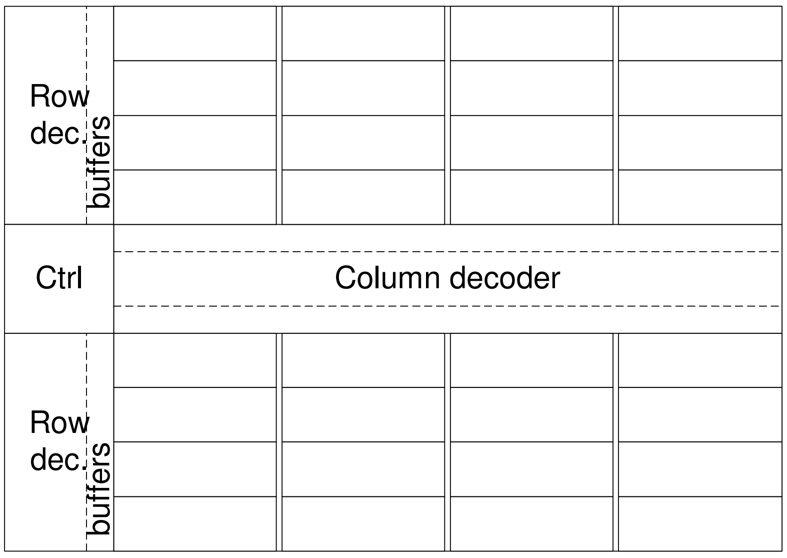


Figure 4-5 SRAM divided into two memory arrays

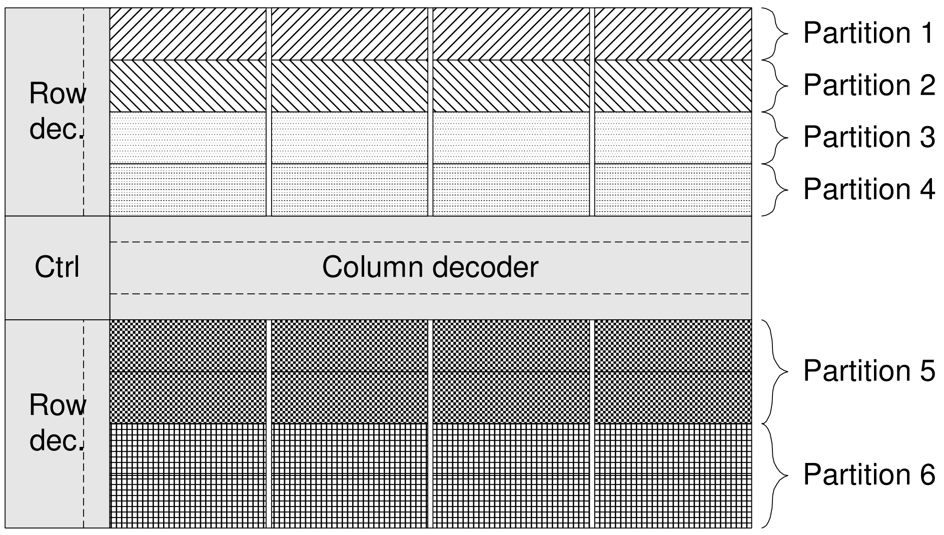


Figure 4-6 Example of two size partitions

4.2.4.3 Powering policy

Ideally a powering policy with three power states would be used:

- fully powering a partition being accessed;
- low power partitions with valid data but not being accessed;
- and, powering off empty partitions

However, partitions in low power mode are potentially susceptible to soft-error injections by glitch attacks. Hence, only two possible power states are suggested:

- Power ON: this mode should be used with all partitions with valid data, whether they are being accessed or not.
- Power OFF: this mode should be used with empty partitions.

Each partition should have its own power ring so that they can be powered independently. Work [52] shows that PMOS power gates resulted in 86% energy saving when in standby with no impact on area or relative read time. Hence, using PMOS transistors for power gating is suggested.

4.2.4.4 Scrambler

The memory partitioning approach should be reflected on the scrambler circuitry. Logical and physical memories should have the same number of partitions and of the same size. The best solution here is having a two tier scrambler, where one tier scrambles the addresses within a memory partition, and the other tier scrambles the partition order. Figure 4-7 shows an example of how the scrambler could work. Here, the first tier scrambles the partitions' order, and the second tier scrambles the addresses within each partition. In this example all partitions share the second tier scrambler; hence, their contents are scrambled in the same way.

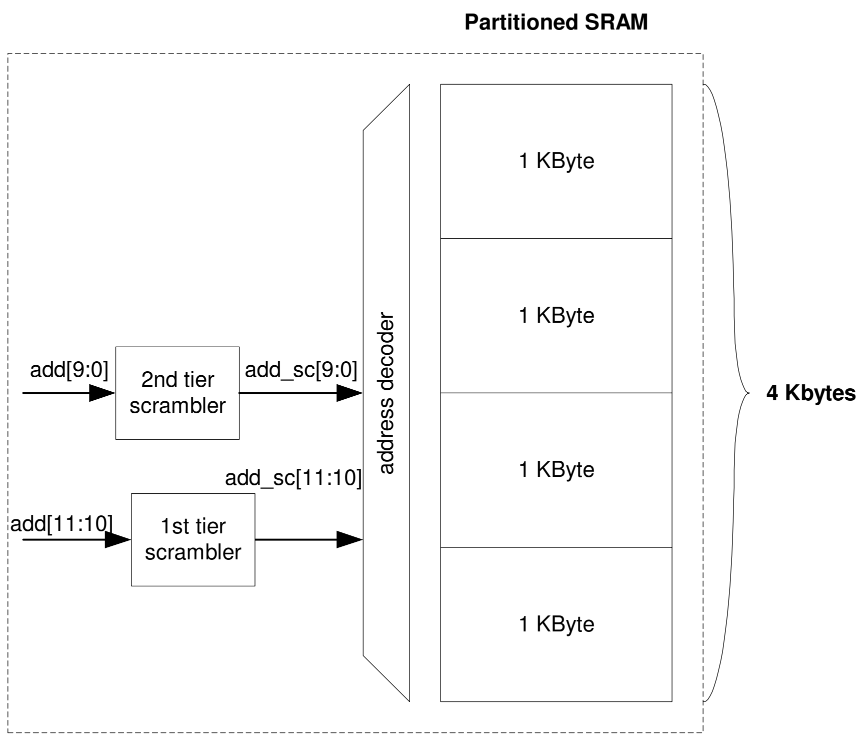


Figure 4-7 Example of a two tier scrambler

4.2.5 Potential power savings on SCD-A with a partitioned memory and coding considerations

High level programming language compilers for Atmel Smart Cards are developed by third party companies. Whilst ideally compilers should take care of the memory management, adapting the compilers to accommodate a partitioned memory might be costly and might not be guaranteed.

For the case that there is not compiler support, software developers should manage the memory power state. Two approaches could be used to control the partitions' power state: a) initialisation only; and b) ad-hoc mode. Initialisation consists of powering off the partitions that will not be used under any circumstance. This approach can only be used in the cases where there is more memory available than needed, such as in the case study covered before.

The SCD-A's SRAM is made of two 3-Kbyte memory units. When running the SCOS on the SCD-A, only less than half the available SRAM memory is used. With the proposed memory partitioning approach, all the required memory could be concentrated into a single memory unit and power off the non-used memory unit. Based on the power saving figures of [45], the proposed partitioning approach could result in saving up to 43% of the 6-Kbyte SRAM's current static power. This power saving figure could be achieved within a few instruction after powering up the SCD-A by just commanding the memory power management unit (MPMU) to disable or power off the unused memory unit. This is a limited yet simple approach to reduce power consumption with a minimal code overhead.

Ad-hoc approach can result in further power reduction by powering memory partitions on or off as they are needed. Powering partitions on and off increases the dynamic power consumption [62], so this technique should be used with care. The energy cost of powering off a partition should be lower than energy cost of keeping it powered on.

Carefully analysing the OS memory usage shows that certain memory blocks could be linked to certain functions. Some functions are needed for specific tasks and/or for a determined amount of time. After these tasks have been completed, the value of these memory blocks can be ignored. Powering off the partitions related to these memory blocks could help further reducing the static power consumption. An obvious example is the APDU communication buffer. Once an APDU command is received and decoded, if no further access is required to the communication buffer until the command has been executed and a replay is being constructed, then, the communication buffer could be powered off to save on static power. Again, the energy cost of either power state would determine the applicability of this policy.

Another example could be the case of the execution of a routine or function which has a considerable need of memory (e.g. a complex arithmetic function). Instead of using the heap, a partition could be assigned to this function, so that the partition is powered on just before the function begins its execution, and powered off after completing it. For this case, powering the partition on and off could be done from within the function.

Yet another method to save energy is when the Smart Card goes into IDLE state, where all data in the SRAM is lost except for the lower 128 bytes. With the proposed memory partitioning approach, when the Smart Card goes into IDLE state, only the logical partition 0, $P0_L$, would remain powered on (automatically controlled by the MPMU). Again, taking the work [45] as a reference, this partitioning approach could save up to 86% of the static leakage of the periphery logic and up to $((n-1)*86/n)$% of the static leakage generated by the partitions, where n is the number of partitions. On the SCD-A case, with 8 partitions per 3KB memory instantiation (i.e. 16 partitions in total), the partition leakage reduction could be around 80.63%

All power saving figures are based on hypothesis and estimations, hence simulations need to be run to validate these values. An environment similar to that described in [72] could be used to simulate the energy savings by the proposed memory partitioning approach.

4.2.6 Conclusions

It has been shown that memory usage is highly application specific. The effectiveness of memory partitioning as a leakage reduction technique, then, depends on the thorough study of the memory usage by these and other applications, as well as by understanding how variables are used by the OS and applications. The case study has also shown the potential power savings by just powering embedded monolithic memories independently.

Also, the pros and cons of this technique have been argued, and possible solutions to the different issues here discussed have been presented too. Undertaking such approach will have an impact to the whole system and process, starting from the SRAM module itself, the Smart Card system, the compiler and even the programming approach. The success of this approach is tied to the success at each of these levels.

4.2.7 Future work

Atmel's Memory Group should decide on the feasibility of the proposed partitioning approach and the continuation of this research. Also, the memory access analysis could be refined so that OS SRAM accesses could be differentiated from application ones. If necessary, additional applications could be simulated too.

Finally, if this topic is to be followed, methods for managing the partitions at compile time should be investigated, and compiler designers approached to integrate these methods in their compilers.

4.3 Supply Voltage versus Performance

Supply voltage reduction is being used to minimise the leakage current. Reducing the supply voltage of a device, however, also reduces its performance. Applied to SRAMs, the main approach aims to reduce the memory's supply voltage when it is not accessed (idle mode), and raising the memory's supply voltage back to normal when it is being accessed (active mode). Accessing the memory in low power mode could reduce the device's performance, but more importantly result in a read/write failure.

Some recent works have solved this memory access limitation and achieved successful read and write cycles with SRAMs in low power mode and even at sub-threshold voltage levels [56-58, 73]. Their key achievement is, undoubtedly, the ability to access the SRAM at such low supply voltages without incurring on a read/write failure.

On Atmel Smart Cards, the maximum performance of the SRAMs can be up to double of the maximum performance they can be subject to. In other words, these SRAMs can be clocked at up-to twice as fast the AVR can be. This raised the question of the feasibility of powering the SRAMs constantly to a voltage level below their nominal value in order to reduce the leakage current but without losing performance.

The initial analysis of this proposal was made by the Memory Group, as the SRAMs needed to be characterised and they already had the right environment to do so. The contributions in this work relate to its proposal and the analysis of the Memory Group's results. The following sections highlight the Memory Group's work and conclusions, followed by my discussion on the results and possible future lines of action.

The following sections cover the results obtained by the Memory Group; a discussion on these results and the validity of this leakage reduction technique; and the conclusions of this research line. Finally, some possible future developments are proposed. Since the simulations were carried out by the Memory Group, there are no supporting reports in the Volume II.

4.3.1 Results of the simulations carried out by the Memory Group

The Atmel Memory Group carried out some simulations of the critical path of a 3Kx9 SRAM typically used in Smart Card devices. The SRAM was modelled on a 0.13µm technology node and the simulations where run on the following two corners:

- Typical parameters at 25C

- Worst-case parameters at 125C

The simulations looked at the access time and the read SNM of a SRAM cell for different supply voltages, ranging from 0.5V to 1.2V. The relationship between the access time and the supply voltage of the 3Kx9 SRAM is shown in Figure 4-8. The SNM variations in relation to the supply power are presented in Figure 4-9.

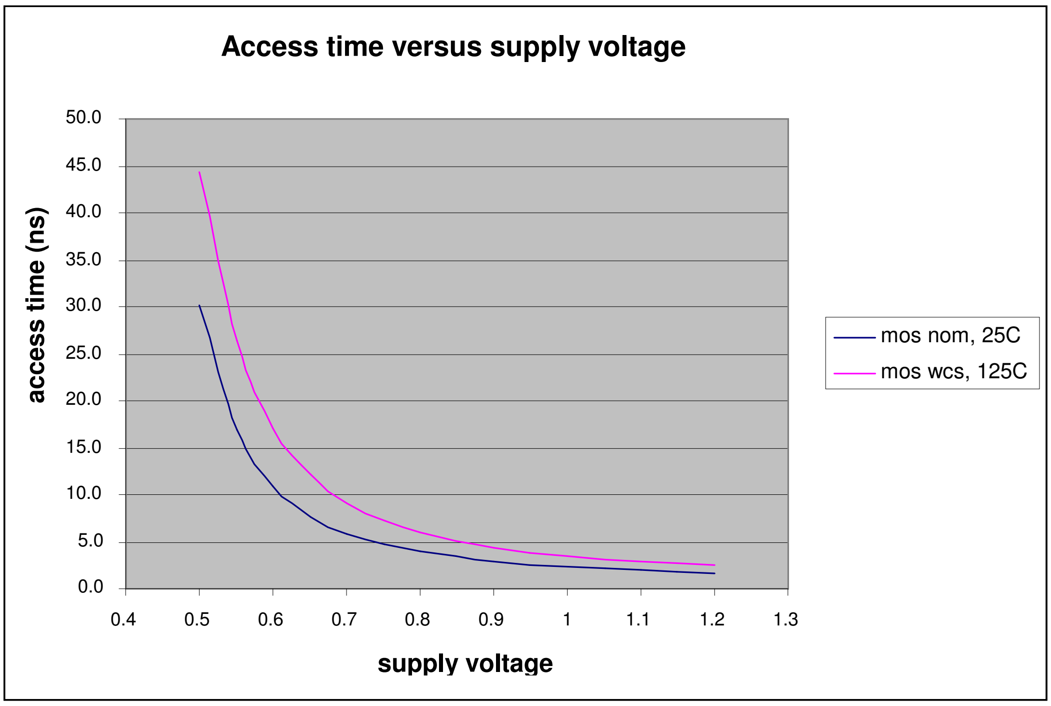


Figure 4-8 Simulated SRAM access time vs. supply voltage [12]

In their report, [12], they concluded that these results show a trend in the access time in relation to the supply voltage. A key difference between the simulation results and a real device, mainly at 0.13µm and below, is the effects that random doping fluctuations have on transistor mismatches and threshold voltage variations. These process variations would be the limiting factor for the lowest possible supply voltage, not the access time.

Finally, they proposed to run Monte-Carlo simulations with random doping fluctuations. However, the transistor models of the technology used for these simulations did not allow this feature.

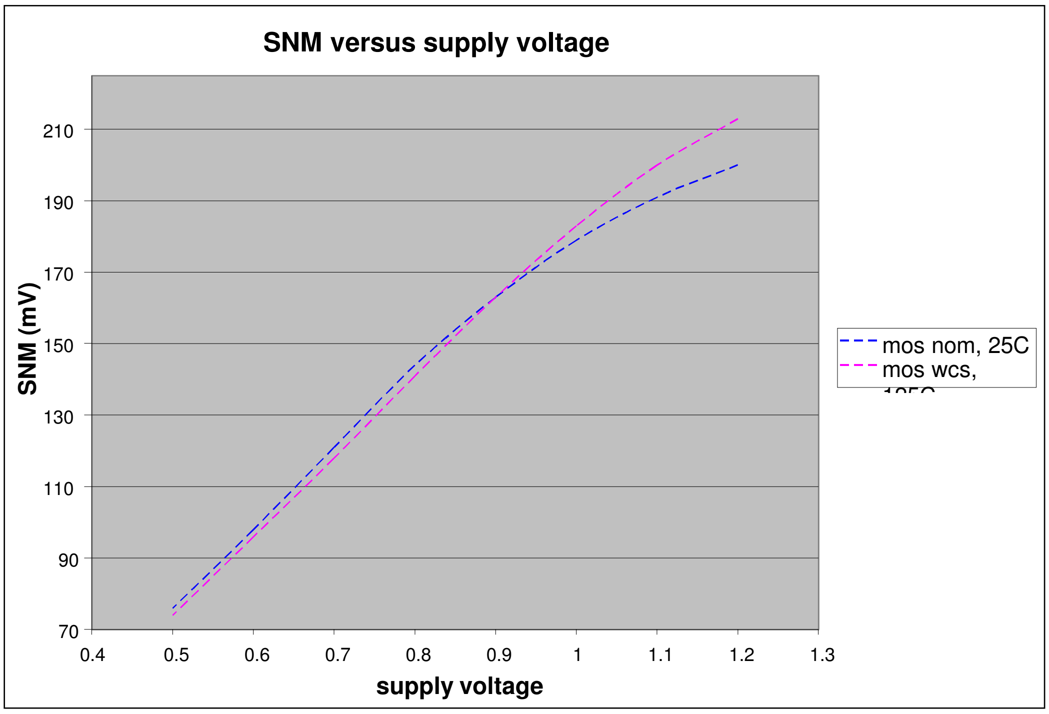


Figure 4-9 Simulated SNM vs. supply voltage[3]

4.3.2 Discussion

Reducing the supply voltage increases the access time and reduces the read SNM. An increase of the access time impacts solely on the system's performance, whereas decreasing the SNM impacts on the SRAMs robustness. From data retention and security perspective, the latter one is more important that the former one. As the Memory Group report highlighted, the SNM can be particularly bad for lower supply voltages and, while this is true, it is also true that simulation results depend on the model characterisation. In fact, the use of better models was suggested in their report, so that simulations could produce more accurate SNM results. This can offset the actually available SNM data in any direction, for good and for bad.

The main limiting factor raised in their report against the proposed leakage reduction technique was the process variation. However, SRAM designs capable of running below its nominal supply voltage have been presented, [56]. This could imply several things. On the one side, that the characterisation model used in these simulations is not accurate enough at voltage levels below their theoretical nominal supply voltage. This can result in misleading the simulation readings. If this is the case, this would be an issue that needs to be tackled; as with each new technology process, low power becomes more and more important, and the use of

[3] Figure generated from the SNM data available in the report [12]

several power sources could eventually become a standard even for Smart Card devices. Hence, the libraries or models should expand their characterisation range.

It could also mean that Atmel's process variations are too high in comparison to their competitors or other commercial processes. Process and foundry changes are out of the scope of this research. Hence, this has not been looked further. However, and if this is the case, the trade-offs and benefits from developing a tighter process control should be balanced out. How does tightening the process affect the yield? How does it benefit the low power design? Would it be possible to develop a process focusing on low power or improving current ones?

It might also imply the need to develop new SRAM bit cell architectures to improve the read SNM at lower voltages. This approach has actually been followed on several works where they have powered the SRAM well below their nominal supply power. Some of these works have even achieved read and write cycles at sub-threshold voltage. This seems to imply that, after all, the leakage reduction technique proposed here might still hold valid for a few technology generations to come.

Finally, and in addition to these technology limitations, the implementation costs should also be analysed. This leakage reduction technique implies the Smart Card needing two internal voltage regulators; one for the SRAM and another one for the rest of the device. Like the main voltage regulator, the SRAM voltage regulator should also have built-in glitch detectors. All this additional circuitry not only increases the area, but also offsets the power consumption. Furthermore, the need for voltage level shifters on the SRAM's address and data buses would increase the memory access time. Hence reducing the SRAM performance and limiting the minimum voltage level it can be powered at without reducing the Smart Card's overall performance.

4.3.3 Conclusions

Despite initial simulation results and analysis, that imply the unsuitability of this approach, recent papers have shown that it is possible to access SRAM memories when powered at a lower voltage level than their nominal value. There are three possible reasons why initial simulation results might have induced to think the unsuitability of this approach: a) library models not being accurate enough at lower supply voltages; b) Atmel having a technology that suffers a relatively high process variation; and c) the need of designing a new SRAM bit cell that results in better SNM when powered even at low supply voltage.

4.3.4 Future work

Three main lines of action could be considered for future work: a) determine the accuracy level of the simulation models when running at low voltage level; b) simulate the memory's critical path modelling process variations; and, c) estimate the implementation costs in terms of power and memory access time overhead.

4.4 *Memory Cell Hardening*

As already covered, reducing a SRAM's supply voltage reduces its performance and SNM, and this last parameter determines a memory's robustness. In the Smart Card business, losing out on robustness is not an acceptable trade-off. Some leakage reduction techniques have boosted the SNM when in low power mode so that successful read and write operations can be achieved even when the memory is in low power mode [57]. The SNM obtained by these techniques is, however, still below the 200mV defined by [12] as the minimum safe value.

This research line was initiated to: a) evaluate how different leakage reduction techniques would respond to attacks Smart Cards can be subjected to; and b) determine how a custom SRAM bit cell impacts the memory's robustness. The first part would use a test-chip developed by the Atmel Memory Group, 01HLB. Unfortunately, constant delays in manufacturing another test-chip needed for the test, the 01VGA, and later issues with the test setup impeded the evaluation of the different leakage reduction techniques instantiated in 01HLB.

The following subsections provide a better idea of what was achieved and what was intended to be achieved in this research. The first subsection covers the custom SRAM bit cells and their simulated SNM for different supply voltage levels. The next subsection discusses the tests intended to be run on the SRAM memories with leakage reduction techniques instantiated in the 01HLB test chip. Finally the future lines of action are laid-out. There are no supporting reports in the Volume II.

4.4.1 Custom SRAM bit cell design and simulation results

Custom cell bits were designed, all revolving around the same principle, increasing the cell's hysteresis as an approach to minimise undesired flips and increasing the SNM. These custom SRAM cells were based on Schmitt triggers instead of standard inverters as Schmitt triggers have build-in hysteresis, which in theory should harden the cells against undesired bit flips. Four designs where instantiated in total:

- Standard cell, a standard 6 transistor cell, such as the one depicted in Figure 4-10, was instantiated for comparison purposes.
- Schmitt half, Figure 4-11, where partial Schmitt triggers were used instead of inverters.

- Schmitt half sf weak feedback, same as the previous one but with a weakened feedback.
- Schmitt 2, Figure 4-12, where both SRAM cell bit inverters were replaced with a Schmitt inverter.

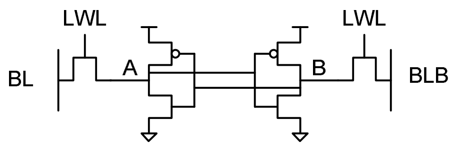


Figure 4-10 Standard 6-bit transistor memory cell

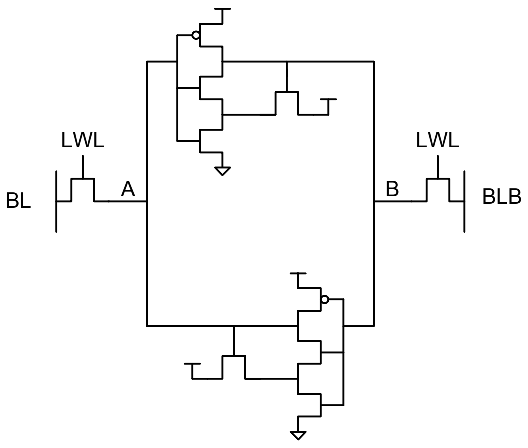


Figure 4-11 Half Schmitt inverter based memory cell

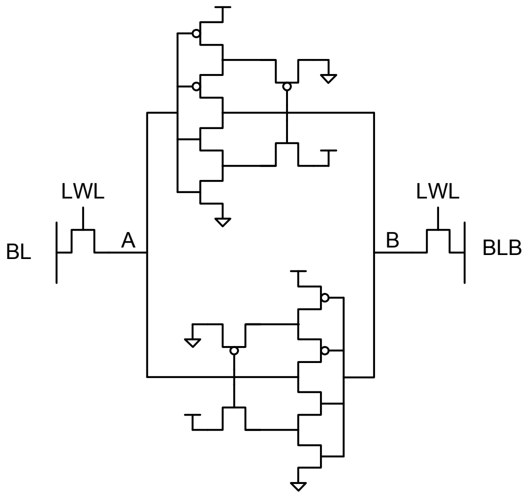


Figure 4-12 Schmitt inverter based memory cell

Simulations were run to calculate the read SNM of each of these four memory cells. These results are collected in Table 4-1. Graphical representations of the SNM for the standard cell and the Schmitt 2 cell for Vdd supplies of 1.6v and 0.7v are shown in Figure 4-13.

Table 4-1 SNM of different bit cell designs for different Vdd values

SRAM cell type	Vdd					
	1.6v	**0.8v**	**0.7v**	**0.4v**	**0.2v**	**0.1v**
Standard	0.3159v	0.1795v	0.1505v	0.0753v	0.0291	0.0042v
Schmitt half	0.5087v	0.2371v	0.2047v	0.1032v	0.0358v	0.0027v
Schmitt half sf weak feedback	0.5001v	0.2485v	0.2151v	0.1135v	0.0418v	0.0029v
Schmitt 2	0.5138v	0.2416v	0.2034v	0.1019v	0.0334v	0.00082v

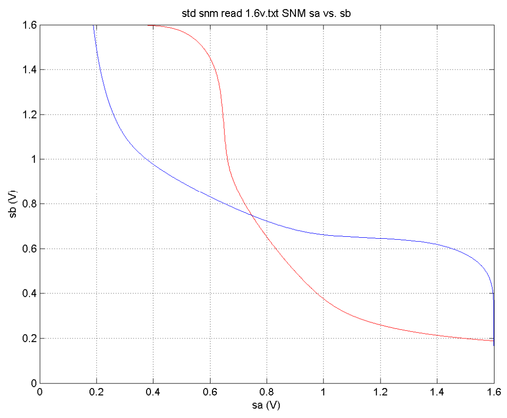
Figure 4-13 a) and b) show the standard memory's SNM when Vdd is set to 1.6v and 0.7v respectively. Figure 4-13 c) and d) show the Schmitt 2 memory's SNM when, again, Vdd is set to 1.6v and 0.7v respectively.

Simulation results show how custom SRAM bit cells provide a better SNM than the standard one. Taking 200mv as the absolute lower limit [12], standard cells would struggle when Vdd goes to half the nominal supply voltage, whereas custom cells could achieve supply voltage lower than half the nominal one. This can also be seen in the SNM diagrams in Figure 4-13, where the custom cell has a higher switching point than standard one.

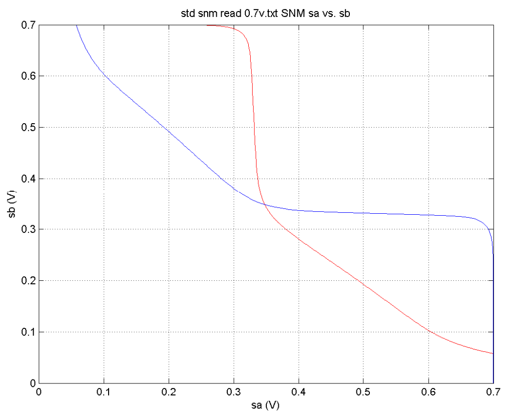
The use of Schmitt inverters for the custom SRAM cell has resulted in increasing the SNM. However, there are other aspects where an SRAM has to excel on too; these are area and power consumption. Schmitt inverters are bigger than normal ones, and hence, the custom SRAM cells using in these simulation are also bigger than a standard one. The custom SRAM cells' sizes are 10 and 14 transistors, whereas standard cell's are made of 6 transistors.

The current consumption for the custom cells was not measured. However, it is well known that Schmitt inverters can be leaky when the input value gets close to the switching point. From the SRAM's point of view, when the value of a bit cell is to be changed, the bit cell inputs would be set to voltage values close to Vdd and GND and away from the switching point. Hence, the hysteresis of the Schmitt inverters might not affect the write access, yet hardening it against undesired flips.

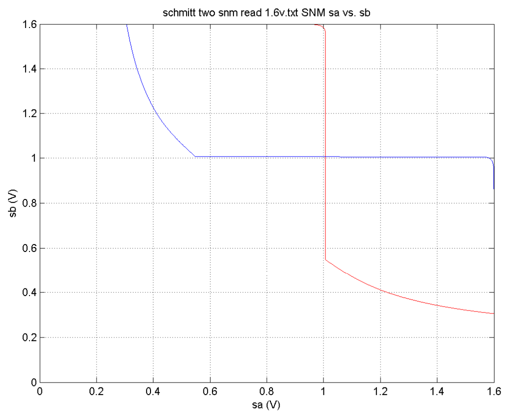
However, if the cell inputs are set to values close to the switching point, the leakage reduction achieved by any other technique could be offset by the leakage added by this memory cell.



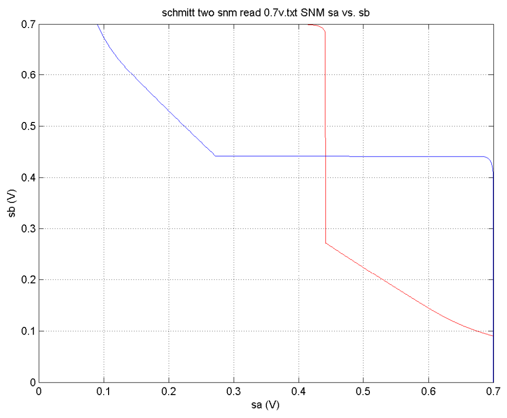
a)



b)



c)



d)

Figure 4-13 SNM graphs for different SRAM bit cell structures and supply voltages: a) SNM of a standard bit cell powered at 1.6V; b) SNM of a standard bit cell powered at 0.7V; c) SNM of a Schmitt half sf weak bit cell powered at 1.6v; and d) SNM of a Schmitt half sf weak bit cell powered at 0.7v

4.4.2 Silicon Test Methodology

The Memory Group designed a test-chip, 01HLB, which instantiated six different 64KByte memory, one standard and five with different leakage reduction techniques .These memories were evaluated by the Memory Group for behaviour and power consumption. However, their robustness when subjected to typical Smart Card attacks was not tested. In fact there is no available data regarding the behaviour of standard Smart Card SRAM's when subjected to glitch attacks either. Hence, the standard SRAM instantiation in 01HLB would be tested against glitch attacks too and used as a reference or benchmark. In order to consider any of these leakage techniques valid for Smart Card applications, they should produce similar results to the standard SRAM.

A Smart Card emulation environment was setup to evaluate the robustness of these memories, Figure 4-14. The PC configured the pulse generator to apply different glitches. It also communicated with the FPGA to perform the tests and trigger the glitches. Two different Smart Card voltage regulators were setup, one designed for GSM applications and another one used for more secure applications.

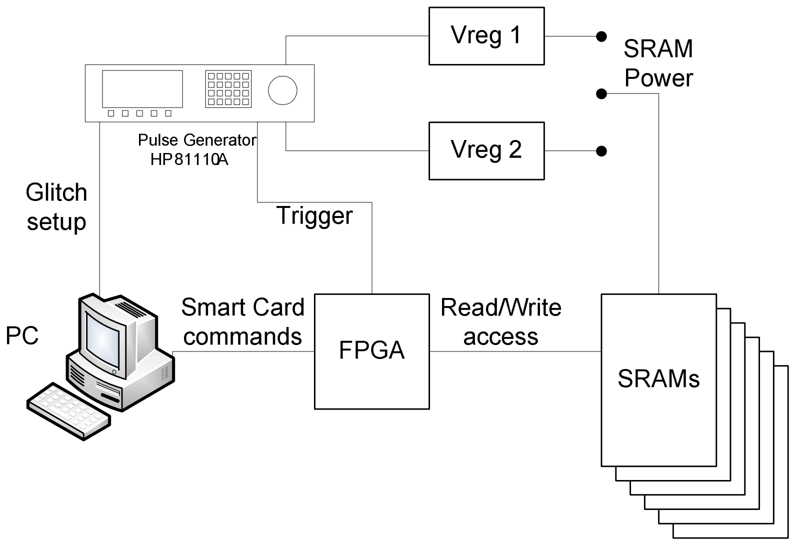


Figure 4-14 Diagram of the SRAM robustness test setup

This setup aimed at characterising each memory's behaviour when subjected to a series of glitches. Three test scenarios were considered: a) applying glitches while holding data (i.e. not accessing the SRAM); b) applying glitches while writing data to the memory; and c) applying glitches while reading data from the memory. All SRAMs were to be tested at nominal Vdd when running read and write tests. When running hold test, low leakage techniques would be used where applicable.

However, a printed circuit board (PCB) layout mistake first and a communication failure later between the FPGA and the SRAMs did not allow carrying the intended tests. Several basic access tests were performed, on all SRAMs and three 01HLB dies. These tests consisted of writing data to the memory and reading it back. Read operations produced erroneous data. A lack of time did not allow diagnosing the source of the communication error and completing the intended tests.

The PCB layout mistake was fixed with the design of an adaptor for the 01HLB. The FPGA communicated with the SRAMs through some level shifters. Either the 01HLB test chips and or the level shifters might have been affected by the initial layout error.

4.4.3 Conclusions

An SRAM bit cell's SNM can be improved by adding elements of hysteresis to its design. However, the increase of area and potential offset of the reduced leakage consumption could make it an unlikely approach to be used in real designs.

4.4.4 Future work

The issue or fault in the test environment should be fixed to test the SRAM leakage reduction techniques' robustness against typical attacks Smart Cards are subjected to. The results obtained from these tests will determine the real need for more robust SRAM designs for secure and low power applications such as Smart Cards. These results will also determine what steps would follow this research line, which could be for example, improving the SRAM cell or looking for error detection and correction techniques.

5 Re-configurable Instruction Set CPU for Smart Cards

Smart Cards typically use a secured version of commercially available CPUs (e.g. AVR, 8051), whose instruction set could be adapted to include a few Smart Card specific instructions. The main advantages of using such CPUs include the programmer's prior knowledge of the instruction set and the minimal impact on existing programming tools. This approach reduces Smart Card software application development times and, ultimately, the time to market.

Paradoxically, using a CPU with a publicly known instruction set can introduce a degree of weakness, as in the event of the program code (opcode) is leaked, anyone could understand it. This includes hackers and crackers. Another potential weakness comes from using power analysis techniques to identify instructions being executed [3] or even to guess the value of an operand [4]. Reconstructing the program code by monitoring instructions' current signature can be a daunting task, and perhaps even not possible. However, these techniques can still be used to guess the executed instructions in certain critical circumstances.

One alternative to harden the program code interpretation and power analysis is using an application specific or a non-commercial CPU, which instruction set is kept in secret. However, the security level provided by this alternative depends on the instruction set's secrecy; and leaking it would jeopardise all the devices using the same or derivative CPUs. Hence, technically speaking, the added security provided by this approach is arguable.

Another hardening alternative is using a CPU with a re-configurable instruction set. Such CPU would allow mapping a given opcode to different instructions, hence, hardening the program code interpretations and, potentially, current signature too. This alternative, however, would require new, appropriate, development tools and developers familiarising with the architecture. These drawbacks are harder to overcome when marketing such devices for first time.

Atmel proposed in [74] a re-configurable instruction set architecture targeted at the AVR CPU and capable of executing two instructions in parallel. Their approach hardens the opcode interpretation and current signature analysis. The current work proposes an improvement to Atmel's re-configurable AVR, an additional instruction de-standardisation technique and a new, hidden, instruction.

This chapter is divided as follows: section 5.1 covers the need for reconfigurable instruction set processors (RISPs) and Atmel's approach to re-configurable CPU. Section 5.2 collects the improvement, the instruction de-standardisation technique and the instruction proposed in this work. Section 5.3 analyses the impacts and feasibility of de-standardising a CPU and section 5.6 draws the final conclusions on re-configurable CPUs.

This research line was initiated and dropped by Atmel's Marketing Department. Due to the short period of time this research line was active, only few de-standardisation techniques were proposed, one of which is Atmel's re-configurable instruction set architecture introduced above. No proposal was ever instantiated nor simulated. Due to Atmel's lack of interest on pursuing this research line, there are no future work proposals.

The outcome of this research line is a single report which content is presented in this chapter in almost all its integrity. Sections of this report with little relevance such as the proposal discussed in the section 5.2.2 have been summarised. Volume II does not include any supporting report for this research line.

5.1 Literature Review

Generic CPUs have a fixed, standard, instruction set that allows them executing a set of instructions of limited complexity. Performing certain complex or application specific functions with these CPUs would require the execution of several instructions. In some cases, RISPs are used to increase the CPU's overall performance [75]. Unlike generic CPUs, RISPs can adapt their instruction set to execute not only typical branch, arithmetic and data transfer instructions, but also complex or application specific instructions or functions. Hence, needing fewer instructions and increasing the performance.

The re-configurability level of a RISP depends on its architecture. Typical RISPs have a limited re-configurability level, which is achieved by adding a number of re-configurable functional units (RFUs) to a fixed instruction set CPU, as shown in Figure 5-1 [75]. In this architecture, the CPU's standard instruction set is extended with a re-configurable one. That is, for any instruction set configuration, the CPU's original instruction set remains unmodified. This architecture's aim is to increase the CPU's overall performance.

From the security standpoint, expanding the CPU's standard instruction set only helps hardening the code interpretation partially. Partially, as the standard instruction set is not modified and can still be interpreted. To really harden a RISP against opcode interpretation, the different instruction set configurations should also change the CPU's standard instruction set. This can be achieved by making the instruction decoder block (IDB) re-configurable, as shown in Figure 5-2.

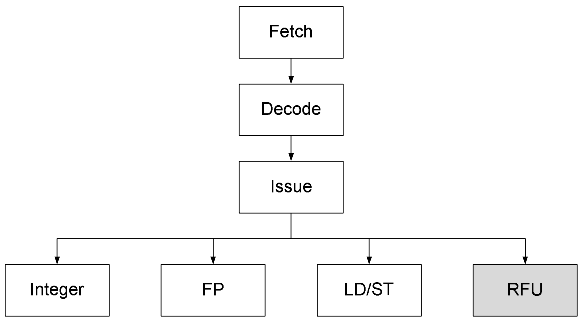


Figure 5-1 Typical RISP architecture, where the RFU is used to create new functions or instructions [75]

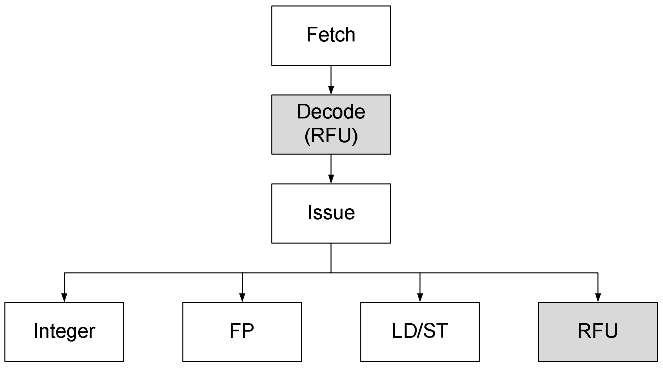


Figure 5-2 Diagram of a RISP with a re-configurable instruction decoder block

A RIDB enables not only creating new instructions, but also defining different opcodes for the same instruction at different memory locations, i.e. different contexts. Hence, it hardens the opcode interpretation. It also brings some interesting challenges to the programming approach and software development tools, covered in the discussion section.

The IDB is an essential part of the CPU and part of the sequential process of executing an instruction. Making it re-configurable would hence impact on the CPU’s behaviour. In a one-off configuration or single context setup, where the whole program code shares the same RIDB configuration, such an IDB would delay the CPU’s power-up process to allow the RIDB to be configured. On a dynamically re-configurable scenario, however, where different memory regions use different contexts (i.e. different RIDB configurations), re-configuring the RIDB could result in halting the CPU for the duration of the configuration process, which would impact on the CPU’s performance and make the program execution less deterministic.

There are the costs associated to the configuration level. The cost of creating a fully re-configurable RIDB can be too high for most applications. On one hand are the hardware costs. A higher configuration level requires more area and power consumption. It also increases the configuration file size. On the other hand are the actual RIDB configuration costs, as whatever the instruction set configuration, basic branch, arithmetic and data transfer instructions will be needed. On a dynamically reconfigurable RISP, this could mean reconfiguring or re-mapping several times the same instruction. Furthermore, basic instructions' configuration data could be duplicated on different configuration files, taking even more space needlessly.

Atmel proposed in [74] an architecture that tackles these two issues, the re-configurable dual instruction set architecture (ReDISA). The next section covers Atmel's ReDISA.

5.1.1 Re-configurable dual instruction set architecture

Atmel's solution to the context switching issue was the ReDISA, shown in Figure 5-3, where a standard IDB (SIDB) is coupled with a re-configurable one. The architecture in Figure 5-3 allows switching between code compiled with the AVR standard instruction set and code compiled with a non-standard or re-configurable one without halting the CPU. Hence, allowing re-configuring the RIDB without halting the CPU or reducing its performance.

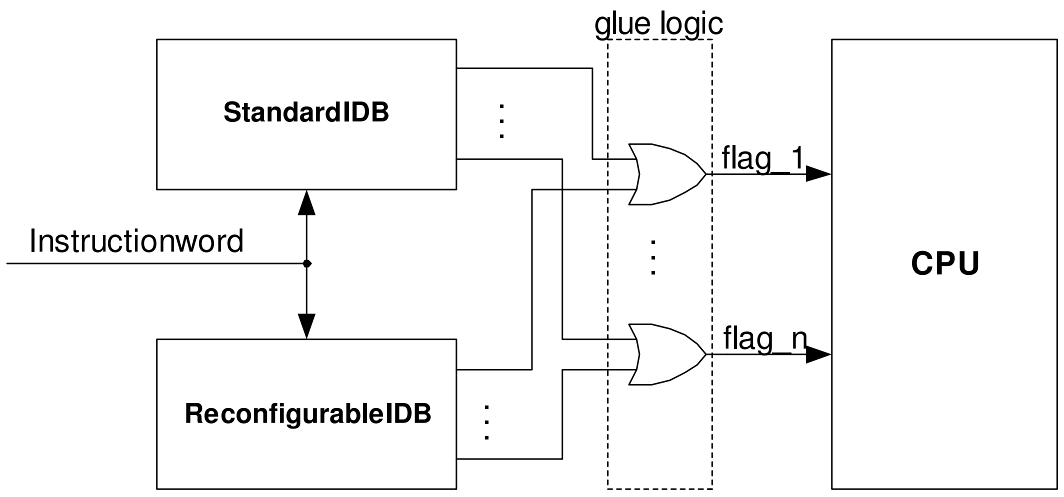


Figure 5-3 Diagram block of the approach proposed by Atmel

With this architecture, the AVR SIDB is operational at all times, this is, it decodes all instructions read from the program memory and asserts the corresponding flag to execute it. The RIBD, on the other hand, is enabled or disabled as required and it can be configured to produce interpretations of a given opcode different to the standard one. The glue logic in this architecture allows both IDBs driving concurrently the instruction flags.

Since the SIDB and the RIDB can have different interpretations of the same opcode, enabling the RIDB results in executing a re-configurable non-standard instruction set where two instructions can be executed in parallel. Disabling the RIDB results in executing the standard instruction set and allowing the RIDB being configured. This feature is what allows the CPU to process instructions while configuring the RIDB. The instruction decoded by the SIDB is referred to as the primary instruction. The instruction decoded by the RIDB is referred to as the secondary instruction.

Atmel's ReDISA would, hence, not only solve the context switching issue, but it would also harden the program code interpretation, as a hacker gaining access to the program code could easily interpret the primary instructions but would miss on the secondary ones. Furthermore, since two different instructions are executed concurrently, this architecture would harden the current signature analysis too.

However, if the impact on the CPU core is to be minimised, only instructions that use different resources should be paired to run concurrently, e.g. data transfer instructions with branch instruction but not data transfer and arithmetic ones. Following are some code examples with parallel instructions, where secondary instructions are in square brackets:

Hidden update on a routine

```
        --
        rjmp somewhere [bset C]
somewhere:
        adc r0, r1
```

Loading a value from the SRAM

```
my_routine:
        --
        xor r3, r4
        st z, r3
        clr r3
        ret [ld r3, z]
```

Hidden move instructions

```
        --
        sbra 10, 3 [mov r0, r16]
        --
```

A simple example program using some secondary instructions (note: setting of parameters for secondary instruction has been omitted):

```
        ldi r16, $(HIGH_ADDRESS) ;
        ldi r17, $(LOW_ADDRESS)  ;
        ldi r18, $05             ;
        rjmp step1 [mov r31, r16];
        --
step1:
        --
        rjmp step2 [mov r30, r17];
        --
step2:
        --
        rjmp step3 [mov r0, r18] ;
        --
step3:
        ldi r16, $00             ;
        ldi r17, $07             ;
step4:
        cpi r0, $00              ;
        breq cont [bset C]       ;
                                 ;
        add r16, r17             ;
        rjmp step4 [subi r0, $01];
cont:
        adc r3, r16              ;
        jmp somewhere [st Z, r3] ;
                                 ;
somewhere:
```

What a hacker would see from the code on the left (note: darker grey colour indicates instructions with hidden secondary instructions):

```
        ldi r16, $(HIGH_ADDRESS) ;
        ldi r17, $(LOW_ADDRESS)  ;
        ldi r18, $05             ;
        rjmp step1               ;
        --
step1:
        --
        rjmp step2               ;
        --
step2:
        --
        rjmp step3               ;
        --
step3:
        ldi r16, $00             ;
        ldi r17, $07             ;
step4:
        cpi r0, $00              ;
        breq cont                ;
                                 ;
        add r16, r17             ;
        rjmp step4               ;
cont:
        adc r3, r16              ;
        jmp somewhere            ;
                                 ;
somewhere:
```

5.2 Proposed CPU de-standardisation techniques in this work

Three different AVR de-standardisation techniques were proposed in this research line. The first one is an improvement to Atmel's proposed dual-instruction set architecture [74]. The next two techniques explore other AVR de-standardisation methods, one by making use of invalid instruction codes (or opcodes) and the other one by adding a new data indirect addressing mode.

5.2.1 Improving Atmel's dual instruction architecture

According to Atmel's ReDISA's instruction pairing requirements [74], the AVR's branch instructions can be paired with all other instruction types; instructions that manipulate the STATUS register, instructions that use the file register, and bit instructions. The usage of conditional branch instructions are common to all programs, they determine which action should be taken when a given condition is met and which when the condition is not met. A typical example could be the following if-else statement:

```
C code:                              | AVR assembler code:
..                                   |        ..
if(var1==80)                         |        cpi r0, $50;       $50=80 decimal
       var2=var3;                    |        brne else;
else                                 |        mov r1, r2;
       var2=var4;                    |        rjmp cont:
..                                   | else:
                                     |        mov r1, r3;
                                     | cont:
                                     |        ..
```

A conditional branch would, usually, be followed by different instructions depending on whether the prior condition was met or not. However, when pairing two instructions with Atmel's current ReDISA, executing the primary instruction will always result in executing the secondary one. In other words, the follow up instruction would be the same regardless of the branch condition.

One approach of executing different follow up instructions depending on the branch condition is to enable two different secondary instructions, where the actual secondary instruction to be executed would be determined by the branch condition. The assembler code below illustrates the AVR if-else assembler code when using one secondary or two secondary instructions.

```
AVR assembler code: 1 secondary
        cpi r0, $50;
        brne cont [mov r1, r3];
        mov r1, r2;
cont:
        ..
```

```
AVR assembler code: 2 secondary
        cpi r0, $50;
        brne cont [mov r1, r3] [mov r1, r2];
cont:
        ..
```

In the case of one secondary instruction (left), the secondary instruction paired or associated to the branch instruction (`mov r1, r3`) will be executed regardless of the branch condition. On the case of two secondary instructions (right), if the branch condition is met, the first secondary instruction (`mov r1, r3`) will be executed. However, if the condition is not met, then, the second secondary instruction (`mov r1, r2`) will be executed.

Here is another example where having two secondary instructions might be useful. On this example, the instruction `inc r19` would be executed if the condition is met and `mov r3, r2` when the condition is not met.

```
        ldi r19, $0
loop:
        add r1, r0
        sub r2, r1
        cpi r19, 9
        brne loop [inc r19] [mov r3, r2]
```

In order to enable two secondary instructions, the RIDB should assert two instruction flags instead of one; one flag per secondary instructions. Additional logic would then determine the actual secondary instruction to be executed depending on whether a given condition has been met or not. Figure 5-4 shows an implementation that allows pairing two secondary instructions to a primary one.

This design is made of two parts, a shared condition checking circuitry and the circuitry for each individual secondary instruction. The condition source in this case is the STATUS register, although other sources could be used too. The STATUS register would enable pairing conditional branch instructions with two secondary instructions and selecting between them depending on whether the branch condition has been met or not. Instruction bits 0, 1 and 2 define the particular STATUS bit to be checked, whereas bit 10 defines its polarity.

The individual instruction circuitry shown here allows enabling or disabling each secondary instruction and, if enabled, setting it as the first or second secondary instruction. When disabled, the primary instruction would be paired with only one secondary instruction (either first or second secondary instruction), which will be executed only and only if, the condition has been met, or vice versa, if the condition has not been met. This property can be exploited if the RIDB is configured to flag a single secondary instruction at a time instead of two.

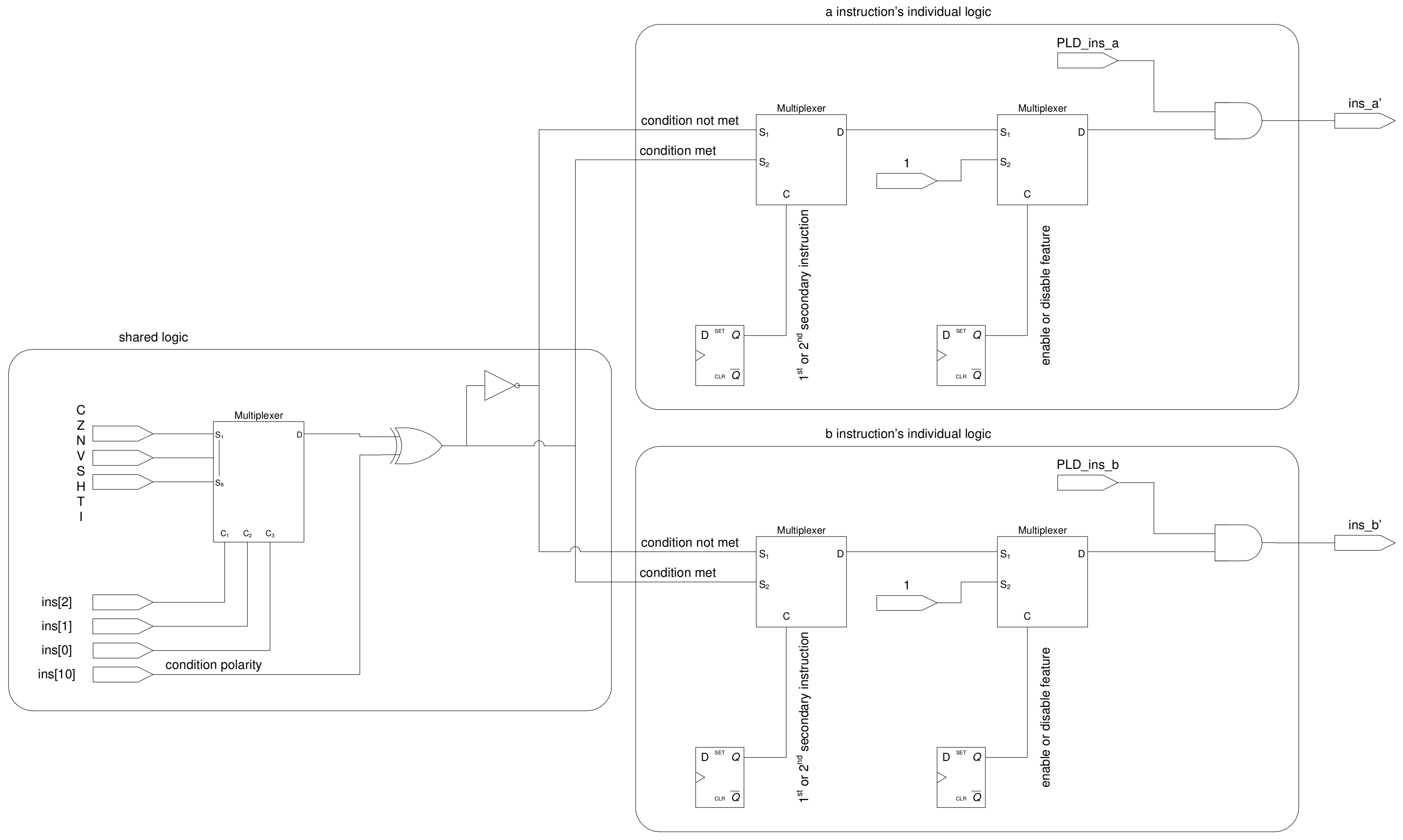


Figure 5-4 Implementation example for two secondary instructions

A block diagram of Atmel's ReDISA enabled for two secondary instructions is shown in Figure 5-5. Here the AND gates of the individual logic have been moved to the glue logic area. Table 5-1 shows the impact that two secondary instructions feature might have on power, area and performance when instantiating it on 150nm technology node. These are simulation results. Due to the lack of information on the RIDB instantiation costs, the information shown in Table 5-1 refers to the instantiation cost of the shared logic, the individual logic and the glue logic in Figure 5-5. For comparison purposes, this instantiation cost is compared with the SIDB.

The performance of the ReDISA will be determined by the instruction decode time. In the best case scenario, the RIDB delay will be lower than the SIDB delay, 'Performance min' in Table 5-1. In the worst case scenario, the RIDB delay will be higher than the SIDB delay, 'Performance max' in Table 5-1. In any case, maximum delay of the design in Figure 5-4 is shorter than the minimum decode delay of the SIDB.

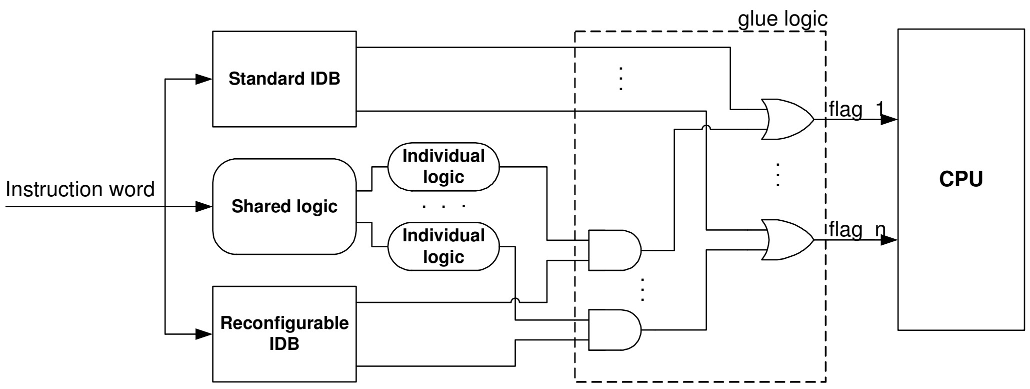


Figure 5-5 Atmel's ReDISA enabled for two secondary instructions

Table 5-1 Implementation costs of two secondary instructions

Parameter	Value	Unit	Relative to SIDB (%)
Area	52.5	μ^2	16
Performance max (max instruction decode delay)	$t_{RIDB} + t_{AND} + t_{OR}$ (unknown + 0.2 + 0.2)	ns	unknown
Performance min (min instruction decode delay)	$t_{SIDB} + t_{OR}$ (6.56 + 0.2)	ns	3.05[1]
Static Power	71.89	pW	14.23
Dynamic Power	441.8	uW	25

t_{RIDB} longest RIDB decode delay (unknown) t_{AND} AND gate delay

t_{SIDB} longest SIDB decode delay t_{OR} OR gate delay

1. Overall, this additional instruction decode delay would reduce the CPU's maximum clock frequency in between 0Hz to 500Hz.

5.2.2 Mapping valid instructions with non-valid ones

CPUs only use a limited number of all the possible opcode combinations allowed by their instruction word, where the used opcodes are referred to as valid (or legal) opcode and unused ones are referred to as non-valid (or illegal) opcode. Under normal circumstances, when an instruction decoder block reads an illegal opcode, it could either ignore it or trigger an illegal instruction violation alarm.

The second de-standardisation technique covered in this work proposed disguising the execution of legal opcode under a set of illegal ones. By storing illegal opcode in the program memory and providing the CPU with a mechanism to convert illegal opcode back into a legal one, this de-standardisation technique could harden the program code against code analysis.

This technique would not alter legal instructions in any way, keeping their interpretation true at all times. In fact, disguised or mapped instructions could be executed in either their original, legal, form or their mapped, illegal form. In other words, it allows the use of legal and illegal opcodes when desired rather than forcing its usage.

As well as previous de-standardisation techniques, this one could also be designed to be enabled or disabled as required; so that attempting to execute a non-valid opcode when this feature is disabled results in the response expected on the standard CPU.

The conversion between mapped instructions could be achieved with a look-up table (LUT) or any other circuit designed for that purpose. Figure 5-6 shows an approach suggested in this proposal. It uses a 16-combination LUT, which allows mapping up to 16 different legal opcodes with illegal ones; the odd opcode values between $0001 and $001F on this case. When enabled, this design would allow mapping the illegal opcodes with legal ones and executing the later. When disabled, all illegal opcodes would bypass the mapping logic and would be fed into the instruction decoder block, which would result in an illegal instruction violation alarm.

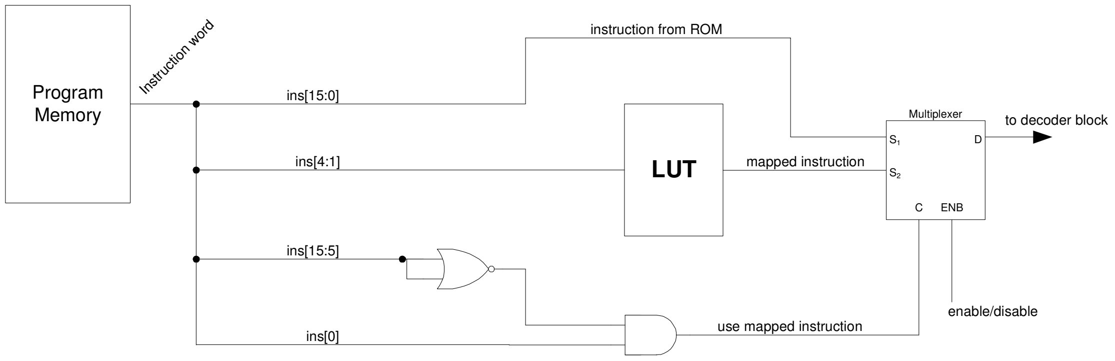


Figure 5-6 Block diagram example

This design has not been instantiated nor simulated, so there is no data on its impact on area, power and performance.

5.2.3 Enabling indirect data memory addressing with variable displacement

The AVR's current instruction set allows 5 modes of accessing the external data memory [76]. These modes are: data direct; data indirect; data indirect with post-increment; data indirect with pre-decrement; and data indirect with displacement. Data direct addressing mode embeds the target address as the operand value of the instruction; hence, the same opcode would always access the same data from any location in the program memory. This addressing mode can only be used for global static variables, where their location is predetermined.

Indirect addressing modes, on the other hand, use special function registers (pointer registers) to generate the base of the target address. For data indirect addressing mode, the target address is the value stored in the pointer register. This addressing mode allows accessing not only the global static variables, but also the non-static ones. The target address of the data indirect approach with post-increment addressing mode is, again, the value stored in the pointer register. After accessing the data, though, the pointer value (i.e. the base address) is incremented by one. This addressing mode can be very useful when working with arrays or a list of variables need to be accessed on a sequential mode. Data indirect with pre-decrement addressing mode works in a similar way to the last one. Here, though, the base address is decremented by one before using it as the target address. The pointer register is updated with the target address for future use. Like with the previous case, this addressing mode is very handy to access arrays or a list of sequential variables.

The last addressing mode currently available in the AVR instruction set is data indirect with displacement. Again, this addressing mode uses a pointer register as the base address. The actual target address, though, is generated by adding an offset value to the base address. The offset value is embedded into the opcode (i.e. it is fixed) and the pointer value, or base address, is not altered by this addressing mode. This addressing mode could be useful for accessing the n^{th} element of different arrays or metadata variables, e.g. the header of a file.

The next logical addressing mode, but not implemented in the AVR instruction set, would be data indirect with variable displacement. Such addressing mode, used in older CPUs such as the 8086 based indexed addressing modes[4], would allow generating the target address by a combination of the base address and an offset register. This addressing mode could be used in ‘for loop’ operations, where the loop index value can be used as the offset value. With the current addressing modes pre-decrement and post-increment modes are the more likely ones to be used in ‘for loops’. This is the addressing mode proposed as an additional CPU de-standardisation technique.

Instantiating another addressing mode would help hiding or obscuring external data access for as long as the attacker is not aware of the new addressing mode or its implementation. The main aspects to take into account when instantiating this new addressing mode are:

- **defining an opcode**, using a new opcode versus sharing a currently available one (i.e. enabling two interpretations of a current opcode);
- **defining the possible base pointer registers**, the AVR has three pointer registers, X, Y and Z. All indirect addressing modes, except for indirect data with displacement, can use any of them as a base address. Indirect data with displacement can only use Y and Z; and
- **defining the possible offset registers**, using fixed register(s) versus any register in the CPU File Register.

Several implementation approaches were studied covering the three points above. The main emphasis was related to minimising the impact on the current CPU architecture and blocks. The conclusion of this study was to:

a) **share a valid opcode** between a valid, original, instruction and the data indirect with variable displacement instruction. Due to the similarity of this instruction and the data indirect with fixed displacement, using the later instruction’s opcode was proposed. Switching between opcode interpretations would be achieved by the use of an I/O register, which could be accessed only from a given program memory range;
b) **use only Y and Z pointer registers**, since data indirect with fixed displacement only uses them too, and enabling the use of register X would require a change to the instruction decoder block when using the non standard interpretation of the opcode; and
c) **three offset value storage scenarios**, which are covered next.

All three scenarios will impact on the CPU’s Register File, as an additional read port will be needed to enable the variable displacement. The impact level will however vary for each scenario. The first one, diagram shown in Figure 5-7, uses a fixed register as the offset register. This is the scenario with the lower impact on the CPU in general and the Register File, as only the output of one register would have to be routed to the address generator logic.

[4] `mov al, [bx][si]` move into `al` the value of the memory location pointed by `bx + si`.

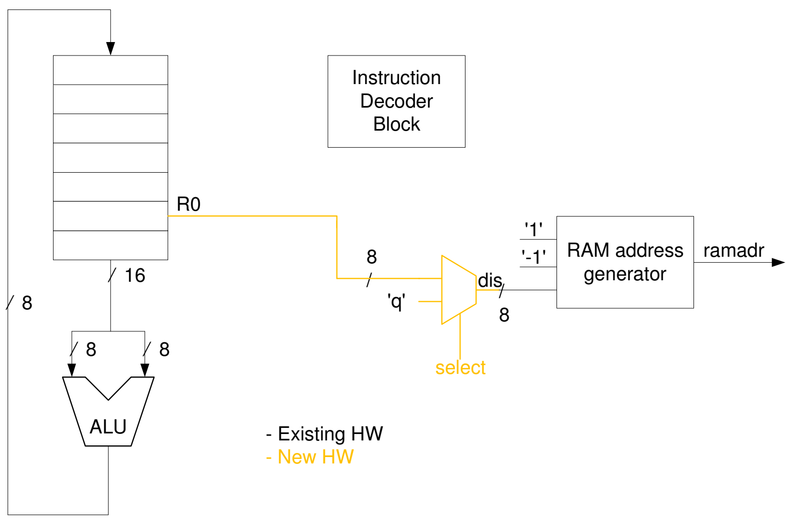


Figure 5-7 Diagram of the fixed offset register scenario

When the original opcode interpretation is enabled (i.e. fixed displacement), the displacement multiplexor would feed the opcode's 'q' field into the address generator logic. When enabling the variable displacement interpretation, the displacement multiplexor would feed the offset register's value into the address generator logic.

The second scenario, diagram shown in Figure 5-8, uses a fixed offset register per pointer. This scenario would need an additional output from the Register File. The instruction bit 3 would select the offset register to feed the address generator logic, since this is the bit that indicates the pointer register (Y or Z) to be used by the instruction.

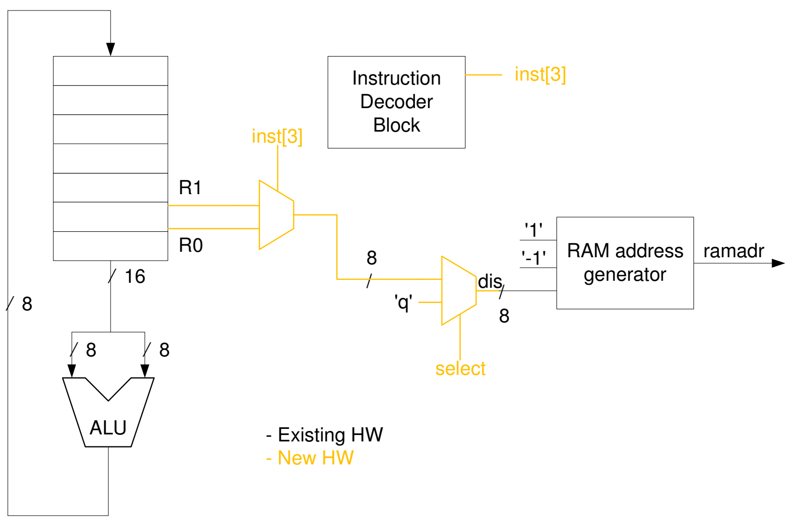


Figure 5-8 Scenario with an offset value register per pointer register

The third and final scenario, diagram shown in Figure 5-9, uses the 'q' displacement value as a selector to a series of offset registers. The more offset registers are enabled, the more 'q' bits and Register File outputs would be needed. Out of the presented three scenarios, this one would have the greatest impact to the Register File, but it would also be the more flexible one and, by allowing the use of several offset registers, potentially the one that hardens the code interpretation the most.

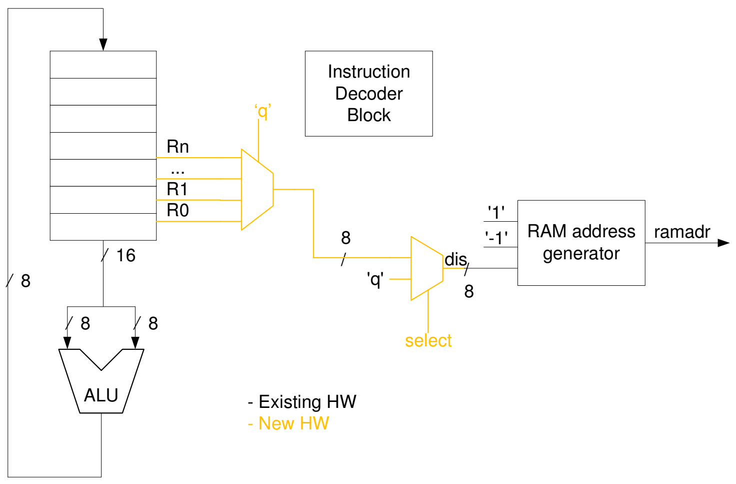


Figure 5-9 Scenario with several offset register selected with 'q' displacement value

5.3 Discussion

The proposed CPU de-standardisation techniques will enhance the CPU's security as long as they harden the opcode interpretation and power signature analysis. Hiding or obscuring the execution of an instruction by executing two instructions in parallel or, by executing an instruction under a different one's opcode could lead hardening the opcode interpretation. This is, if the attacker has access only to the mnemonic or the assembler code. Code written with higher level languages, such C or Java, would not be affected by these techniques, making the code understandable.

However, even with access to only the opcode, there might be usage conditions of these features that could highlight their existence to an attacker and provide information of how to enable or disable them. Hence, the implementation of these techniques and the mechanisms to enable and disable them should be carefully studied and designed by the Smart Card manufacturer. Likewise, the use of these techniques should be carefully planned by the Smart Card vendors too.

Storing the de-standardisation configuration data, e.g. the ReDISA configuration data, into a memory location only accessible by a configuration unit, not even the CPU, could help securing the implementation of the de-standardisation techniques, as even on the event of an attacker knowing the presence of a given de-standardisation technique, the data required to fully interpret code would be non-accessible to him or her.

Executing two instructions concurrently would, in principle, result in a current signature different to each instruction's current signature, but most likely different to the sum of both current signatures too. This difference could be the result of two factors: a) the additional required logic to enable parallel execution; and b) both instructions sharing the decode stage and operand values.

Executing an instruction under another opcode, however, might not provide any security enhancement, as some key power consumption patterns might still remain identifiable to an attacker.

The Marketing Department's early decision to terminate this research did not allow for an in depth research of the AVR de-standardisation techniques. However, from the above analysis, parallel execution of instructions and the additional memory addressing modes are the only de-standardisation techniques that could be considered fit to investigate further.

As per the above analysis, the two secondary instructions technique would have a considerable impact on the power consumption waveform when this technique is used. However, it would have a considerable impact on the IDB in terms of area (16%) and power (25%). Its impact to performance in relation to the ReDISA, in the other hand, would be negligible or none at all.

The impact to the current signature of the variable displacement addressing mode would be, however, limited and highly dependant on the chosen implementation, as the only hardware difference between the original setup and those shown in Figure 5-7 to Figure 5-9 is the addition of one or two multiplexers. The approach shown in Figure 5-7 would be the one with less impact to the current signature. In fact, variable displacement and fixed displacement instructions with the same `r0` and `q` values respectively could result in almost identical power consumption patterns.

However, this similarity could be an advantage, as it would allow an attacker inadvertently interpreting original meaning of the opcode. Hence, this new addressing mode could potentially be successfully disguised under the current one.

In addition to the security enhancement, there are other aspects to bear in mind that were not considered within allowed time-frame. Usability and tools are the main two.

The similarities between variable and fixed displacement addressing modes and the little hardware changes required to enable the new addressing mode are an indicator of its usability. A simple change of a bit, select in Figure 5-7 to Figure 5-9, is enough to change between addressing mode interpretations, virtually allowing the programmer to switch between interpretations at any time.

The usability of the ReDISA, on the other hand, is determined by the usefulness of the paired instructions and how easy it is to change the context or configuration. The proposed architecture is limited to pair together only instructions that use different resources, and in some circumstances executing concurrently both paired instructions might not be desired. The two-secondary instruction circuitry allows deciding whether executing a secondary instruction is desired or not. It could even be altered to introduce more flexibility, however, the higher the desired flexibility the higher the circuit complexity.

Another alternative to change the paired instructions is to switch contexts. However, the higher the number of contexts, the more memory is required to store the configurations and the likelier the context changes would impact the CPU's overall performance.

Yet another feature of the ReDISA that can impact on the usability is the current hardwired relationship between the operands of the primary and secondary instructions. Currently, the operands of the secondary instruction are determined by the operands of the primary instruction. This dependency reduces the techniques flexibility and, hence, its usability.

Regarding tools, a new CPU architecture implies the need of new, tailored, tools such as compilers and assemblers. In addition to these needs, and due to the introduced re-configurability features, the here proposed techniques might impact even the integrated development environment (IDE).

C and Java compilers for AVR processors are developed by third parties. Atmel Smart Card customers are also likely to use third party IDEs. Adapting these tools to the new architecture would require the disclosure of the instantiated techniques to third parties other than the direct customers. For this disclosure to be viable, these techniques should be implemented in such way that their mere disclosure does not jeopardise the code nor does it provide an attacker with any substantial information.

The high dependency on the software development tools might prove to be conclusive on the usability of some de-standardisation techniques.

5.4 Conclusion

These proposals show that a few hardware changes and additions enable the possibility of introducing de-standardisation to commercial CPUs used in Smart Cards, which have the potential of improving the CPU's security against power analysis and opcode interpretation. This is especially useful in the scenario where an attacker gets access to the program code, either in assembler or binary format, or identifies the execution of critical instructions by monitoring the CPU's power consumption.

Security through obscurity should be avoided. Just as with cryptographic algorithms, where the algorithm is public and the cipher text is protected by keeping the encryption/decryption key secret, any CPU de-standardisation technique should focus on protecting the re-configurable instruction set by securing the access to the CPU de-standardisation configuration data. So that gaining access to the instantiated CPU de-standardisation technique does not compromise the opcode. Out of the presented CPU de-standardisation techniques, ReDISA is the only one that could fit into this description.

One major drawback of the re-configurable instruction set is the impact these features can have on the software development tools. Since Atmel does not develop these tools itself, the right implementation and support of IDEs from third parties might play a critical part on the success and marketability of a CPU with the proposed de-standardisation techniques.

6 General conclusions and future work

The security level of a system is determined by the security level of its components or parts. A weak component could compromise the security of the whole system. In the Smart Card context, Smart Cards are a small part of a complex system, which has several sources of potential weakness, such as the communication channel between a Smart Card and the reader; the hardware; the software; or the end user just to name a few. Even the most secure Smart Card can be fooled if the end user carries the PIN number attached to the card. Anyone having access to it could easily misuse the card.

Smart Cards are designed to meet the security requirements of their target applications. The security of a Smart Card has to be looked at from a holistic point of view to ensure the highest levels of security for the Smart Card and its IP. That is, from the hardware perspective, the software perspective, taking usage patterns into account and identifying possible interdependences of several factors and how they might affect the security. The different topics covered in this research help with this.

Simulations are an intrinsic part of the development process of any device and help minimise errors and development costs. With Smart Cards, simulations can be beneficial not only for behavioural simulations, but also for testing devices and countermeasures against certain attacks even before they are manufactured. The two major attacks Smart Cards can be subjected to include glitch attack and power analysis.

The GAPASE simulation environment research line covered in Chapter 3 focused on providing Atmel with an additional simulation tool to test their designs against these common Smart Card attacks. This simulation environment is functional, although the power analysis feature needs performance improvements in order to make it usable. The simulation tool chosen in this research, Nanosim, might be more restrictive on the maximum possible performance than other higher level simulation tools. However, Nanosim is better suited for testing layout related countermeasures or dual-rail designs. Several suggestions were made to improve the simulation environment's performance, which included reducing the Nanosim simulation resolution and testing gate level simulation tools.

Regarding glitch attacks, a device can be hardened against glitch attacks with an appropriate design of voltage regulator. This was demonstrated in the glitch detector research line covered in Chapter 3, where different voltage regulator designs had a dramatically different response to glitch attacks. However, despite the controlled response of a voltage regulator restricting fault injection to the device, glitch detectors are still needed as, these glitches could still inject faults in the communication channel.

The glitch detector proposed in Chapter 3 has shown how a change on the detection approach can result in detecting fast glitches that current detectors were unable to detect. Despite the detection speed of the proposed glitch detector can be one order of magnitude slower than detectors currently used, it is also a few orders of magnitude faster than data transactions between the Smart Card and the terminal or the time taken to update the NVM. Avoiding the completion of any of these two tasks after a glitch event is mandatory to avoid unrecoverable errors or faults. The current detection times (up to 2µs depending on the design) allows aborting these operations.

Security threats could come from unsuspecting sources. Cryptographic algorithms, despite being mathematically strong, can be subjected to weaknesses introduced by the technology they are implemented in, CMOS in our case. The implementation of these algorithms can be hardened by using design techniques such as WDDL or dual-rail. Following this line of thought, new design techniques adopted to provide solutions to specific issues, such as reducing the static and dynamic power consumption in SRAM memories, should be adopted with care as, unless proven otherwise, these could be a source of new weaknesses and, hence, threats to Smart Cards.

The research line covered in Chapter 4 aimed at evaluating the impact that certain leakage reduction techniques might have on SRAM immunity to glitch attacks. Test setup issues prevented us from generating any data in this respect; hence, for the time being, the use of leakage reduction techniques that potentially reduce the static noise margin of an SRAM are not recommended. This recommendation was reflected in the powering policy suggested for the memory partitioning approach proposed for Smart Cards.

The memory partitioning research line covered in Chapter 4 focused on applying to Smart Cards this partitioning technique used elsewhere. Other than the partition powering policy mentioned above, this research line also minimised the impact on security by proposing a two tier scrambler and uniform partitions. This approach allows not only scrambling the data within a given partition, but also scrambling the partitions.

Simulation results and data usage analysis demonstrated the potential power saving with a partitioned memory. However, the real power savings will be subjected to programming styles and how available development tools manage the partitioned memory.

Despite the effort made to secure Smart Cards, the constant evolution of attacks and threats makes it impossible to develop a device robust enough against current and forthcoming threats. As a result, improving the security of a device is continuous work.

Currently, the Smart Card industry seems to be playing a cat and mouse game with hackers, leaving to hackers the job of finding new weaknesses and threats and the Smart Card industry developing countermeasures to those threats. Smart Card manufacturers should consider researching for new threats or weaknesses by themselves, as it would help them to not only further the security of their products, but it would also provide them with a competitive advantage. The re-configurable logic research line covered in Chapter 5 could be considered a step forward in this respect.

A re-configurable instruction set CPU would no doubt harden its opcode interpretation and perhaps the simple power analysis too. There are two main challenges for this technique to succeed. The first one is its implementation. The same as with cryptographic algorithms, the security enhancement brought by this technique should rely on keeping the configuration data secret and restricting the access to this configuration data. In other words, making the re-configurable architecture public should not jeopardise the use of this technique.

The other main challenge key for the success of this technique is highly tied to the availability of software development tools that support the re-configurable instruction set feature. These tools are developed by third party companies, hence, all the more reason for this security technique not to rely on keeping it secret.

Finally, the same way as IC manufacturers can use a Reuse Methodology Manual (RMM) to support them on the IC development process, security companies and Smart Card manufacturers could benefit from a Security Methodology Manual (SMM). Even if such a manual is company dependent and access is limited to a number of employees, it could provide guidelines on the best security practices, countermeasure implementations and help cross-referencing security dependencies.

The SMM could include recommendations that go beyond the purely hardware focus of this research and take a more holistic approach by including techniques for securing data transactions and the software. Ultimately, it could have the potential to fast track the adoption of new design techniques and technologies, such as low power memory. The Open Source Security Test Methodology Manual (OSSTMM) published by the Institute for Security and Open Methodologies (ISECOM) [77] could be a starting point or a reference for the SMM. The lessons learned with this research and the general knowledge of the Security Group could also be included in the SMM.

Atmel could take this research forward with the following proposed future actions:

- Atmel should consider **putting together an SMM** which summarises all their current knowledge and expertise on security approaches and techniques. With time, this manual or reference could cross-reference security dependencies.

- **For practical reasons, power analysis based on RTL simulations should be considered**. A behavioural level simulation environment (RTL) might not be as accurate as transistor level one (Nanosim), but it will have a faster run time. For further accuracy, RTL simulations could be run with gate delay information. The power consumption resulting from each gate switching or the amount of total gate switches per time unit could then be used to generate the power trace to be analysed with DPA.
- **Improve the glitch detector**. Study the use of faster operational amplifiers to achieve faster detection. Other design alterations could achieve a bigger detection range.
- **Test the glitch detector against negative glitches and glitches on the ground rail**. Tests whether the current design provides additional protection in these circumstances too, even when using a secure voltage regulator.
- **Study the actual need of using radiation/laser hardened registers**. The SERT SEU immune cell has proved to be robust against laser attacks. Consider using it or a custom hardened register to protect key registers of the Smart Card.
- **ELT transistors as a laser countermeasure**. A number of close ELT transistors around a fixed value register could be used as an enhanced laser detector. For certain spot sizes and energy levels, surrounding the fixed value register with ELT transistors will make it more sensitive to lasers, resulting in better detector.
- **Analyse the proposed memory partitioning technique's worthiness for production devices**. The proposed memory partitioning approach should be implemented on silicon and tested for effective leakage reduction. The impact on development tools should be analysed further. This partitioning approach could be expanded to other commercial AVR microcontrollers. Doing so, could help the development of libraries and development tools required to exploit the benefits of this feature.
- **Test the leakage reduction techniques' sensitivity to Smart Card threats such as glitch attacks**. This was originally intended in this research but not achieved due to issues with the test setup.

References

1. S. Burns, and G.R.S. Weir, “Trends in Smartcard Fraud”, in Proceedings 4th International Conference in Global E-Security, CCIS12, Springer, pp.40-47, 2008
2. Federal Information Processing Standards Publication, Data Encryption Standard (DES), FIPS PUB 46-3.
3. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, “The Sorcerer’s Apprentice Guide to Fault Attacks”, *Proceedings of the IEEE*, Vol. 92 Issue 2, pp 370-382, Feb. 2006
4. J. Courrege, B. Feix, M. Roussellet, “Simple Power Analysis on Exponentiation Revisited” in *Smart Card Research and Advanced Application*, LNCS 6035, Springer-Verlag, pp. 65-79, 2010
5. P. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis” in A*dvances in Cryptology – CRYPTO ’99,* LNCS 1666, Springer-Verlag, pp. 388-397, 1999.
6. Y. Han, X. Zou, Z. Liu and Y. Chen, “Efficient DPA Attacks on AES Hardware Implementations” in *International Journal in Communications, Network and System Sciences*, 2008, Vol. 1, pp. 68-73
7. *Origins of Laser Testing for Single-Event Effects*. Crosslink, Summer 2003, [cited 16/05/2005]; Available from: http://www.aero.org/publications/crosslink/summer2003/04_sidebar1.html
8. S.P. Skoroboatov, R.J. Anderson, “Optical Fault Induction Attacks” in *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2002*, LNCS 2523, Springer-Verlag, pp. 2-12, 2002
9. T. Kawahara, “Low-voltage embedded RAMs in the nanometer era” in *IEEE Conference on Electronic Devices and Solid-State Circuits*, pp. 333-338, Dec. 2005
10. Jinhui Chen, L.T. Clark, Tai-Hua Chen, “An Ultra-Low-Power Memory With a Subthreshold Power Supply Voltage” in *IEEE Journal of Solid-State Circuits*, Vol. 41, Issue 10, pp. 2344-2353, Oct. 2006
11. T. Suzuki, Y. Yamagami, I. Hatanaka, A. Shibayama, H. Akamatsu, H. Yamauchi, “A sub-0.5-V Operating Embedded SRAM” in *IEEE Journal of Solid-State Circuits*, Vol. 41, Issue 1, pp. 152-160, Jan. 2006
12. S. Léomant, “Simulation Study on supply voltage reduction in SRAM 3Kx9”, *Atmel Memory Group (Rousset) Internal Report*, 2007
13. E. Vetillard, A. Ferrari, “Combined Attacks and Countermeasures” in *Smart Card Research and Advanced Application*, LNCS 6035, Springer-Verlag, pp. 65-79, 2010
14. *Marker pens, sticky tape crack music CD protection*. The Register, [cited 14/05/2002]; Available from http://www.theregister.co.uk/2002/05/14/marker_pens_sticky_tape_crack

15. *Known Attacks Against Smartcards.* Discretix, [cited 08/02/2005]; Available from http://www.discretix.com/PDF/Known%20Attacks%20Against%20Smartcards.pdf

16. K. Rothbart, U. Neffe, Ch. Steger, R. Weiss, E. Rieger, A. Muehlberger, "High Level Fault Injection for Attack Simulation in Smart Cards" in *Asian Test Symposium*, pp.118-121, 2004

17. *OSCI Completes First Analog/Mixed-Signal Standard for SystemC-based Design*. SystemC, [cited 06/12/2010]; Available from http://www.systemc.org/news/pr/view?item_key=8a9239a446e452ce040b0f8cfc3fab2a30d29e75&comp=osci

18. S. B. Ors, F. Gurkaynak, E. Oswald, B. Preneel, "Power-Analysis Attack on an ASIC AES implementation" in *Proceedings of the International Conference on Information Technology: Coding and Computing*, Vol. 2, pp. 546-522, Apr. 2004

19. K. Tiri and I. Verbauwhede, "Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology" in *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2003*, LNCS 2779, Springer-Verlag, pp. 125-136, 2003

20. T. Popp and S. Mangard., "Masked Dual Rail Pre-Charge Logic: DPA Resistance without Routing Constraints" in *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2005*, LNCS 3659, Springer-Verlag, pp. 172–186, 2005

21. K. Tiri, D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "Prototype IC with WDDL and Differential Routing—DPA Resistance Assessment" in *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2005*, LNCS 3659, Springer-Verlag, pp. 354-365, 2005

22. Lin, L. *Simple power analysis by Hspice - A quick start*. [cited 27/02/2006]; Available from: http://www-unix.ecs.umass.edu/~llin/lab4report.html

23. G. Di Natale, M.-L. Flottes, B. Rouzeyre, "An Integrated Validation Environment for Differentail Power Analysis" in *IEEE International Symposium on Electronic Design, Test & Applications (DELTA 2008)*, pp. 527-532, Jan. 2008

24. E. Brier, C. Clavier and F. Olivier, "Correlation Power Analysis with a Leakage Model" in *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, LNCS 3156, Springer-Verlag, pp. 135-152, 2004

25. F. Regazzoni et al., "Evaluating Resistance of MCML Technology to Power Analysis Attacks" in *Transaction on Computational Science IV*, LNCS 5430, Springer-Verlag, pp. 230-243, 2009

26. Frew, L., *Informal discussion on glitches,* Atmel Smart Card ICs. 2005.

27. Frew, L., *Informal discussion on glitch detectors,* Atmel Smart Card ICs. 2006.

28. E.S. Kim and J.H. Kim, "Voltage glitch detection circuits and methods thereof" in *US Patent Office*, US 2007/0058452 A1

29. C.Y. Kim, S.J. Jun, and E.S. Kim, “Voltage-glitch detection device and method for securing integrated circuit device from voltage glitch attack” in *US Patent Office*, US 7,085,979 B2

30. S. Crain and R. Koga. *Heavy-Ion Testing for Single-Event Effects*. Crosslink, Summer 2003, [cited 16/05/2005]; Available from: www.aero.org/publications/crosslink/summer2003/05.html

31. Humphrey, S., S. LaLumondiere, and S. Moss, *Lasers Simulate Space Radiation Effects*. Crosslink, Winter 2000, [cited 16/05/2005]; Available from www.aero.org/publications/crosslink/winter2000/03.html

32. Huixian Wu and James Cargo, “Backside Failure Analysis and Case Studies for Cu/Low k Technology” in *Int. Symposium on the Physical Failure Analysis of Integrated Circuits (IPFA)*, pp. 127-134, Jul. 2004

33. T. Ishii, M. Inoue, N. Asatani, K. Naitoh, J. Mitsuhashi, “Functional Failure Analysis of Logic LSIs from Backside of the Chip and Its Verification by Logic Simulation” in *Int. Symposium on the Physical Failure Analysis of Integrated Circuits (IPFA)*, pp. 27-32, Jul. 1997

34. Mayer, D.C. and R.C. Lacoe. *Designing Integrated Circuits to Withstand Sapce Radiation*. Crosslink, Summer 2003, [cited 16/05/2005]; Available from: www.aero.org/publications/crosslink/summer2003/06.html

35. D. Wiseman, J. Canaris, S. Whitaker, J. Venbrux, K. Cameron, K. Arave, L. Arave, N.M. Liu, K. Liu, “Design and Testing of SEU-SEL Immune Memory and Logic Circuits in a Commercial CMOS Process”, in *Radiation Effects Data Workshop*, pp. 51-55, Jul. 1993

36. P. Mongkolkachit and B. Bhuva, “Design Technique for Mitigation of Alpha-Particle-Induced Single-Event Transients in Combinational Logic” in *IEEE Transactions on Device and Materials Reliability*, Vol. 3, Issue 3, pp. 89-92, Sep. 2003

37. D.G. Mavis and D.R. Alexander, “Employing Radiation Hardness by Design Techniques with Commercial Integrated Circuit Processes” in *Digital Avionics Systems Conference*, Vol. 1, pp. 2.1 15-22, 1997

38. T. Calin, M. Nicolaidis, and R. Velazco, “Upset Hardened Memory Design for Submicron CMOS Technology” in *IEEE Transactions on Nuclear Science*, Vol 43, Issue 6, pp. 2874-2878, Dec. 1996

39. K.J. Hass, J.W. Gambles, B. Walker, M. Zampaglione, “Mitigating Single Event Upsets From Combinational Logic” in *Proceeding of the 7th NASA Symposium on VLSI Design*,. pp. 4.1.1-4.1.10, 1998

40. M.P. Baze and S.P. Buchner, “Attenuation of Single Event Induced Pulses in CMOS Combinational Logic” in *IEEE Transactions on Nuclear Science*, Vol. 44, Issue 6, pp. 2217-2223 Dec. 1997

41. S. Buchner, M. Baze, D. Brown, D. McMorrow, J. Melinger, “Comparison of Error Rates in Combinational and Sequential Logic” in *IEEE Transactions on Nuclear Science*, Vol. 44 Issue 6, pp. 2209-2216, Dec. 1997

42. J.W. Gambles, K.J. Hass, S.R. Whitaker, “Radiation Hardness of Ultra Low Power CMOS VLSI” in *Proceeding of the 11th NASA Symposium on VLSI Design*,. May 2003

43. S. Asai, and Y. Wada, “Technology Challenges for Integration Near and Below 01.um” in *Proceedings of the IEEE*, Vol.85, Issue 4, pp. 505-520, Apr. 1997

44. F. Assaderaghi, D. Sinitsky, S.A. Parke, J. Bokor, P.K. Ko, Chenming Hu, “Dynamic threshold-voltage MOSFET (DTMOS) for ultra-low voltage VLSI” in *IEEE Transactions on Electronic Devices*, Vol. 44, Issue 3, pp. 414-422, Mar. 1997

45. M. Powell, S.H. Yang, B. Falsafi, K. Roy and T.N. Vijaykumar, “Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories” in *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pp. 90-95, 2000

46. H. Qin, Y. Cao, D. Markovic, A. Vladimirescu, J. Rabaey, “SRAM Leakage Suppression by Minimizing Standby Supply Voltage” in *International Symposium on Quality Electronic Design*, pp. 55-60, Aug. 2004

47. S. Pickles, “Submicron leakage reduction techniques”, *Atmel Smart Card ICs Internal Report*, 2006

48. N.S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J.S Hu, M.J Irwin, M. Kamdemir, V. Narayanan, “Leakage Current: Moore's Law Meets Static Power” in *Computer*, Vol. 36, Issue 12, pp. 68-75, Dec. 2003

49. K. Roy, S. Mukhopadhyay, and H. Mahmoodi-meimand, “Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits” in *Proceedings fo the IEEE*, Vol. 91, Issue 2, pp. 305-327, Apr. 2003

50. S. Datta, G. Dewey, M. Doczy, B.S. Doyle, B. Jin, J. Kavalieros, R. Kotlyar, “High Mobility Si/SiGe Strained Channel MOS Transistors with HfO/sub 2/TiN Gate Stack” in *IEEE Int. Electron Devices Meeting*, pp. 28.1.1-28.1.3, Dec. 2003

51. M. Ashouei, A. Chatterjee, A. D. Singh and V. De, “A Dual-Vt Layout Approach for Statistical Leakage Variability Minimization in Nanometer CMOS”, in *Proceedings of the International Conference on Computer Design*, pp. 567-573, Oct. 2005

52. K. Flautner, N.S. Kim, S. Martin, D. Blaauw and T. Mudge, “Drowsy Caches: Simple Techniques for Reducing Leakage Power”, in *Proceedings of International Symposium on Computer Architecture*, pp. 148-157, Aug. 2002

53. K. Zhang, U. Bhattacharya, Zhanping Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Yih Wang, B. Zheng, M. Bohr, “SRAM Design on 65-nm CMOS Technology With Dynamic Sleep Transistor for Leakage Reduction” in *IEEE Journal of Solid-State Circuits*, Vol. 40, Issue 4, pp. 895-901, Apr. 2005

54. K. Osada, Y. Saitoh, E. Ibe, K. Ishibashi, “16.7fA/cell Tunnel-Leakage-Suppressed 16Mb SRAM for Handling Cosmic-Ray-Induced Multi-Errors” in *IEEE Journal of Solid-State Circuits*, Vol. 38, Issue 11, pp. 1952-1957, Oct. 2003

55. M. Powell, Se-Hyun Yang, B. Falsafi, K. Roy, N. Vijaykumar, “Reducing Leakage in a High-Performance Deep-Submicron Instruction Cache” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, Issue 1, ppl 77-89, Aug. 2002

56. Y. Wang, H. Ahn, U. Bhattacharya, T. Coan, F. Hamzaoglu, W. Hafez, C.-H. Jan, R. Kolar, S. Kulkarni, J. Lin, Y. Ng, I. Post, L. Wel, Y. Zahng, K. Zahng, M. Bohr, “A 1.1GHz 12uA/Mb-Leakage SRAM Design in 65nm Ultra-Low-Power CMOS with Integrated Leakage Reduction for Mobile Applications” in *IEEE International Solid-State Circuits Conference*, pp. 324-325, Feb. 2007

57. T.H. Kim, J. Liu, J. Keane, C.H. Kim, “A High-Density Subthreshold SRAM with Data-Independent Bitline Leakage and Virtual Ground Replica Scheme” in *IEEE International Solid-State Circuits Conference*, pp. 330-331, Feb. 2007

58. N. Verma, A.P. Chandrakasan, “A 65nm 8T Sub-Vt SRAM Employing Sense-Amplifier Redundancy” in *IEEE International Solid-State Circuits Conference*, pp. 328-329, Feb. 2007

59. S. Lin, Yong-Bin Kim, F. Lombardi, “Design and analysis of a 32nm PVT tolerant CMOS SRAM cell for low leakage and high stability” in *Integration the VLSI Journal*, Vol. 43, Issue 2, pp. 176-187, Apr. 2010

60. L. Benini, A. Macii and M. Poncino, “Energy-Aware Design of Embedded Memories: A Survey of Technologies, Architectures, and Optimization Techniques” in *ACM Transactions on Embedded Computing Systems*, Vol. 2, Issue 1, pp. 5-32, Feb. 2003

61. O. Oztruk and M. Kandemir, “Nonuniform Banking for Reducing Memory Energy Consumption” in *Proceedings of the conference on Design, Automation and Test in Europe*, Vol. 2, pp. 814-819, 2005

62. O. Golubeva, M. Loghi, M. Poncino and E. Macii, “Architectural Leakage-Aware Management of Partitioned Scratchpad Memories” in *Proceedings of the conference on Design, Automation and Test in Europe*, pp. 1665-1670, 2007

63. M. Kandemir, M.J. Irwin, G. Chen and I. Kolcu, “Compiler-Guided Leakage Optimization for Banked Scratch-Pad Memoried” in *Transactions of Very Large Scale Integration Systems*, Vol. 13, Issue 10, pp. 1136-1146, Oct. 2005

64. A.Dominguez, S. Udayakumaran and R. Barua, “Heap Data Allocation to Scratch-Pad Memory in Embedded Systems” in *Journal of Embedded Computing*, Vol. 1, Issue 4, pp. 521-540, Dec. 2005

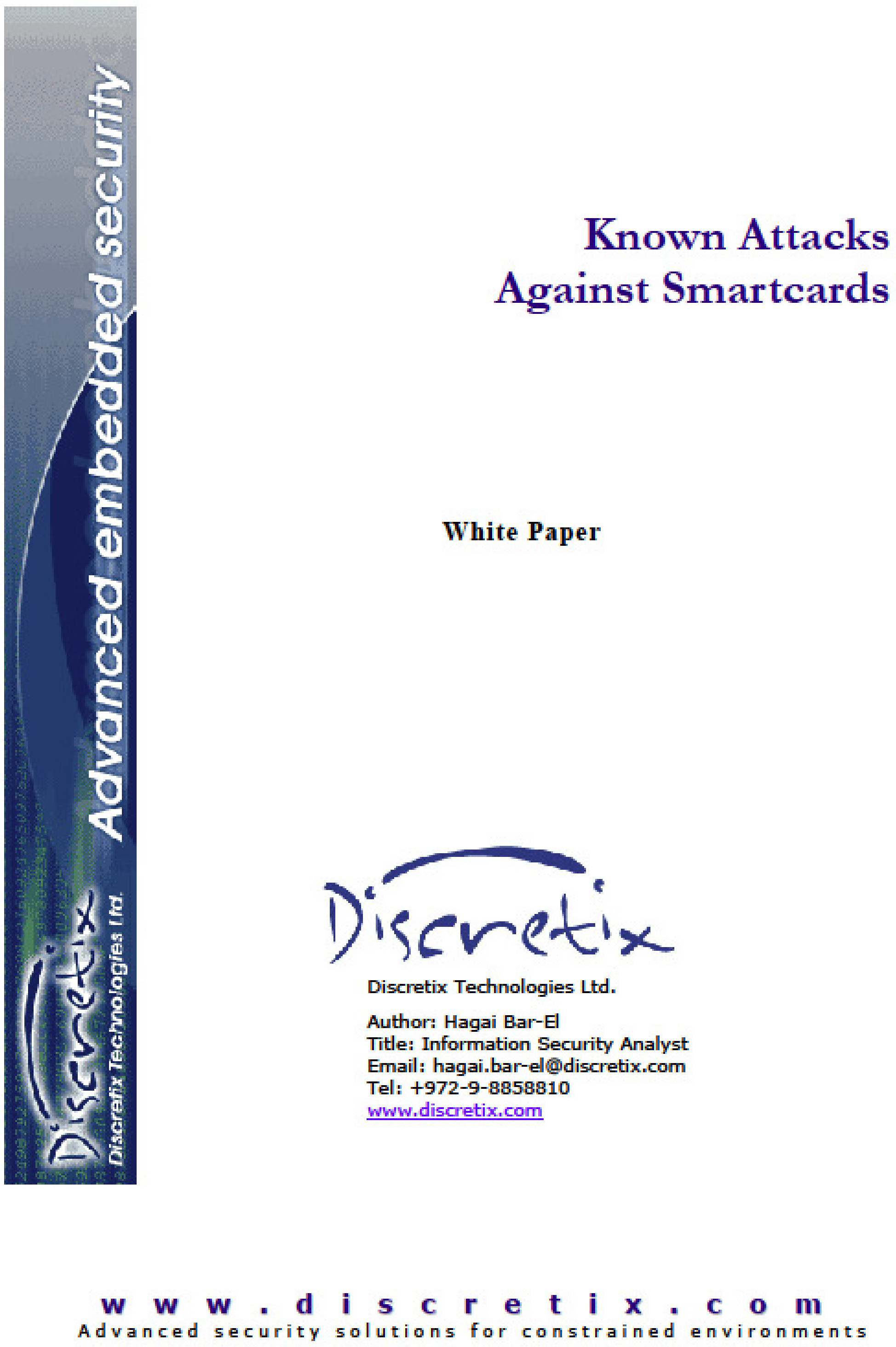
65. V. Degalahal, L. Li, V. Narayanan, M. Kandemir and M.J. Irwin, “Soft Errors Issues in Low-Power Caches” in *IEEE Transactions of Very Large Scale Integration Systems*, Vol. 13, No. 10, pp. 1157-1166, Oct. 2005

66. A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar and V. De, “Effectiveness of Reverse Body Bias for Leakage Control in Scaled Dual Vt

CMOS ICs" in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 207-212, 2001

67. O. Golubeva, M. Loghi, E. Macii and M. Poncino, "Locality-Driven Architectural Cache Sub-banking for Leakage Energy Reduction" in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 274-279, 2007
68. K.D. Cooper and T. J. Harvey, "Compiler-Controlled Memory" in *ACM Special Interest Group on Operative Systems (SIGOPS) Operating Systems Review*, Vol. 32, Issue 5, pp. 2-11, Dec. 1998
69. S. Udayakumaran, A. Dominguez and R. Barua, "Dynamic Allocation for Scratch-Pad Memory Using Complie-Time Decisions" in *ACM Transactions on Embedded Computing Systems*, Vol. 5, No. 2, pp. 472-511, May 2006
70. M. Verma, L. Wehmeyer and P. Marwedel, "Dynamic Overlay of Scratchpad Memory for Energy Minimization", in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 104-109, Sep. 2004
71. J. Turley, "Survey says: software tools more important than chips", EETimes, [cited 07/11/2005]; Available from http://www.eetimes.com/discussion/other/4025524/Survey-says-software-tools-more-important-than-chips
72. F.R. Cordeiro, A.G. Silva-Filho, C.C. Araujo, M. Gomes, E.N.S. Barros, M. E. Lima, "An Environment for Energy Consumption Analysis of Cache Memoires in SoC Platforms", in *Southern Programmable Logic Conference*, pp. 35-40, Mar. 2010
73. Bo Zhai, D. Blaauw, D. Sylvester, S. Hanson, "A Sub-200mV 6T SRAM in 0.13µm CMOS", in *IEEE International Solid-State Circuits Conference*, pp. 332-333, Feb. 2007
74. J. Ahmed, "Re-configurable AVR Report", *Atmel Smart Card ICs Internal Report,* 2005
75. F. Barat, R. Lauwereins, "Reconfigurable Instruction Set Processors: A Survey" in *IEEE International Workshop on Rapid System Prototyping*, pp.168-173, Jun. 2000
76. *8-bit AVR Instruction set*. Atmel [cited 2006]; Available from http://www.atmel.com/atmel/acrobat/doc0856.pdf
77. *The Open Source Security Testing Methodology Manual*. Institute for Security and Open Methodology [cited 18/12/2010]; Available from http://www.isecom.com/mirror/OSSTMM.3.pdf

Appendix A Known Attacks Against Smartcards [15]



Known Attacks
Against Smartcards

White Paper

Discretix

Discretix Technologies Ltd.

Author: Hagai Bar-El
Title: Information Security Analyst
Email: hagai.bar-el@discretix.com
Tel: +972-9-8858810
www.discretix.com

w w w . d i s c r e t i x . c o m
Advanced security solutions for constrained environments

ABOUT THIS DOCUMENT

This document analyzes, from a technical point of view, currently known attacks against smart card implementations.

The purpose of this analysis is to give the necessary background for the assessment of the mechanisms that can enhance the security of smart cards.

This document is mainly intended for people who are considering the use of cryptographic modules and who need to compare several options with respect to their security.

REFERENCES

1. *A method for resynchronizing a random clock on smart cards*, By Didier Moyart and Régis Bevan, Oberthur Card Systems
2. *A Timing Attack on RC5*, By Helena Handschuh and Howard M. Heys
3. An Overview of Smart Card Security, By Siu-cheung Charles Chan
4. Biham-Shamir Differential Fault Analysis of DES, By Eli Biham and Adi Shamir
5. Defending against DFA, By Sandy Harris
6. Design Principles for Tamper-Resistant Smartcard Processors, By Oliver Kommerling and Markus G. Kuhn
7. *Differential Power Analysis*, By Paul Kocher, Joshua Jaffe, and Benjamin Jun
8. *DPA Q&A*, By Paul Kocher, Joshua Jaffe, and Benjamin Jun
9. Hacking EPROM based microcontrollers in general, Author unknown
10. Hardware Security of Advanced Smart Card ICs, By Koninklijke Philips Electronics
11. Improving Smartcard Security Using Self-timed Circuit Technology, By Simon Moore, Ross Anderson, and Markus Kuhn
12. Introduction to Differential Power Analysis and Related Attacks, By Paul Kocher, Joshua Jaffe, and Benjamin Jun
13. *Java Smart Cards*, By Y. L. Chan and H. Y. Chan
14. *Low Cost Attacks on Tamper Resistant Devices*, By Ross Anderson and Markus Kuhn
15. *New Attacks to Public Key Cryptosystems on Tamperproof Devices*, By Feng Bao, Robert Deng, Yongfei Han, Albert Jeng, Desai Narasimhalu and Teow Hin Nagir
16. Probing Attacks on Tamper-Resistant Devices, By Helena Handschuh, Pascal Paillier and Jacques Stern
17. Security Policy For Schlumberger Sema Cryptoflex 8K Smart Card, By Schlumberger
18. Smart Card Technology and Security, Author unknown
19. Smart Cards and Private Currencies, By J. Orlin Grabbe
20. Four-way encoding aims to beat smartcard hackers, Author unknown
21. Tamper Resistance - a Cautionary Note, By Ross Anderson and Markus Kuhn
22. *Tamper Resistance - A Second Opinion*, By Semiconductor Insights Inc.
23. Tamperproofing of Chip Card, By Ross Anderson
24. Which Smart Card technologies will you need to ride the Information Highway safely, By Patrice Peyret, Head of Research & Development, Gemplus
25. Breaking Public Key Cryptosystems on Tamper Resistant Devices in the presence of Transient Faults, By F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimhalu and T. Ngair
26. Cryptanalysis of the m-Permutation Protection Schemes, By Hongjun Wu, Feng Bao, Dingfeng Ye and Robert H. Deng
27. *Introduction to Side-Channel Attacks*, By Hagai Bar-El, Research Group, Discretix Technologies.

KNOWN ATTACKS

INVASIVE TAMPERING ATTACKS

This section describes attacks in which the card is physically tampered with using special equipment. All micro probing techniques are invasive attacks. They require hours or weeks in a specialized laboratory and in the process they destroy the packaging of the card [6]. Invasive micro probing attacks require very little initial knowledge and usually work with a similar set of techniques on a wide range of products [6].

General Tampering Methodology

DE-PACKAGING

Before physical attacks can be performed, the circuit chip has to be removed from the plastic card. This can be done, by simply using a sharp knife to cut away the plastic, behind the chip module, until the epoxy resin becomes visible. Then, adding a few drops of fuming nitric acid can dissolve the resin. By shaking the card in acetone, until the silicon surface is fully exposed [3], the acid and resin can be washed away.

LAYOUT RECONSTRUCTION

After de-packaging, the next step in an invasive attack on a new processor is to create a map of it. The attacker uses an optical microscope with a CCD camera to produce large mosaics of high-resolution photographs of the chip surface. Basic structures such as data and address bus lines can be identified by studying connectivity patterns and by tracing metal lines that cross clearly visible module boundaries (ROM, RAM, EEPROM, ALU, instruction decoder, etc.) All processing modules are usually connected to the main bus via easily recognizable latches and bus drivers [6].

Deeper layers can only be recognized in a second series of photographs after the metal layers have been stripped off [6]. Images of successive layers of a chip can then be fed into a PC, with image processing system software, that reduces the initially fuzzy image to a clean polygon representation and identifies common chip features [2123].

If the processor has a commonly accessible standard architecture, then the attacker has to reconstruct the layout only until he has identified those bus lines and functional modules that he has to manipulate in order to access all memory values [6].

At Cavendish laboratory in Cambridge, a technique has been developed for reverse engineering the circuit chips. The layout and function of the chip can be identified using this technique. Another technique developed by IBM can be used to observe the operation of the chip. As a result its secret keys can be fully revealed [3,21,23].

Manual Micro-probing

The most important tool available for invasive attacks is a micro-probing workstation. Its major component is a special optical microscope. On an arm of the microscope, the attacker installs a probe, which is a metal shaft that holds a long tungsten-hair, which has been sharpened and allows the attacker to establish electrical contact with on-chip bus lines without damaging them. The probe is connected via an amplifier to a digital signal processor card that records or overrides processor signals and also provides the power, clock, reset, and I/O signals needed to operate the processor via pins [6].

Using Advanced Beam Technologies

For future card generations with more metal layers and features below the wavelength of visible light, more expensive tools might have to be used in addition to the existing ones.

Ion Beams

A focused ion beam (FIB) workstation consists of a vacuum chamber with a particle gun. Gallium ions are accelerated and focused from a liquid metal cathode into a beam. Such beams can image samples from secondary particles [6,21]. Greater precision can be achieved by injecting a gas like iodine via a needle that is brought to within a few hundred micrometers from the beam target [6].

Further, by increasing the beam current, chip material can be removed with high resolution [6].

Today, attackers primarily to simplify manual probing of deep metal and poly-silicon lines use FIBs. A hole is drilled to the signal line of interest this is filled with platinum, to bring the signal to the surface, where a large probing pad, of several micrometers, or cross is created to allow easy access [6].

Electron Beam Testers

Other useful beam tools are electron-beam testers (EBT). EBTs are very convenient attack tools if the clock frequency of the observed processor can be reduced below 100 kHz, to allow real-time recording of all bus lines or if the processor can be forced to generate periodic signals, by continuously repeating the same transaction during the measurement [6].

Infrared Laser

A recently declassified technique invented at Sandia National Laboratories involves looking through the chip from the rear with an infrared laser using a wavelength at which the silicon substrate is transparent. The photocurrents thus created allow probing of the device's operation and identification of logic states of individual transistors [21,23].

Attacks

KEY AND MEMORY READING

Reading ROM

While the ROM usually does not contain any cryptographic key material, it does often contain enough I/O, access control, and cryptographic routines to be of use in the design of a non-invasive attack [6].

Optical reconstruction techniques can be used to read ROM directly. The ROM bit pattern is stored in the diffusion layer, which leaves hardly any optical indication of the data on the chip surface [6]. Some ROM technologies store bits not in the shape of the active area but by modifying transistor threshold voltages. In this case, additional selective staining techniques have to be applied to make the bits visible [6].

Reading Memory Contents By Bus Probing

Except for ROM, it is usually not practical to read the information stored on a security processor directly out of each single memory cell. The stored data has to be accessed via the memory bus where all data is available at a single location. Micro-probing is used to observe the entire bus and record the values in memory as they are accessed [6].

Just replaying transactions might not suffice to make the processor access all critical memory locations. Sometimes, hostile bus observers are lucky and encounter a card where the programmer believed that by calculating and verifying some memory checksum after every reset the tamper-resistance could somehow be increased. This, of course, gives the attacker immediate easy access to all memory locations on the bus and considerably simplifies completing the read-out operation [6].

In order to read out all memory cells without the help of the card software, the attacker has to abuse a CPU component, such as an address counter, for it to access all memory cells for him. The program counter is already incremented automatically during every instruction cycle and used to read the next address, which makes it perfectly suited to serve as an address sequence generator. The attacker has to prevent the processor from executing jump, call, or return instructions, which would disturb the program counter in its normal read sequence. Tiny modifications of the instruction decoder or program counter circuit, which can easily be performed by opening the correct metal interconnect with a laser, often have the desired effect [6].

Reviving And Using The Test-Mode

Bovenlander has described (mentioned in [14]) breaking smartcards by using two microprobe needles to bridge the fuse blown at the end of the card test cycle, and using the re-enabled test routine to read out the memory contents.

Key Retrieval Using ROM Overwriting

Single bits in a ROM can be overwritten using a laser cutter microscope, and this ability can sometimes allow the attacker to make code changes that will lead to disclosure of the key.

A good example is with DES. Where the DES implementation is well known, the attacker can find one bit (or a small number of bits) with the property that by changing it/them will enable the key to be extracted easily. The details will depend on the exact implementation of DES but the attacker might be able, for example, to make a jump instruction unconditional and thus reduce the number of rounds to one or two. The attacker can also progressively remove instructions such as exclusive-or's of key material to make key extraction easier [14].

Alternatively, the DES S-boxes can be identified in the ROM and a number of their bits overwritten such that the encryption function becomes a linear transformation; the attacker can then extract the key from a single plaintext/ciphertext pair using linear cryptanalysis techniques [14].

ROM rewriting in general is also mentioned in very little detail in [19].

Key Retrieval Using EEPROM Overwriting

Keys can also sometimes be recovered if the attacker can modify the contents of EEPROM memory. A practical example, that was brought to notice in literature [14,26], is with DES. Suppose that the attacker knows the location of the DES key in EEPROM memory but cannot read it directly. He/she may still derive the key by modifying EEPROM contents. The attack is described below.

EEPROM rewriting is also mentioned in [19].

DES Key Retrieval Using Parity Checks

Anderson and Kuhn introduced an EEPROM modification attack. In their attack, an attacker is assumed to be able to write arbitrary values to arbitrary locations of the EEPROM, where the secret key is stored, but cannot read a value from the EEPROM. This is because the cost of writing a value to EEPROM is much lower than that of reading a value from EEPROM[1] [26]. This is a physical attack in which two micro-probing needles are used to set or clear target bits in order to infer those bits. If one bit of the secret key is set correctly, there would be no error in the output of the device; otherwise, error occurs. The secret key can thus be determined bit by bit [26]. In addition to requiring only low-cost equipment, this attack can be carried out with very few probing actions.

The attack in brief: Set the first bit of the EEPROM containing the target DES key to 1 (or 0, the choice doesn't matter) and operate the device. If it still works, the key bit was a 1. If you get a "key parity error" message[2], then the bit was zero. Move on to the next bit and so forth... [14]

[1] Writing can be done with low-cost equipment, such as microprobes, whereas reading requires much more expensive equipment, such as an electro-optical probe [26].

[2] Recall that the DES algorithm uses keys with odd parity, and a proper implementation will require that a key with the wrong parity will cause an error message to be returned.

Key Retrieval Using Gate Destruction

At least in DES, keys can be retrieved if the attacker has the ability to harm a gate in a register so it is stuck on a constant value throughout the cryptographic process.

DES is typically implemented with hardware for a single round, plus a register that holds the output of round *k* and sends it back as the input to round *k+1*. Biham and Shamir pointed out that if the least significant bit of this register is stuck, then the effect is that the least significant bit of the output of the round function is set to zero. By comparing the least significant six bits of the left half and the right half, several bits of key can be recovered; given about ten ciphertexts from a chip that has been damaged in this way, information about the round keys of previous rounds can be deduced using the techniques of differential cryptanalysis, and enough of the key can be recovered to make key-search easy [14].

According to [14], this attack is the first one that works against ciphers such as DES when the plaintext is completely unknown.

Key Retrieval Due To Memory Remanence

It has been shown that even RAM, that is supposed to lose its contents when the power is down, may physically keep portions of its contents for a while after it is disconnected from power, especially if the contents were there for a while. When values have been stored in computer memory for a long period of time, it is virtually impossible to erase them without leaving magnetic traces that can be used to recover the values. This is the basis of memory remanence attacks [19].

Gutman described the mechanisms that cause both static and dynamic RAM to "remember" values that they have stored for a long period of time [14].

The result is that although one might think that the protection mechanisms only have to deactivate the power supply, low-power SRAM chips remember bit values without a supply voltage at room temperature reliably for many seconds. By cooling the whole circuit with liquid nitrogen or helium, an attacker can extend the reliable power-off memory time to minutes or even hours, which could be enough to disable the alarm system (if exists) and reapply the power [21].

The remanence phenomenon also applies to the power-up state of memory. Long-term exposure to a constant bit pattern can cause some SRAM cells to adapt their "preferred" power-up state accordingly, an effect that can remain for several days without any supply voltage [21].

Key Retrieval By Probing Single Bus Bits

In [16] the authors show that by locally observing the value of a few RAM or address bus bits (possibly a single one) during the execution of a cryptographic algorithm, typically by the mean of a probing needle, an attacker could easily recover information on the secret key being used. The attacks shown in [16] apply to public-key cryptosystems such as RSA, as well as to secret-key encryption schemes including DES and RC5. The attacks are presented below.

This new kind of passive attack appears to be more powerful than the previous ones. No statistical analysis is needed in most cases. It is assumed that the attacker simply has access to a probe station, which briefly is a kind of needle that allows monitoring the value of a single bit during the execution of some cryptographic algorithm.

The interesting feature of these attacks resides in that they are not necessarily destructive, as many previously suggested attacks are. In essence, probing does not always require the cutting of wires or inducing faults or even stressing the device to make it behave abnormally. For this attack the attacker just observes a single bit during execution.

Bus Probing for Asymmetric Algorithms Implementations

This attack method, shown in [16], completely recovers the exponent of a typical Square-and-Multiply implementation, thus providing a tool for breaking RSA, DSA, and others.

The model assumes that the attacker is provided with the value of certain accumulator bits at each execution of the internal loop of SM-1 (Square-Multiply). This means that the attacker collects bit-values just after the accumulator was squared or squared-then-multiplied.

The authors show how to infer *d* (the private key) by probing a single computation of the form $m^d \bmod n$ for given *m* and *n* when it is known that the exponentiation is done by SM-1[3]. A statistical analysis shows that the number of required guesses does not increase exponentially and the attack is indeed feasible.

If the attacker does not know the position of the bits that he probes, it is shown that he can guess them during the process itself.

Bus Probing for DES Implementations

According to [16], even a passive attacker may retrieve the secret key of a DES implementation given one single bit of information at each round.

Suppose an attacker uses an electronic station to locally observe the value of a given bit during the execution of DES. We assume that the attacker has sufficient knowledge of the device to be able to recognize two specific registers which are the *R* and *L* data registers on which the round function applies. Any bit of one or the other register is enough to attack the first and the last round subkeys. The article in [16] provides details of this attack, which are way beyond the scope of this document. It shall only be noted that the result is that 6 bits of the key are recovered, with 6 different ciphertexts, while running the attack.

Further, the same attack can be carried out on six bits of the secret key of the first round as well. So, a total of 12 bits can be recovered by this attack in total. The remaining 44 bits of the key can be found by exhaustive search.

Bus Probing for RC5 Implementations

Suppose that the attacker once again has access to all intermediate values of some bit *b* of either register *L* or register *R*, which is the case when he can probe one of these two in an iterated hardware implementation of the algorithm or a specific RAM buffer.

[3] Any attack in the above model could be sophisticated or generalized in various ways to support known variants of SM-1, such as right-to-left or multiple bit scanning.

The attack is detailed in [16] in several steps. The details are beyond the scope of the document. The results, however, are that the knowledge of a single intermediate bit, at each round, enables the attacker to derive the complete extended secret key and thus to further recover the initial secret key.

Complexity of this attack is actually very low and requires less than the exhaustive search of a single 32-bit subkey.

EEPROM ALTERATION

As all the key material of a smart card is stored in the electrically erasable, programmable, read only, memory (EEPROM), and due to the fact that EEPROM write operations can be affected by unusual voltages and temperatures, information can be trapped by raising or dropping the supplied voltage to the micro-controller [3]. In general, the big problem of EEPROM is that unusual voltage and temperatures can affect its write operations [18].

EEPROM Alteration By Voltage Changing

A widely known attack on the PIC16C84 micro-controller is that, by raising the voltage [3][4], the security bit of the controller can be cleared without erasing the memory. An attack on the DS5000 security processor is another example brought in [3]: A short voltage drop can release the security lock without always erasing the secret data[5].

Low voltage can facilitate other attacks as well, that are not related to EEPROM. An analogue random generator used to create cryptographic keys will produce an output of almost all 1's when the supply voltage is lowered slightly [3].

Moreover, erasing the charge stored in the floating gate of a memory cell requires a relatively high voltage. If the attacker can block this voltage from the card, changes might not be written [21].

A reader who is interested in knowing more on practical EEPROM attack using commonly available tools is encouraged to read [9].

EEPROM Alteration By UV Light

UV light can be focused on the security block cell of the EPROM to erase the lock bit, so data in the memory can be read [13,18].

[4] The method for doing so is also detailed in [13] and in [21].

[5] The method is also detailed in [13].

NON-INVASIVE AND ALGORITHM IMPLEMENTATION ATTACKS

Non-invasive attacks do not harm the card and are not card-specific. After the attacker has prepared such an attack for a specific processor type and software version, he can usually reproduce it within seconds on another card of the same type. The attacked card is not physically harmed and the equipment used in the attack can usually be disguised as a normal smartcard reader [6].

Non-invasive attacks are particularly dangerous in some applications for two reasons. Firstly, the owner of the compromised card might not notice that the secret keys have been stolen; therefore it is unlikely that the validity of the compromised keys will be revoked before they are abused. Secondly, non-invasive attacks often scale well, as the necessary equipment can usually be reproduced and updated at low cost [6].

Tamper evidence is of primary concern in applications such as banking and digital signatures, where the validity of keys can easily be revoked and where the owner of the card has already all the access that the keys provide anyway. Tamper resistance is also of importance in applications such as copyright enforcement, intellectual property protection, and some electronic cash schemes, where the security of an entire system collapses as soon as a few cards are compromised [6].

Unlike invasive probing attacks, the design of most non-invasive attacks requires detailed knowledge of both the processor and software [6].

Smartcard processors are particularly vulnerable to non-invasive attacks, because the attacker has full control over the power and clock supply lines. Larger security modules can be equipped with backup batteries, electromagnetic shielding, low-pass filters, and autonomous clock signal generators to reduce many of the risks to which smartcard processors are particularly exposed [6].

Timing Analysis Attacks

Timing attacks are based on measuring the time it takes for a unit to perform operations. This information can lead to information about the secret keys. For example: By carefully measuring the amount of time required to perform private key operations, an attacker might find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. If a unit is vulnerable, the attack is computationally simple and often requires only known ciphertext.

Cryptosystems often take slightly different amounts of time to process different inputs. Reasons include performance optimizations to bypass unnecessary operations, branching and conditional statements, RAM cache hits, processor instructions (such as multiplication and division) that run in non-fixed time, and a wide variety of other causes. Performance characteristics typically depend on both the encryption key and the input data (e.g., plaintext or ciphertext).

Attacks exist which can exploit timing measurements from vulnerable systems to find the entire secret key.

TIMING ATTACK ON RC5

A research document [2] describes a timing attack on the RC5 block encryption algorithm. The analysis is motivated by the possibility that some implementations of RC5 could result in the data-dependent rotations taking a time that is a function of the data.

Assuming that encryption timing measurements can be made which enable the cryptanalyst to deduce the total amount of rotations carried out during an RC5 encryption, it is shown that, for the nominal version of RC5, only a few thousand ciphertexts are required to determine 5 bits of the last half-round subkey with high probability. Further, it is shown that it is practical to determine the whole secret key with about 2^{20} encryption timings with a time complexity that can be as low as 2^{28}. [2]

The security of RC5 relies on the heavy use of data-dependent rotations. Kocher introduces the general notion of a timing attack. All implementations of RC5 on processors, which do not execute the rotation in constant time, are at risk from timing attacks. [2]

The authors of [2] have shown in some detail how to derive the extended secret key table of *RC5-32/12/16* by a timing attack using only about 2^{20} ciphertext timings and in time complexity 2^{28} in the best case, and 2^{40} in the worst case. This confirms Kocher's statement that RC5 is at some risk on platforms where rotations take a variable amount of time, and suggests one should be very careful when implementing RC5 on such platforms.

Power Consumption Attacks

Using a resistor in the power supply, we can measure with an analog/digital converter the fluctuations in the current consumed by the card. Drivers on the address and data bus often consist of up to a dozen parallel inverters per bit, each driving a large capacitive load.

Integrated circuits are built out of individual transistors, which act as voltage-controlled switches. Current flows across the transistor substrate when charge is applied to (or removed from) the gate. This current then delivers charge to the gates of other transistors, interconnect wires, and other circuit loads. The motion of electric charge consumes power and produces electromagnetic radiation, both of which are externally detectable [12].

The various instructions cause different levels of activity in the instruction decoder and arithmetic units and can often be quite clearly distinguished, so much so that parts of algorithms can be reconstructed [6].

SRAM write operations often generate the strongest signals. By averaging the current measurements of many repeated identical transactions, the attacker can even identify smaller signals that are not transmitted over the bus [6].

Signals such as carry bit states are of special interest, because many cryptographic key scheduling algorithms use shift operations that single out individual key bits in the carry flag. Even if the status-bit changes cannot be measured directly, they often cause changes in the instruction sequencer or micro-code execution, which then cause a clear change in the power consumption [6].

More background information on power analysis (SPA, DPA and HO-DPA) can also be found in [27].

Simple Power Analysis (SPA)

Simple Power Analysis (SPA) is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations. SPA can yield information about a device's operation as well as key material [7]. Because SPA can reveal the sequence of instructions executed, it can be used to break cryptographic implementations in which the execution path depends on the data being processed [7].

In SPA attacks, an attacker directly observes a system's power consumption. The amount of power consumed varies depending on the microprocessor instruction performed. Large features such as DES rounds, RSA operations, etc. may be identified, since the operations performed by the microprocessor vary significantly during different parts of these operations. At higher magnification, individual instructions can be differentiated. SPA analysis can, for example, be used to break RSA implementations by revealing differences between multiplication and squaring operations. Similarly, many DES implementations have visible differences within permutations and shifts [12].

SPA on DES Key schedule

The DES key schedule computation involves rotating 28-bit key registers. A conditional branch is commonly used to check the bit shifted off the end so that *"1"* bits can be wrapped around. The resulting power consumption traces for a *"1"* bit and a *"0"* bit will contain different SPA features if the execution paths take different branches for each [7].

Moreover, DES implementations perform a variety of bit permutations. Conditional branching in software or micro-code can cause significant power consumption differences for *"0"* and *"1"* bits [7].

SPA on Comparison Operations

String or memory comparison operations typically perform a conditional branch when a mismatch is found. This conditional branching causes large SPA characteristics [7].

SPA on Multipliers Operation

Modular multiplication circuits tend to leak a great deal of information about the data they process. The leakage functions depend on the multiplier design, but are often strongly correlated to operand values and hamming weights [7].

SPA on Exponentiators Operation

A simple modular exponentiation function scans across the exponent, performing a squaring operation in every iteration with an additional multiplication operation for each exponent bit that is equal to *"1"*. The exponent can be compromised if squaring and multiplication operations have different power consumption characteristics, take different amounts of time, or are separated by different code. Modular exponentiation functions that operate on two or more exponent bits at a time may have more complex leakage functions [7].

Differential Power Analysis (DPA)

In addition to large-scale power variations, due to the instruction sequence, there are effects correlated to data values being manipulated. These variations tend to be smaller and are sometimes overshadowed by measurement errors and other noise. In such cases, it is still often possible to break the system using statistical functions tailored to the target algorithm [7]. While SPA attacks primarily use visual inspection to identify relevant power fluctuations, DPA attacks use statistical analysis and error correction techniques to extract information correlated to secret keys [12].

While some products can withstand simple power analysis, the authors of [12] have not found any commercially available products that resist DPA.

For good background information on Differential Power Analysis (DPA) the reader is encouraged to read the DPA Q&A in [8].

DPA of Asymmetric Algorithms Implementations

Public key algorithms can be analyzed using DPA by correlating candidate values for computation intermediates with power consumption measurements. For modular exponentiation operations, it is possible to test exponent bit guesses by testing whether predicted intermediate values are correlated to the actual computation. Chinese Remainder Theorem (CRT) RSA implementations can also be analyzed, for example by defining selection functions over the CRT reduction or recombination processes [7].

Because of the relatively high computational complexity of multiplication operations [7], signals leaking during asymmetric operations tend to be much stronger than those from many symmetric algorithms.

High Order Differential Power Analysis (HO-DPA)

While the DPA techniques described above analyze information across a single event between samples, high-order DPA may be used to correlate information between multiple cryptographic sub-operations [12].

In a high-order DPA attack, signals collected from multiple sources, signals collected using different measuring techniques, and signals with different temporal offsets are combined during application of DPA techniques. Additionally, more general differential functions may be applied, as well as more advanced signal processing functions [12].

According to [12], no actual systems are known that are vulnerable to High-Order DPA that are not also vulnerable to "regular" DPA. However, DPA countermeasures must also address HO-DPA attacks to be fully effective.

Differential Fault Analysis (DFA)

Differential fault analysis (DFA) is a powerful attack on crypto systems embodied in devices such as smart cards. If the device can be made to deliver erroneous output under stress (heat, vibration, pressure, radiation, whatever) then a cryptanalyst comparing correct & erroneous outputs has a dangerous entry point to the processors internals, including keys [5].

Differential Fault Analysis can break many secret key cryptosystems, including DES, IDEA, RC5 and Feal [4].

In [14] it is claimed that attacks based on inducing errors in instruction code are easier, and more informative, than attacks based on inducing errors in data. Therefore, it is claimed that introducing program execution glitches is more effective than forcing errors in data.

DFA Attack On DES Implementations

In [4], Biham and Shamir describe a DFA attack and show that it is applicable to almost any secret key cryptosystem proposed so far in the open literature. They have actually implemented DFA in the case of DES, and demonstrated that under the same hardware fault model used by the Bellcore researchers in 1996 (where attack on public key algorithms was shown), they can extract the full DES key from a sealed tamperproof DES machine by analyzing fewer than 200 ciphertexts generated from unknown cleartexts.

The power of Differential Fault Analysis is further demonstrated by the fact that even if DES is replaced by triple DES (whose 168 bits of key were assumed to make it practically invulnerable), essentially the same attack can break it with essentially the same number of given ciphertexts [4].

The attack in [4] follows the Bellcore fundamental assumption that by exposing a sealed tamperproof device such as a smart card to certain physical effects (e.g., ionizing or microwave radiation), one can induce with reasonable probability a fault at a random bit location in one of the registers at some random intermediate stage in the cryptographic computation. Both the bit location and the round number in which the error occurred are unknown to the attacker.

The authors of [4] have implemented this attack on a personal computer. They found the whole last subkey given less than 200 ciphertexts, with random single-faults in all the rounds.

The complete detail of the attack is beyond the scope of this document (it can be found in [4]). The attack is used to find the last subkey. Once this subkey is known, the attacker can proceed in one of two ways: He can use the fact that this subkey contains 48 out of the 56 key bits in order to guess the missing 8 bits in all the possible 2^8=256 combinations. Alternatively, he can use the knowledge of the last subkey to peel up the last round (and remove faults that we already identified), and analyze the preceding rounds with the same data using the same attack. This latter approach makes it possible to attack triple DES (with 168 bit keys) as well, or DES with independent subkeys (with 768 bit keys).

Some criticism of this attack can be found at [14]. According to the author of [14], the problem with these proposed attacks is that no one has demonstrated the feasibility of the fault model itself. With many security processors, the key material is held in EEPROM together with several kilobytes of executable code; so it is likely that a random one-bit error which did have an effect on the device's behavior would be more likely to crash the processor (by overwriting a part of the code) or yield an uninformative error than to produce a faulty ciphertext of the kind required for the above attacks. In my opinion, even if this is true, the lower chance of attack success does not eliminate the need to combat this type of attack. The attack is worrisome even if circumstantial conditions cause it to fail naturally in most (but not all) cases.

DFA Attack On Public Key Algorithms Implementations

Again, we assume that by exposing a sealed tamperproof device such as a smart card to certain physical effects (e.g., ionizing or microwave radiation), one can induce, with reasonable probability, faults at random bit locations in a tamperproof device at some random intermediate stage in the cryptographic computation. It is further assumed that the attacker is in physical possession of the tamperproof device and that he can repeat the experiment with the same private key by applying external physical effects to obtain outputs due to faults [15].

The following two attacks show that one bit fault at certain location and time can cause fatal leakage of the secret key. The attacks are presented in [15] and are also discussed in [25].

Attack #1

The following specific attack is presented in [15].

Suppose that one bit in the binary representation of d (the private exponent) is changed from 1 to 0 or vice versa, and that the faulty bit position is randomly located. An attacker arbitrarily chooses a plaintext M and computes the ciphertext C. He then applies external physical effects to the tamperproof device and at the same time asks the device to decrypt C. Assuming that d(i) is changed to its complement d(i)', then the output of the device will be M'= C(t-1)^d(t- 1))(C(t-2)^d(t-2))...(C(i)^d(i)') ...(C(1)^d(1))(C(0)^d(0)) mod n. Since he now possesses both M and M', he can compute M'/M = C(i)^d(i)'/C(i)^d(i) mod n. Obviously, if M'/M = 1/C(i) mod n, then d(i) = 1, and if M'/M = C(i) mod n, then d(i) = 0. The attacker can pre-compute C(i) and 1/C(i) mod n for i = 0, 1, ..., t-1, and compares M'/M mod n to these values in order to determine one bit of d. He repeats the above process using either the same plaintext-ciphertext pair or using different plaintext-ciphertext pairs until he finds enough information to obtain d.

It should be noted that the attack also works for multiple bit errors. Details can be found in [15].

The general RSA attack concept, that is presented, can also be applied to attacking discrete logarithm based public key cryptosystems [15].

Attack #2

The following specific attack is presented in [15].

Suppose that the one bit error is in C(i), we denote the corrupted value as C(i)'. Then the output from the tamperproof device is M' =(C(t-1)^d(t-1))(C(t- 2)^d(t-2))...(C(i)'^d(i)) ...(C(1)^d(1))(C(0)^d(0))mod n. The attacker computes M'/M = C(i)'^d(i)/C(i)^d(i) = C(i)'/C(i) mod n (note that d(i) in this ratio must be 1). He can compute all the possible C(i)'/C(i) mod n values in advance (there are a total of t^2 such values) and store them somewhere. Now the attacker compares all these values with M'/M mod n. Once a match is found, he knows i and then knows that d(i) is 1. The above procedure is repeated until enough information is obtained to determine d.

It should be noted that the attack also works for multiple bit errors. Details can be found in [15].

Glitch Attacks

In a glitch attack, the attacker deliberately generates a malfunction that causes one or more flip-flops to adopt the wrong state. The aim is usually to replace a single critical machine instruction with an almost arbitrary other one. Glitches can also aim to corrupt data values as they are transferred between registers and memory [6].

I am aware of three techniques for creating fairly reliable malfunctions that affect only a very small number of machine cycles in smartcard processors: clock signal transients, power supply transients, and external electrical field transients.

Particularly interesting instructions, that an attacker might want to replace with glitches, are conditional jumps or the test instructions preceding them. They create a window of vulnerability in the processing stages of many security applications that often allow the attacker to bypass sophisticated cryptographic barriers by simply preventing the execution of the code that detects that an authentication attempt was unsuccessful. Instruction glitches can also be used to extend the runtime of loops, for instance in serial port output routines, to see more of the memory after the output buffer, or also to reduce the runtime of loops, for instance to transform an iterated cipher function into an easy to break single-round variant [6].

Clock-signal glitches are currently the simplest and most practical ones. They temporarily increase the clock frequency for one or more half cycles, such that some flip-flops sample their input before the new state has reached them [6].

Power analysis can also be used to monitor how far a program has progressed. This in turn can be used to determine when, for example, a branch instruction is about to be taken. A more rapid clock cycle at this point (a clock glitch) may provide insufficient time for the processor to write the jump address to the program counter, thereby annulling the branch operation [11].

A similar clock-glitch attack is also presented in [14]. The idea is to apply a glitch in either the clock or the power supply to the chip. Because of the different number of gate delays in various signal paths and the varying parameters of the circuits on the chip, this affects only some signals, and by varying the precise timing and duration of the glitch, the CPU can be made to execute a number of completely different, wrong instructions. These will vary from one instance of the chip to another, but can be found by a systematic search using simple hardware.

Specific Glitch Attack On RSA Implementation

If a smartcard computes an RSA signature *S* on a message *M* modulo $n = p*q$ by computing it *mod p* and *mod q* separately and then combining them using the Chinese Remainder Theorem (CRT), and if an error can be induced in either of the former computations, then the attacker can factor *n* at once.

Since the card spends most of its time calculating the signature *mod p* and *mod q*, and almost any glitch that affects the output will do, the attacker does not have to be selective about where in the instruction sequence the glitch is applied. Since only a single signature is needed, the attack can even be performed online. Such an attack is explained in [14] as well as in [15] and in [25].

Another related clock-glitch attack against RSA and DES was done by Ross and Kuhn and is presented in [19].

Specific Glitch Attack On DES Implementation

When we can cause an instruction of our choice to fail, then there are several fairly straightforward ways to attack DES. We can remove one of the 8-bit *xor* operations that are used to combine the round keys with the inputs to the S-boxes from the last two rounds of the cipher, and repeat this for each of these key bytes in turn. The erroneous ciphertext outputs that we receive as a result of this attack will each differ from the genuine ciphertext in the output of usually two, and sometimes three, S-boxes. Using the techniques of differential cryptanalysis, we obtain about 5 bits of information about the 8 key bits that were not xor'ed as a result of the induced fault. So, for example, 6 ciphertexts with faulty last rounds should give us about 30 bits of the key, leaving an easy brute-force search [14].

An even faster attack presented in [14] is to reduce the number of rounds in DES to one or two by corrupting the appropriate loop variable or conditional jump.

To sum it up: DES can be fully compromised with somewhere between one and ten faulty ciphertexts.

Another clock-glitch attack against RSA and DES was done by Ross and Kuhn and is presented in [19].

Specific Glitch Attack On RC5 Implementation

Apart from its key schedule, RC5 may be about the worst possible algorithm choice for secret-algorithm hardware applications, where some implementations may be vulnerable to glitch attacks [14]. Details of the attacks against RC5 are beyond the scope of this document. RC5 is vulnerable to glitches by design of the specific cipher.

About Discretix

Discretix is a semiconductor intellectual property company that develops and licenses advanced embedded security solutions for resource-constrained environments, such as wireless devices and smart-cards, where stringent limits apply to the cost, size and power consumption of the target devices.

Discretix technology has already been adopted by some of the major vendors of wireless baseband and application chipset, as well as smart-card IC vendors.

Discretix Technologies Ltd.

Corporate Headquarters	**Representative in Japan:**
43 Hamelacha Street	Triangle Technologies KK
Beit Etgarim	Sogo-Hirakawacho Bldg.
Poleg Industrial Zone	4F 1-4-12
Netanya 42504	Hirakawacho Chiyoda-ku
Israel	Tokyo, Japan
Tel: +972 9 885 8810	Te l: +81 3 5215 8760
Fax: +972 9 885 8820	Fax: +81 3 5215 8765
Email: marketing@discretix.com	Email: japan.info@discretix.com

Appendix B GAPASE file: `simulation.cfg`

```
# simulation.cfg : Configuration file for simulation.
# This file is used by run_simulation script to simulate a design on nansim.
#
# Created by Asier Goikoetxea Yanci <asier.goikoetxea@ekb.atmel.com>
# 2004
#
# Version 1.7: 2008-I-16
# * Variable ROUNDS added
# * Variable FULL_RESET added
# * Variable FORCE_NETS added
# * Variable LOAD_RUN_CYCLES added
# * The example of KEY_NET and DATA_NET changed
#
# Version 1.6: 2007-VI-19
# * Variable VOLTAGE_NODE added
# * Variable TEMPERATURE added
# * Variable MODEL_LIB added
# * Variable MODEL_LIB_CALLS added
# * Variable LOG_LEVEL added
#
# Version 1.5: 2006-VI-24
# * Variable TARGET_KEY_BITS added to decide which bits are going to be targeted
#
# Version 1.4: 2006-V-24
# * DESIGN_RC_TYPE, DETACH and CPU_NUMBER variables added
#
# Version 1.3: 2005-XI-16
# * SAMPLE variable added
#
# Version 1.2: 2005-IX-xx
# * Required variable OUTPUT_DIR replaced with SIMULATION_NAME
# * Variable SIMULATION_TYPE created
# * Optional simulation parameters grouped according to simulation type
# * Simulation parameters reordered
# * Power analysis simulation enabled
# * Optional parameter R removed
# * Forgotten Ground pulse variables added (previously it was implemented)
#
# Version 1.1: 2005-I-21
# * It creates a directory in SIMVISION directory, where simulation's output
#   and configuration are stored.
#
# Version 1.0: 2004-XII-16
# * File created


###############################################
###############################################
##  Required information for the simulation  ##
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####


# This parameter defines the simulation's name and the name of a directory to be
# created on simvision directory, where simulation's output is going to be moved
# to and this file copied to, so simulation settings are also stored.
SIMULATION_NAME=

# Enter design's hspiceS netlist file name
#DESING_NETLIST=cmos/hspiceFinal
DESIGN_NETLIST=

# Enter a value to indicate the voltage to which the design is going to be
# powered. If POWER is not set, design would be powered with a constant source
# (to which noise and/or a pulse could be applied by setting POWER_NOISE and
# PULSE_* variables)
VOLTAGE=

# Enter the name of the supply node. This is the node used to power the circuit.
# It will be powered at VOLTAGE volts and POWER, if set.
#VOLTAGE_NODE=vdd!
VOLTAGE_NODE=
# Enter name of stimulus file. Note, when running a PA simulation, the contents
# of the file pointed by this parameter are not used during the simulation and
# they are overwritten instead, so if you want to keep a copy of this stimulus
# inputs, make sure that this is not the only copy you have of this file or
# create a file that you are happy to overwrite.
#STIMULUS=stimulus_in.sp
STIMULUS=

# Enter an integer to indicate the simulation time. Expected required
# simulation time will be calculated by the simulation environment when running
# a power analysis. If the provided simulation time is shorted than the expected
# one, the simulation environment may update it at the user's discretion.
SIMULATION_TIME=

# Enter a value to indicate the temperature in Celsius degrees to which the
# design is going to be exercised.
TEMPERATURE=

# Enter the model library name, path inclusive.
MODEL_LIB=

# Enter a comma separated list of the required library calls by the model.
#MODEL_LIB_CALLS=process_tolerances,mos_wcs,techno,nonmc,nonmatching,model_58k8,rlow,c
high
MODEL_LIB_CALLS=

# Define which kind of simulation is going to be performed. Possible values
# are: {NORMAL,GLITCH,PA}
# *NORMAL -> only DESIGN_RC_* optional parameters are checked too
# *GLITCH -> DESIGN_RC_* and glitch related parameters are checked too
# *PA     -> DESIGN_RC_* and power analysis parameters are checked too
SIMULATION_TYPE=

# Indicate whether the Nanosim simulation should be detached from the console.
# By enabling detaching the simulation, on the event of closing the console
# where the simulation environment is called from, on-going Nanosim simulations
# won't be killed. Possible values are: {YES,NO}
# NOTE: with this option, ONLY Nanosim simulations are detached. This option
# does not detach the simulation environment itself. This must be done by the
# user.
DETACH=

####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
##  Required information for the simulation  ##
###############################################
###############################################

#----------------------------------------------------#

#####################################################
#####################################################
##  General optional information for the simulation  ##
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####


# Enter the desired message log level for this simulation. This is an optional
# parameter. Possible values are: {HIGH,LOW}
LOG_LEVEL=

# Enter design's RC parasitic netlist file name. This is an optional parameter.
#DESING_RC_NETLIST=
DESIGN_RC_NETLIST=

# Enter design's RC netlist format type. Two are the valid formats for this
# file: SPEF and HSPICE. Default format is HSPICE. If this parameter is left
# undefined or set to HSPICE, HSPICE format will be understood. For SPEF
# formatted files, this parameter should be defined with SPEF. This is an optional
# parameter, only to be entered if DESIGN_RC_NETLIST is defined.
#DESING_RC_TYPE=
DESIGN_RC_TYPE=

####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
##  General optional information for the simulation  ##
#####################################################
#####################################################

#----------------------------------------------------#
```

```
#######################################################
#######################################################
##  Glitch optional information for the simulation  ##
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####

# Enter name of power file which defines the power source for the design. Power
# vector must be called vpower_waveform and nominal VOLTAGE values used. As this
# waveform would be added to original VDD, a 1 on vpower_waveform would be
# interpreted as 2*VOLTAGE, a 0 would be interpreted as VOLTAGE, a -1 would be
# interpreted as 0v and so on. This is an optional parameter.
#POWER=power.sp
POWER=

# Enter time to start applying the power waveform. POWER_START time is
# calculated as DELAY_FOR_PULSE. This parameter is only looked at if POWER
# parameter is defined. If POWER is defined and this parameter is not defined,
# it will be set to default value (i.e. 0 nano-seconds).
#POWER_START=0n
POWER_START=

# Define a pulse to be applied to power line. Parameter DELAY_FOR_PULSE must be
# defined in order to parse pulse definition parameters. If DELAY_FOR_PULSE is
# not defined, remaining pulse definition parameters will be ignored. Use
# absolute time values. Use nominal values of VOLTAGE. This pulse would be added
# to VDD, therefore a nominal voltage of 1 would be interpreted as 2*VOLTAGE, a 0
# would be interpreted as VOLTAGE, a -1 would be interpreted as 0v and so on.
# Parameters PULSE_START_VALUE and PULSE_END_VALUE are the only optional ones,
# all other parameters must be defined if a pulse is desired. If optional
# parameters are not defined, they would be interpreted as 0, i.e. VOLTAGE.
# DELAY_FOR_PULSE value can be setup to refer the pulse to a particular clock
# edge, e.g. positive-edge of clock number 14 (it is also possible to refer to a
# neg-edge). The formula to calculate delay value for a neg-edge is:
#
#  (n_clocks * period) --> to set pulse's origin at beginning of neg-edge
#  (n_clocks * perios) + tfall/2 --> to set pulse's origin at midway of neg-edge
#
# The formula to calculate delay value for a pos-edge is:
#
#  (n_clocks * period) + period/2 --> to set pulse's origin at beginning of
#       pos-edge
#  (n_clocks * period) + period/2 + trise/2 --> to set the pulse's origin at midway
#       of pos-edge
#
# All pulse times are relative times, i.e. an offset value to DELAY_FOR_PULSE,
# also, all time must be positive numbers. Therefore, if applying a pulse just
# before clock 'z' starts is desired, DELAY_FOR_PULSE should be calculated for
# clock 'z-1'. If clock 'z' is used instead, the earliest the pulse could start
# is DELAY_FOR_PULSE, and only on the case that PULSE_START_TIME is 0n.
#
# When setting a time value, use n for nanoseconds; u for microseconds; and m for
# milliseconds. This is an optional parameter.
# required pulse parameter
DELAY_FOR_PULSE=

# required pulse parameter
PULSE_START_TIME=

# optional pulse parameter (optional)
PULSE_START_VALUE=

# required pulse parameter
PULSE_P1_TIME=

# required pulse parameter
PULSE_P1_VALUE=

# required pulse parameter
PULSE_P2_TIME=

# required pulse parameter
PULSE_P2_VALUE=

# required pulse parameter
PULSE_END_TIME=

# optional pulse parameter (optional)
PULSE_END_VALUE=

# Enter name of file that defines GND waveform source for the design. Vector
# must be called vgnd_waveform and use nominal VOLTAGE values. As this waveform
# would be added to original GND, a 1 on vpower_waveform would be interpreted as
# VOLTAGE, a 0 would be interpreted as 0, a -1 would be interpreted as -VOLTAGE
# and so on. This is an optional parameter.
#GND=gnd.sp
GND=

# Enter time to start applying the GND waveform. GND_START time is calculated
# as GND_DELAY_FOR_PULSE. This paremeter will be looked at only if parameter GND
# is defined. If parameter GND is defined and this one is not, it will be set to
# default value (i.e. 0 nano-seconds).
#GND_START=0n
GND_START=

# Define a pulse to be applied to GND line. Parameter GND_DELAY_FOR_PULSE must
# be defined in order to parse pulse definition parameters. If
# GND_DELAY_FOR_PULSE is not defined, remaining pulse definition parameters will
# be ignored. Use absolute time values. Use nominal values of VOLTAGE. This
# pulse would be added to GND, therefore a nominal voltage of 1 would be
# interpreted as VOLTAGE, a 0 would be interpreted as 0, a -1 would be
# interpreted as -VOLTAGE and so on. Parameters GND_PULSE_START_VALUE and
# GND_PULSE_END_VALUE are the only optional ones, all other parameters must be
# defined if a pulse is desired. If optional parameters are not defined, they
# would be interpreted as 0, i.e. 0v. GND_DELAY_FOR_PULSE value can be setup to
# refer the pulse to a particular clock edge, e.g. positive-edge of clock number
# 14 (it is also possible to refer to a neg-edge). The formula to calculate
# delay value for a neg-edge is:
#
#  (n_clocks * period) --> to set pulse's origin at beginning of neg-edge
#  (n_clocks * period) + tfall/2 --> to set pulse's origin at midway of neg-edge
#
# The formula to calculate delay value for a pos-edge is:
#
#  (n_clocks * period) + period/2 --> to set pulse's origin at beginning of
#       pos-edge
#  (n_clocks * period) + period/2 + trise/2 --> to set pulse's origin at midway
#       of pos-edge
#
# All pulse times are relative times, i.e. an offset value to
# GND_DELAY_FOR_PULSE, also, all time must be positive numbers. Therefore, if
# applying a pulse just before clock 'n' starts is desired, GND_DELAY_FOR_PULSE
# should be calculated for clock 'n-1'. If clock 'n' is used instead, the
# earliest the pulse could start is GND_DELAY_FOR_PULSE, and only on the case
# that GND_PULSE_START_TIME is 0n.
#
# When setting a time value, use n for nanoseconds; u for microseconds; and m
# for milliseconds. This is an optional parameter.
# required pulse parameter
GND_DELAY_FOR_PULSE=

# required pulse parameter (optional)
GND_PULSE_START_TIME=

# optional pulse parameter
GND_PULSE_START_VALUE=

# required pulse parameter
GND_PULSE_P1_TIME=

# required pulse parameter
GND_PULSE_P1_VALUE=

# required pulse parameter
GND_PULSE_P2_TIME=

# required pulse parameter
GND_PULSE_P2_VALUE=

# required pulse parameter
GND_PULSE_END_TIME=

# optional pulse parameter (optional)
GND_PULSE_END_VALUE=

# Enter file name of noise to be applied to stimulus. Again, use nominal values.
# Prefixing signal name is recommended. Prefixes 'org_' and 'nom_' are forbidden
# on noise signal's vector names. This is an optional parameter.
#STIMULUS_NOISE=noise.sp
STIMULUS_NOISE=

# Enter time to start applying the noise to the stimulus. STIMULUS_START is
# calculated as DELAY_FOR_PULSE. This parameter will be looked at only if
# parameter STIMULUS_NOISE is defined. If parameter STIMULUS_NOISE is defined
# this one is not, it will be set to default value (i.e. 0 nano-seconds).
#STIMULUS_NOISE_START=0n
STIMULUS_NOISE_START=

####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
##  Glitch optional information for the simulation  ##
######################################################
######################################################

#------------------------------------------------------------#

##############################################################
##############################################################
##  Power analysis optional information for the simulation  ##
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####
####VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV####

#
# How simulation will be performed
#
#          ____      ____      ____      ____      ____      ____
#clk  ____/    \____/    \____/    \____/    \____/    \____/    \*
#
#
#reset_/\________/\________/\________/\________/\________/\_______**
#
#           _____     _____     _____     _____     _____     ____
#run  -----/     \---/     \---/     \---/     \---/     \---/    **
#
#         _______   _______   _______   _______   _______   ______
#key  ---<_______>-<_______>-<_______>-<_______>-<_______>-<______**
#
#         _______   _______   _______   _______   _______   ______
#data ---<_______>-<_______>-<_______>-<_______>-<_______>-<______**
#
#
#
# * Note: Clock signal will be automatically generated by simulation environment
# ** Note: Although some registers have asynchronous reset input, external reset
#          signal will be generated considering that such input does not exist
#          or that it is synchronous.
# *** Note: Data, key, enable and run registers' input and outputs will be
#           forced only after reset and before next positive clock edge, being
#           released during positive clock cycle.
#

# Enter cryptographic block identifier. Possible values are:
# {DES, AES-128, AES-192, AES-256}
# This parameter is mandatory when performing a Power Analysis.
#CRYPT=DES
CRYPT=

#DOM, CORRELATION
#ANALYSIS_TYPE=

#YES or NO
#GENERATE_D_FUNC=

#
#AMOUNT_OF_PLAIN_TEXT=

# Enter clock frequency at which the cryptographic block will be exercised.
# Provide clock frequency in KHz. Note that this entry is required even when
# cryptographic block is asynchronous. On such case, this signal will not affect
# it. This parameter is mandatory when performing a Power Analysis.
#CLOCK=12000
CLOCK=

# Enter the number of rounds to be simulated. In order to carry a normal DPA on
# a DES module, getting the power trace of the first round is enough. For more
# complex power analysis (e.g. second order DPA), the power trace of two rounds
# is required. This parameter lets you configure the amount of rounds to be
# simulated and extracted. The higher the number of rounds, the longer it will
# take the simulation process.
#ROUNDS=1
ROUNDS=

# Enter the desired number of times each data should be run. The more samples
# are taken the better. However, different crypto blocks may require different
# amount of samples to expose their weaknesses, were very weak/unsecure block
# require less samples than those theoretically secure blocks.
#SAMPLES=10
SAMPLES=

# Deprecated. It wont be used in future versions
CPU_NUMBER=

# Enter the name of clock input signal. This parameter is mandatory when
# performing a Power Analysis.
#CLOCK_IN_NAME=clkCpu
CLOCK_IN_NAME=

# Enter the name of reset input signal. This parameter is mandatory when
# performing a Power Analysis.
#CRYPT_RESET=reset
CRYPT_RESET=

# Determine whether the cryptographic module should be fully reset or not
# after generating power consumed by each plaintext. When fully resetting the
# cryptographic module, all registers and nets will be force to a reset status.
# When not fully resetting it, only registers will be forced to a reset status.
# On this later case, KEY and DATA registers are excluded. This parameter is
# mandatory when performing a Power Analysis. Possible values are:
# {YES, NO}.
#FULL_RESET=YES
FULL_RESET=

# Determine whether the KEY and DATA registers' inputs or outputs should be
# forced at the beginning of each plaintext encryption. This parameter is
# mandatory when performing a Power Analysis. If not defined, a warning message
# will be generated and it will be set to INPUT. Possible values are:
# {INPUT, OUTPUT}.
#FORCE_NETS=INPUT
FORCE_NETS=

# Determine whether KEY and DATA load operation and launching the encryption
# should happen in the same clock cycle or not. If they happen in the same
# clock cycle, encryption simulation will be launched straight away. If they
# happen on different clock cycles, one clock cycle will be used to load KEY
# and DATA and the following one will launch the encryption. This parameter is
# valid only when forcing the inputs of KEY and DATA registers (FORCE_NETS). In
# such case, this parameter is mandatory. If not defined, a warning message
# will be generated and it will be set to DIFFERENT. Possible values are:
# {SAME,DIFFERENT}
#LOAD_RUN_CYCLES=DIFFERENT
LOAD_RUN_CYCLES=

# Enter the name of the register that triggers encryption loop/run. This
# parameter is mandatory when performing a Power Analysis.
#START_CRYPT=run
START_CRYPT=

# Enter hex key for cryptographic modules. This parameter is mandatory when
# performing a Power Analysis.
#KEY=1234DEF
KEY=

# Enter target key bits. This parameter is mandatory when performing a Power
# Analysis. Possible values are:
# SBOX_1 -target bits on SBOX 1;
# SBOX_2 -target bits on SBOX 2;
# SBOX_3 -target bits on SBOX 3;
# SBOX_4 -target bits on SBOX 4;
# SBOX_5 -target bits on SBOX 5;
# SBOX_6 -target bits on SBOX 6;
# SBOX_7 -target bits on SBOX 7;
# SBOX_8 -target bits on SBOX 8;
#TARGET_KEY_BITS=ALL
TARGET_KEY_BITS=

# Deprecated. It wont be used in future versions
# Enter the name of registers used to store the key. This is an optional
# parameter. This or KEY_NET parameter must be defined. If this parameter is
# defined, parameter KEY_NET will be ignored. If this parameter is not defined,
# then parameter KEY_NET will be checked.
#KEYREG=keyreg
KEYREG=

# Deprecated. It wont be used in future versions
# Enter the name of registers used to store the data. This is an optional
# parameter. This or DATA_NET parameter must be defined. If this parameter is
# defined, parameter DATA_NET will be ignored. If this parameter is not defined,
# then parameter DATA_NET will be checked.
#DATAREG=desreg
DATAREG=

# Deprecated. It wont be used in future versions
# Enter each key registers' input and output data net's name. These names can be
# found in design's netlist. If left empty, run_simulation script will
# automatically search for appropriate net names. This is an optional parameter.
# This or KEYREG parameter must be defined. If KEYREG parameter is defined, this
# parameter will be ignored. If KEYREG parameter is not defined, then this
# parameter will be checked.
#KEY_NET=    KEY0_0in  KEY0_0enn  KEY0_0out  KEY0_1in  ...  KEYn_nenn KEYn_nout
#               |         |          |         |        |      |         |
#KEY_NET=    NET53      NET55      NET123    NET125    ...  NET155    NET150
KEY_NET=

# Deprecated. It wont be used in future versions
# Enter each data registers' input and output data net's name. These names can
# be found in design's netlist. If left empty, run_simulation script will
# automatically search for appropriate net names. This is an optional parameter.
# This or DATAREG parameter must be defined. If DATAREG parameter is defined,
# this parameter will be ignored. If DATAREG parameter is not defined, then this
# parameter will be checked.
#DATA_NET=   DATA0_0in  DATA0_0enn  DATA0_0out  DATA0_1in  ...  DATAn_nenn DATAn_nout
#               |          |           |          |         |      |          |
#DATA_NET=   NET53      NET55       NET123     NET125      ...  NET155     NET150
DATA_NET=

####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
####^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^####
##  Power analysis optional information for the simulation  ##
##############################################################
##############################################################
```

Appendix C

These are the different types of Single Event Effects inducible by radiation as covered in [34]:

- ***Single Event Upset:*** *Generally a transient condition in which the output state of a digital device is affected (e.g., a bit-flip in a memory cell or a change of state of an inverter). The state recovers after being rewritten, causing no permanent damage.*
- ***Single Event Latchup:*** *Condition characterized by an anomalous high current state, where the current can go from picoamps (10^{-12}) to amps. If the power is cycled before damage occurs, SEL may only be transient.*
- ***Single Event Burnout:*** *Permanent failure due to maintaining a high current state for an extended period of time.*
- ***Single Event Gate Rupture:*** *Permanent failure caused by dielectric breakdown in the semiconductor oxide layer.*
- ***Single Event Total Dose:*** *Permanent failure caused by a single particle that produces enough ionization or displacement damage in a transistor to permanently degrade its performance. SETD are more significant now because technology advances have led to very small transistor sizes, making it possible for a particle's path to encompass much of an entire transistor.*
- ***Single Event Transient:*** *Effects (e.g., current spikes in operational amplifiers) of short time duration that may lead to other effects downstream of the affected site that are longer in duration.*