Exploration of FDSOI Back-Biasing Techniques to Hinder Cryptographic Attacks Based on Leakage Current

Kenneth Palma

July 2023

List of Figures

1.1 Moore's law observational trends 14
1.2 Prospective trends on IoT nodes 16

3.1 FSM diagram 30
3.2 Setup and hold Timing 31
3.3 Schematic representation of the input of a D flip-flop 31

4.1 Cross-sectional view of a flip-well CMOS FDSOI pair (Figure extracted from [1]). 49

5.1 Leakage current consumption of a standard-cell D flip-flops as a function of absolute body bias. While this is just an example of a particular flip-flop, it is representative of the behaviour observed. 53
5.2 Dummy cryptosystem implementing a bitwise XORing function between the plaintext ($X[0..7]$) and a secret key ($k[0..7]$) 54
5.3 Idealized discrete body bias generator 54
5.4 PCC between the leakage current of a register array and the HW calculated with all possible keys for a fixed body bias value $|V_{bbn}| = 0$ V 57
5.5 PCC between the leakage current of a register array and the HW calculated with for all possible keys, in the presence of a random body bias 58
5.6 Measured leakage current vs Hamming Weight of different plaintext X in an 8-bit register. The solid line depicts equation (4.3) for a given register, while the markers represent measured current values for different plaintexts displayed according to the calculated Hamming Weight with a correct key (Blue) and an incorrect key (Red). 63
5.7 Collection of register leakage current vs Hamming Weight curves. Each line represents a possible realization of Equation (4.3) for different values of I_0. The markers represent measured current values for different plaintexts displayed according to the calculated Hamming Weight with a correct key 65

5.8 PCC between the leakage current and the Hamming Weight of an 8-bit register array in the presence of a random body bias as a function of the maximum number of steps 67
5.9 PCC between the leakage current and the Hamming Weight of an 8-bit register array in the presence of a random body bias, with a fixed maximum Dynamic Range, as a function of the s_{max} . . 68
5.10 Effect of averaging on the PCC between the Leakage Current and the Hamming Weight of the register array in the presence of the proposed countermeasure with a fixed, maximum body bias DR of 1 V. It can be seen how averaging undermines the effect of the countermeasure for large number of repeated encryption processes. 71
5.11 PCC between the numerically simulated leakage current and the Hamming Weight of the theoretical register array under attack for every candidate key in the presence of the proposed countermeasure. The results are obtained with one single encryption process per plaintext. 73
5.12 PCC of the numerically simulated and averaged leakage current and the Hamming Weight, with 10 encryption processes per plaintext. 73
5.13 PCC of the numerically simulated and averaged leakage current and the Hamming Weight, with 100 encryption processes per plaintext. 74
5.14 PCC of the numerically simulated and averaged leakage current and the Hamming Weight, with 1000 encryption processes per plaintext. 74
5.15 Effect of averaging on the PCC between the Leakage Current and the Hamming Weight of a 128-bit register array, with 8 bits under attack, in the presence of the proposed countermeasure with a fixed, maximum body bias DR of 1 V and different number N of traces per plaintext. 77

6.1 PCC between the univariate leakage magnitude (Blue) and its HW, as described in the previous chapter, and between the bivariate leakage magnitude I_{sub} (Orange) and Z as a function of s_{max}. 87
6.2 Mean value of the random variable Z for all possible plaintexts as a function of the round key 88
6.3 Variance of the random variable Z for all possible plaintexts as a function of the round key 89
6.4 Schematic representation of the Trivium Stream Cipher 90
6.5 $t1$, the non-linear function under attack. The result from this function feeds into the first bit of register Y. 90
6.6 PCC between the magnitude I_{sub} and the Hamming Weight of interest in the Trivium as a function of samples per plaintext . . 95

6.7 ϵ as a function of both V_{bbn} and V_{bbp} for LVT, flip-well transistors at 27 ^{o}C. Each curve represents a possible value of V_{bbn}, which ranges from 0 V to 1 V, in 0.1 V increments (from top to bottom). The horizontal red line indicates the points of zero crossing of ϵ. 97
6.8 Reduced contour map of $\epsilon(V_{bbn}, V_{bbp})$. The lines represent the limits where $\epsilon = |1|$ nA, encasing an area where any possible combination of body bias values solves for $\epsilon \leq |1|$ nA. 98
6.9 Reduced contour map of $\epsilon(V_{bbn}, V_{bbp})$ for registers implemented with LVT, FBB transistors at 27 ^{o}C. The colored lines represent the limits where $\epsilon = |1|$ nA, encasing an area where any possible combination of body bias values solves for $\epsilon \leq |1|$ nA. The vertical lines showcase the inferior limits of V_{bbn} that can solve for the conditions 100
6.10 PCC between the Hamming Weight represented by the random variable Z and the bivariate leakage power model (Equation (6.5)) under noiseless conditions in the presence of a Symmetrical (blue) and a Current Balancing (orange) random body bias scheme, where ϵ is systematically kept at 1 nA 102
6.11 PCC between the Hamming Weight and the bivariate leakage power model (Equation 6.5) with non-algorithmic noise in the presence of a Symmetrical (blue) and a Current Balancing (orange) random body bias scheme, in the presence of non-algorithmic GW noise 103
6.12 PCC between the Hamming Weight and the bivariate leakage power model (Equation 6.5) as a function of averaged number of traces in the presence of non-algorithmic GW noise with a Symmetrical (Blue) and a Current Balancing (Orange) random body bias scheme 104
6.13 PCC between Z_n and the bivariate leakage power model with algorithmic and non-algorithmic noise (Equation 6.36) as a function of noise power under a current balancing boby bias, for different number of N averaged traces 105
6.14 PCC between the Hamming Weight and the bivariate leakage power model with algorithmic and non-algorithmic noise (Equation 6.36), as a function of noise power under a a Symmetrical (blue) and a Current Balancing (orange) random body bias scheme for $N = 1000$ averaged traces. 106
6.15 PCC between the Hamming Weight and the bivariate leakage power model of the Trivium with algorithmic and non-algorithmic noise (Equation 6.39), as a function of averaged samples under a Current Balancing (blue) and a Symmetrical (orange) random body bias scheme for a noise power of -134 dBW. 109

6.16 PCC between the Hamming Weight and the bivariate leakage power model of the Trivium with algorithmic and non-algorithmic noise (Equation 6.39), under a Current Balancing (blue) and a Symmetrical (orange) random body bias scheme for $N = 1000$ averaged traces. 110
6.17 Block system representation of the Trivium implemented as a collection of discrete-time state space matrices. The scope saves the total Hamming Weight of the three registers. 111
6.18 Contour map of $\epsilon(v_{bbn}, v_{bbp})$ at 75 C for the registers under study implemented with low threshold voltage, flipped well transistors. The lines represent the limits where $|\epsilon(v_{bbn}, v_{bbp})| \leq 1$ nA, encasing an area where every possible combination of V_{bbn} and V_{bbp} meet the imposed criterion. 114
6.19 Contour map of $\epsilon(v_{bbn}, v_{bbp})$ at 27 C for the registers under study implemented with regular threshold voltage, reverse body bias transistors. The lines represent the limits where $|\epsilon(v_{bbn}, v_{bbp})| \leq 1$ nA, encasing an area where every possible combination of V_{bbn} and V_{bbp} meet the imposed criterion. 115
6.20 Contour map of $\epsilon(v_{bbn}, v_{bbp})$ at 80 C for the registers under study implemented with regular threshold voltage, reverse body bias transistors. The lines represent the limits where $|\epsilon(v_{bbn}, v_{bbp})| \leq 1$ nA, encasing an area where every possible combination of V_{bbn} and V_{bbp} meet the imposed criterion. 115

7.1 Schematic representation of a two-stage CMOS Cross-Coupled Charge Pump 118
7.2 Layout of a designed single inverter 120
7.3 Non-overlapping two phase clock generator layout 121
7.4 Layout of the negative charge pump 121
7.5 Layout of the driver chain 122
7.6 Full layout of the positive and negative Body Bias generator . . . 123
7.7 NMOS hysteresis comparator 124
7.8 Pre-hysteresis stage schematic 125
7.9 Charge-pump pulse skipping control through the hysteresis comparator 126
7.10 Schematic of a threshold-referenced current source 128
7.11 Schematic representation of a PMOS threshold-referenced current source / voltage regulator 129
7.12 Output voltage vs P-Well body bias value (V_{bbp}) of the circuit depicted in Fig. 7.11. The voltage is measured at the node label as $net4$. 130
7.13 Block Diagram representing the possibility of implementing a control system to reach the desired value of V_{bbp} through the circuit depicted in Fig. 7.11 131
7.14 Simplified schematic representation of a possible implementation of part of the transfer loop that would enact the control system 131

7.15 Potential leakage current front-end sensor based on the voltage drop experienced by a shunt resistor . . . 132
7.16 Capacitive leakage current sensor schematic representation . . . 134
7.17 Block diagram representation of the current balancing seeking circuit . . . 138
7.18 Schematic of the ideal positive voltage charge pump representing the drivers . . . 138
7.19 Schematic of the ideal negative voltage charge pump representing the drivers . . . 139
7.20 Diagram representation of the hysteresis comparator. . . . 140
7.21 Contourn map of $|\epsilon(V_{bbn}, V_{bbp})|$ for D flip-flops under study implemented with RVT, RBB transistors at 80 ºC . . . 141
7.22 Contourn map of $|\epsilon(V_{bbn}, V_{bbp})|$ for D flip-flops under study implemented with RVT, RBB transistors at 80 ºC superimposed with its gradient function . . . 141
7.23 3D representation of the function $|\epsilon(V_{bbn}, V_{bbp})|$. . . . 142
7.24 Schematic representation of the Overshoot detector implementation through an XOR gate. . . . 143
7.25 Schematic representation of the comparators informing the progression of the drivers used. . . . 143
7.26 State transition diagram of the sensor portion of the control unit 145
7.27 State transition diagram of the drivers portion of the control unit 145
7.28 Analog-Mixed simulation results of the body bias seeking circuit at 80 ºC . . . 147
7.29 PCC between the Hamming Weight and the bivariate leakage power model (Equation 7.39) as a function of averaged number of traces in the presence of algorithmic and non-algorithmic GW noise with Current balanced body bias (Blue) and without a current balanced body bias (Orange) at 80 ºC . . . 149
7.30 PCC between Z_n and the bivariate leakage power model with algorithmic and non-algorithmic noise (Equation 7.39) as a function of noise power under a current balancing body bias at 80 ºC ($\epsilon = -273$ pA), for different number of N averaged traces . . . 150

8.1 Histogram depicting the distribution of ϵ for body bias values $V_{bbn} = -409$ mV and $V_{bbp} = 775$ mV for 200 samples . . . 153
8.2 PCC as a function of samples of Equation 8.10 at 80 C for the different distributions obtained through a Montecarlo analysis of registers under body bias conditions seen in Table 8.2 . . . 157
8.3 PCC as a function of samples of Equation 8.10 at 80 C for the different distributions obtained through a Montecarlo analysis of registers under body bias conditions seen in Table 8.2 considering a folded distribution . . . 159
8.4 Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 1000 samples per instance . . . 163

8.5 Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 5000 samples per instance 164
8.6 Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance 164
8.7 Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance 165
8.8 Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance 165
8.9 Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance without considering a folded normal distribution 166

10.1 Schematic representation of the circuit implementing the capacitive sensing capabilities of the TtD converter. The *reg*! label refers to the power pin of the registers under test 178
10.2 V_{c1} (purple) and V_{c0} (orange) as a function of time 179

List of Tables

2.1 One of 24 possible substitutions for strings of 2 bits 21
2.2 One of 8! possible permutations for strings of 8 bits. The numbers represent the position of the bits within the string. 22
2.3 XOR truth table 25
2.4 One of 24 possible substitutions for strings of 2 bits. The X1 and X2 bits represent the inputs of the substitution function, while Y1 and Y2, its outputs. 26

5.1 PCC for maximum dynamic range of the Body Bias for different number of averaging encryption processes when $s_{max} = 25$ 70
5.2 Success ratio of secret key identification in numerical simulations under different number of averaging samples 72
5.3 Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack without countermeasure for different number N of averaged traces per plaintext. 78
5.4 Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack with countermeasure for different number N of averaged traces per plaintext. 78
5.5 Forward Body Bias Registers at 80 C -Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack with countermeasure for different number N of averaged traces per plaintext. 79
5.6 Reverse Body Bias Registers at 27 C - Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack with countermeasure for different number N of averaged traces per plaintext. 80

6.1 Key Extraction Progression for different periods of the initialization phase. The pertinent bits of the Initialization Vector have been omitted 91
6.2 Probability Distribution of second XOR gate 94

6.3 Countermeasure parameters 101
6.4 PCC - Theoretical and Numerical Simulations of a 128 bit system in the absence of non-algorithmic noise 107
6.5 PCC - Theoretical and Numerical Simulations of a 128 bit system in the presence of non-algorithmic noise 108
6.6 First and Second Moments of Z_m 108
6.7 PCC - Theoretical and Numerical Simulations of a 128 bit system in the presence of non-algorithmic noise 113

7.1 Drivers' decoder truth table. The signals CD_1 and CD_0 represent the drivers' counter. 144
7.2 Simulation conditions 146
7.3 Results from an initial simulation at 80 ºC 147
7.4 Results at varying temperatures 148
7.5 PCC - Theoretical and Numerical Simulations of a 128 bit system in the presence of non-algorithmic noise 150

8.1 Single Driver Current Balancing Results at varying temperatures 154
8.2 Mean and standard deviation of registers' leakage current 155
8.3 Success rate of key identification for 1000 instances of a simulated CPA at 80 C 167
8.4 Success rate of key identification for 1000 instances of a simulated CPA at 50 C 167
8.5 Success rate of key identification for 1000 instances of a simulated CPA at 35 C 168
8.6 Success rate of key identification for 1000 instances of a simulated CPA at 80 C 168

Contents

1 Introduction **13**
- 1.1 Background . . . 13
- 1.2 Motivation . . . 14
- 1.3 Thesis Objectives . . . 16
- 1.4 Outline . . . 17

2 Fundamentals of Cryptography **19**
- 2.1 Types of Cryptosystems . . . 20
  - 2.1.1 Blockciphers . . . 21
  - 2.1.2 Stream Ciphers . . . 22
- 2.2 Security in Cryptosystems . . . 23
  - 2.2.1 The piling-up lemma and linear cryptanalysis . . . 24

3 Side Channel Attacks & Countermeasures **28**
- 3.1 Finite State Machines and Intermediate Variables . . . 29
- 3.2 Differential Power Analysis Attacks . . . 32
- 3.3 Correlation Power Analysis Attacks . . . 36
- 3.4 Countermeasures to PAA . . . 38
- 3.5 Conclusions . . . 42

4 Leakage Power Analysis Attacks, Countermeasures & FDSOI technology **43**
- 4.1 Technology Scaling and Leakage Currents as a Side-Channel . . . 43
- 4.2 Countermeasures to LPA attacks . . . 47
- 4.3 FDSOI Technology in Cryptosystems . . . 48

5 Random Body Bias as a countermeasure to Leakage Power Analysis Attacks **51**
- 5.1 Initial Investigation & Modelling . . . 51
- 5.2 CPA on Dummy Cryptosystem. Empirical approach . . . 53
- 5.3 Countermeasure Modelling . . . 58
  - 5.3.1 PCC without countermeasure . . . 58
  - 5.3.2 Random body bias scheme . . . 61
  - 5.3.3 Random Body Bias Analysis . . . 64

5.3.4 Random Body Bias: Trace Averaging . . . 68
5.3.5 Algorithmic Noise . . . 75
5.3.6 Other Conditions . . . 78
5.4 Conclusions . . . 80

6 Current Balancing Body Bias 82
6.1 Vulnerabilities to Random Body Bias Countermeasure . . . 83
6.1.1 The AES and Random Body Bias . . . 84
6.1.2 The Trivium and Random Body Bias . . . 88
6.2 Current Balancing Body Bias . . . 96
6.3 Results . . . 99
6.3.1 The AES and Current Balancing Body Bias . . . 100
6.3.2 The Trivium and Current Balancing Body Bias . . . 108
6.4 Temperature and Other Considerations . . . 113
6.5 Conclusions . . . 114

7 Current Balancing Body Bias Circuit Implementation 117
7.1 Circuit Implementation . . . 118
7.1.1 Body Bias Generator . . . 118
7.1.2 Hysteresis Control . . . 122
7.2 Current Balancing - Initial Exploration . . . 127
7.2.1 Threshold-based Voltage reference . . . 127
7.2.2 IDDq . . . 132
7.3 Time-to-Digital Converter . . . 133
7.3.1 Low vs Regular Threshold Transistors . . . 136
7.4 Control Unit & Drivers . . . 137
7.5 Results . . . 146
7.5.1 Circuit Simulation . . . 146
7.5.2 PCC . . . 148
7.6 Conclusions . . . 150

8 Variability Assessment 152
8.1 Initial Approach - Montecarlo Simulations . . . 153
8.2 Variability Analysis - Ideal Distribution . . . 155
8.2.1 Results . . . 157
8.3 Variability analysis - Register Instances . . . 158
8.3.1 Folded Normal Distribution . . . 160
8.3.2 Results . . . 163
8.4 Conclusions . . . 168

9 Conclusions 170
9.1 Publications . . . 172

10 Annexes **173**
10.1 Annex I - Perfect Secrecy 173
10.2 Annex II - PCC Derivation Under Random Body Bias Scheme . 175
10.3 Annex III - Current Balancing Proof of Concept Design 178
10.3.1 V_{ref} 178
10.3.2 f_s and C_{sense} 180
10.3.3 Charge-Pumps 181
10.3.4 Number of Bits, n_{min} and n_{MAX} 182
10.4 Annex IV - PCC for register instances under process and mismatch variability 184

Acknowledgements

This research was funded by Spanish MCIN/AEI/10.13039/501100011033, Project PID2019-103869RB-C33 (Project VIGILANT), by Secretaria d'Universitats i Recerca of Generalitat de Catalunya, and by the European Social Fund.

To my parents, whose sacrifice will be but a blip in the speck of existence, never sufficiently recognized.

To my friends, whose eyes glaze with the hue of incomprehension when I relay my new findings.

To my tutor, for his patience, guidance and confidence.

To E.B.

Chapter 1

Introduction

1.1 Background

A cryptographic system is a circuit which implements a cryptographic algorithm. That is, a System on Chip (SoC), ASIC, or a microcontroller one of the functions of which is to securely prepare information to be sent from the circuit onto, at the very least, a second node or circuit.

For cryptography is the science that studies the secure transmission of information between communicating parties, attempting to ensure that even if a message is intercepted during transmission its contents remain secure and unable to produce information.

For the most part of its formalization during the 20th century, cryptography has remained an abstract endeavour. That is to say, the development of cryptographic algorithms and primitives has been a purely mathematical approach. During this development, it sufficed to prove that a particular cryptographic algorithm was statistically secure. With that, if no means of attacking the algorithm could be proven and the statistical properties complied with the understanding of security at the time, an algorithm could be developed, and be assumed secure, without ever leaving a sheet of paper.

With the boom of integrated circuits and the ever shrinking Moore's Law [2], cryptographic systems would be adapted and migrated into the world of digital computation, where the throughput could be faster, the secret keys longer, and no dedicated implementation was necessary.

But by doing so, cryptographic systems were opened to a type of vulnerability that did not exist before. At the cusp of the 20th century, researchers realized that the power consumption of circuits implementing cryptographic algorithms, when studied during the execution of encryption, could leak information regarding the secret key at the core of the encryption process [3] [4]. In this case, the power consumption could be proven to be a side-channel.

A side-channel is a physical magnitude that can be measured and that somehow "leaks" information about the current state of a cryptosystem. The nature

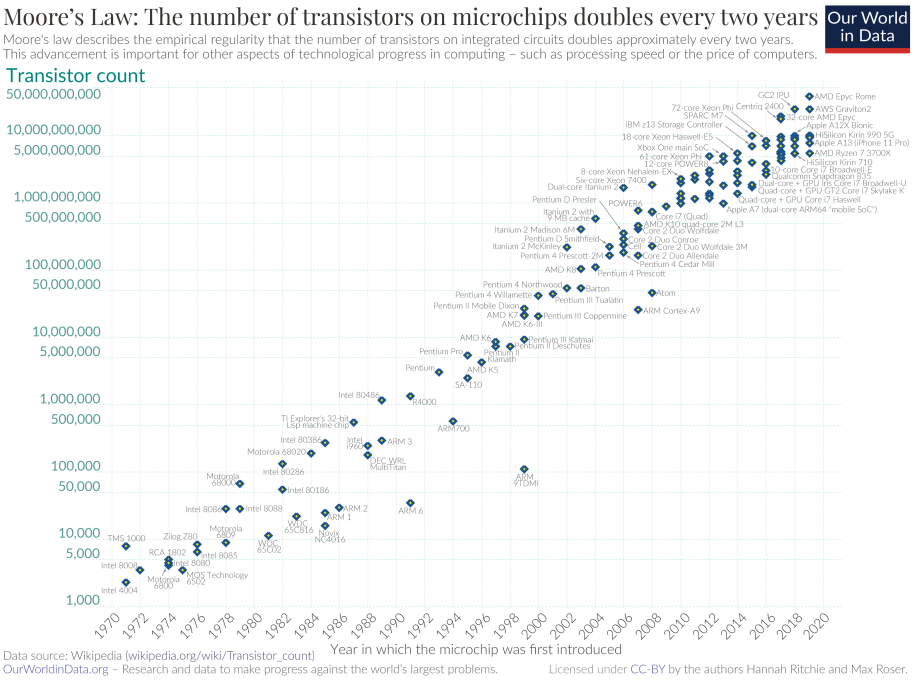

Figure 1.1: Moore's law observational trends

and properties of a side-channel depend on the nature and properties of the system that implements a cryptographic algorithm. They could be heat, mechanical vibrations, electromagnetic emanations or, as in IC implementations, power consumption of the circuit.

Thus, at the transition of the millennial, researchers realized that by studying and collecting power traces of a cryptosystem executing an encryption algorithm under distinct conditions, it could be possible to gather sufficient information so as to derive the secret key, thus breaking the security of the system.

This research focused at the time on smart-cards, where a hefty volume of research has been produced [5] [6]. But more complex microcontrollers, FPGA implementations and ASIC implementations have, with time, been proven to be vulnerable to what are called side-channel attacks.

1.2 Motivation

Thus began an arms-race between this newly gained insight into side-channel attacks that utilized power consumption, also called Power Analysis Attacks, and the study of means to protect circuits against them.

As the knowledge regarding the nature of power consumption as a side channel was refined, as well as with the optimization regarding their statistical and

informational theory analysis, power analysis attack became more efficient [7].

Thus, means to protect circuits against power analysis attacks, traditionally called countermeasures, began to be developed in parallel. In time, a variety of countermeasures was developed as a response to the ever growing understanding of power analysis attacks, attempting to protect cryptographic implementations from prying eyes. While varied in nature, countermeasures fall in roughly two categories; those that "disconnect" the power consumption of the cryptographic system from the power pins through means of integrated power converters of some kind, and those that either introduce noise or decrease the magnitudes of the signals of interest.

With these came a myriad of challenges. Countermeasures are an additional circuit embedded along the original cryptographic circuit, meaning that they increase both the area and power overhead. At the same time, countermeasures must be proven to be effective protecting against power analysis attacks. But a multitude of elements must be taken into account, and protection in some regards does not mean in all regards [8]. At best, a countermeasure is soundly safe until determined otherwise.

At the same time, it was not sufficient to develop countermeasures considering only an understanding of power analysis attacks, for as technology advances and changes, so change the nature of power consumption as a side-channel.

Perhaps the clearest example of this is that brought by the explosion of submicron technology of CMOS integrated circuits. That is, at the beginning of the inception of power analysis attacks and countermeasures, the focus point on how to derive information regarding the inner state of cryptographic systems was that of dynamic power. The power consumed at each rising edge of the clock was the main source of information that could help break a cryptosystem, and so most countermeasures were designed taking this element into account.

However, as transistor nodes entered the nanometer scale, leakage or static power consumption became a prominent source of power consumption [9]. Not only that, but at the beginning of the 2010's, researchers realized that static power consumption could also be used as a side-channel. Thus, a new vulnerability was discovered, and so a new arm-race began anew.

The main focus on power analysis attacks that use leakage power as a side-channel has been on targeted registers arrays, whose leakage current consumption heavily depends on the state that is being stored. With these came new types of countermeasures that attempted to hinder the quality of information extracted from leakage power.

At the same time, with the prospective increase in the number of IoT nodes featured in connected devices (Fig. 1.2) come several security questions that need to be addressed. Side-channel attacks are invasive, in the sense that an attacker requires physical access to the device. Isolated IoT nodes, with an understanding of being designed for low power, low latency and low computational capacity, are particularly at risk, and thus newer security staples are required.

And just as newer transistor nodes have furthered the understanding of other sources of vulnerabilities, newer transistor technologies present an opportunity in the study of countermeasures .

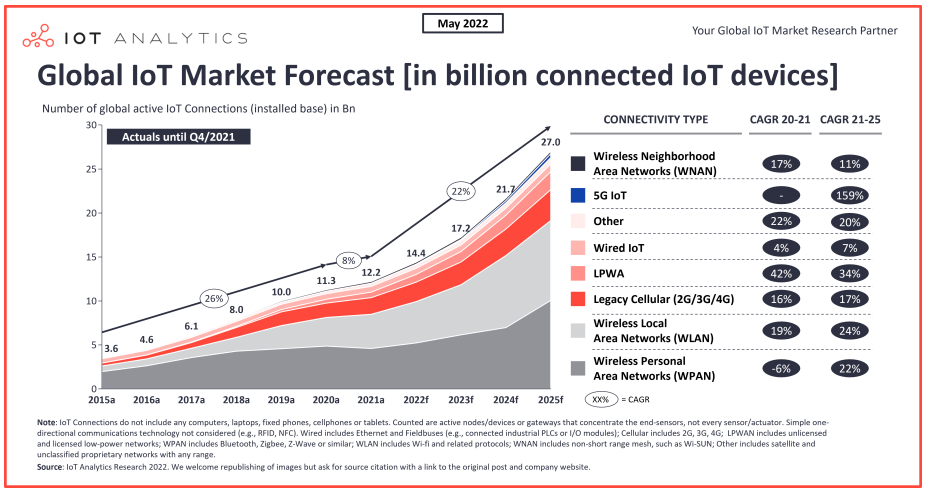

Figure 1.2: Prospective trends on IoT nodes

In particular, FDSOI transistors which, given their structure, allow the application of a wide range of body bias values without loss of functionality, present interesting characteristics regarding their leakage power consumption [10].

First of all, their structure has been proven to diminish the overall leakage current that they incur by curtailing small-channel effects. At the same time, through the modification of their body bias, FDSOI transistors can see their threshold voltage dynamically modified and, with it, their leakage current consumption. As such, FDSOI transistors pose an interesting opportunity to explore in the development of countermeasures against leakage power analysis attacks.

Nonetheless, little literature exists in this regard. As such, this thesis explores the use of CMOS FDSOI technology, through its back biasing capabilities, to design countermeasures against leakage power analysis attacks of cryptographic systems.

1.3 Thesis Objectives

Based on the notions introduced regarding the nature of cryptosystems and side-channel attacks, the objective of this thesis is, mainly, the exploration of FDSOI body biasing techniques to develop effective countermeasures against leakage power analysis attacks.

While some deviations exist from the initial outline of the research plan, the work here presented follows the spirit of its intended conception.

Thus, a systematic approach to the development of countermeasures utilizing the properties of FDSOI technology is employed.

The overall objectives of this work are then subdivided into smaller, progres-

sive problems, which inform the overall structure of the subsequent chapters of this thesis:

- Develop an understanding of the effect of body bias on the leakge current consumption of chosen standard cells implemented in FDSOI technology. In this work, we focus on sequential logic, namely D flip-flops.
- Based on this initial exploration, conceive of a body bias scheme that can help hinder the secret key extraction of cryptosystems through leakage power analysis attacks. In order to do so, the proposed countermeasure must be able to both respond to the statistical properties of cryptosystems and the idiosyncrasies that arise regarding their leakage power consumption. At the same time, the implementation of such a countermeasure must be feasible. Thus, two further objectives arise.
- Test the effectiveness of the proposed countermeasure under conditions that can sufficiently represent real implementation conditions.
- Offer a proof of concept circuit capable of implementing the developed countermeasure.

In order to do so, we rely heavily on electrical and numerical simulations using Virtuoso ADE and Matlab.

1.4 Outline

Chapters 2 through 4 serve as a formalized and more thorough introduction, outlining the necessary materials to be able to understanding this work. Chapter 2 presents an introduction to cryptographic principles, notions, and mathematical derivations that will be used throughout this work. Chapter 3 formalizes the definition of side-channel attacks as related to the power consumption of cryptographic systems, a brief chronology and state of the art, along the countermeasures that have been designed as a response. Chapter 4 summarizes some of the studies made so far regarding leakage current consumption as a side-channel, as well as currently existing countermeasures. At the same time, it introduces more formally the structure and properties of FDSOI transistors, and how they could be used to design countermeasures. It also details the objectives of this thesis, and a brief presentation of the methodology used, more thoroughly developed in the bulk of the thesis.

Chapters 5 through 8 are the bulk of this work, presenting the analysis, design, and implementation of various countermeasures against vulnerabilities in static power consumption of cryptographic systems.

Chapter 5 implements within itself all the objectives summarised above, presenting an initial exploration and implementation of a countermeasure based on a body bias scheme, testing its effectiveness through analysis and numerical simulations of power attacks. At the same time, the chapter also introduces

notions of what types of circuits could be necessary to implement it, along with some simulations of such circuits.

Chapters 6 and onward are a response to the first introduced countermeasure. After its implementation, further analysis revealed that, under certain conditions, the countermeasure proposed in Chapter 5 presents significant vulnerabilities and can be easily surpassed. Thus, Chapter 6 presents an analysis on these vulnerabilities and introduces the theoretical basis of a new countermeasure not susceptible to them, along simulations of its effectiveness. Chapter 7 then presents the design and implementation of a circuit capable of carrying out the body biasing scheme theoretically described in Chapter 6. Finally, Chapter 8 presents an analysis on the effect of mismatch and process variability on the effectiveness of the countermeasure, once implemented with the circuit.

Chapter 2

Fundamentals of Cryptography

Cryptography is the science that studies the secure transmission of information. At its core, cryptography attempts to deal with a simple yet fundamental precept: any message transmitted between two parties can inevitably be intercepted. Cryptography is then the conversion of readily interpretable information into a collection of symbols so devoid from its original message that despite the certainty of its interception, cannot yield any meaning.

Communication, or the transfer of information, is assumed to take place between, at the very least, two parties: One sender, and one receiver. Cryptography literature has commonly referred to the aforementioned parties as Alice and Bob. A third party is also presumed, a nefarious eavesdropper that can readily intercept any communication between the intended parties. The communicating channel, or medium through which the messages are sent, is for the purposes of this work, irrelevant.

In a supposedly secure communication, the sender (Alice) must collect the data that they wish to transmit and modify it. The intended receiver (Bob), on the other hand, must be able to revert the aforementioned modification upon having received data, reverting it to its original form. The nefarious eavesdropper, however, is supposed to be unable to perform the same task as Bob, left only with statistically appropriate gibberish.

Thus, the emitter and intended receiver must share a common knowledge of the process through which the data is modified. This process, at the same time, must be reversible. The process of converting data into an inscrutable text is called encryption, whereas its inverse, the transformation of inscrutable text into its original form is called decryption. Further formalizing these notions, the text or message in its original, interpretable form is called the Plaintext. The encrypted message, which is ultimately transmitted, is called the Ciphertext.

An encryption and decryption processes are, in and of themselves, insufficiently secure, for if the method of obfuscation were to be discovered and

no other consideration played a role, all transmissions made in the same fashion would be at risk. Thus, a mutable element within the same encryption-decryption process should exist. This element constitutes the secret key. The encryption and decryption processes are then dependent on the secret key, and both Alice and Bob must be aware of the particular key used during the process.

The above discussion can be formalized as follows [11]. A cryptosystem is formed by 5 elements, denoted as $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where:

- $\mathcal{P}$ is a finite set of all possible plaintexts.
- $\mathcal{C}$, is a finite a set of all possible ciphertexts (encrypted plaintexts)
- $\mathcal{K}$, the keyspace, is a finite set of all possible keys
- both $\mathcal{E}$ and $\mathcal{D}$ a function space, partitioned by the keyspace, such that, for each element $k \in \mathcal{K}$, there exists an element $e_k \in \mathcal{E}$ and and element $d_k \in \mathcal{D}$ that define a mapping between the set of Plaintexts elements and the Ciphertext, such that:

$$e_k : \mathcal{P} \to \mathcal{C}$$
$$d_k : \mathcal{C} \to \mathcal{P}$$

From the above discussion it can be inferred that this mapping is a bijection, provided that $|\mathcal{P}| = |\mathcal{C}|$, where the operator $|\cdot|$ denotes the order, that is, the number of elements of the sets, which we can assume equal for this text. As such, decryption is a mapping that defines an inverse transformation for every element of the Plaintext:

$$d_k(e_k(x)) = x \quad \forall x \in \mathcal{P}$$

While the plaintext and cipertext can be composed of any arbitrary collection of symbols or letters, and the keyspace can be any manner of ruling, for the purpose of this work it will be assumed that either three of them will exclusively comprise strings of bits. At the same time, for an equal set order we can assume, for simplicity's sake, that both the plaintext and ciphertext will be of the same length n. That is, any plaintext and ciphertext element belongs to S ($x, y \in S$), the set of all bit strings of length n ($S = \{0,1\}^n$). This is not necessarily the case for the secret key, which depending on the encrypting algorithm can be a string of bits of different length than that of the plaintext. This will be further discussed in subsequent sections.

2.1 Types of Cryptosystems

While there exist multiple and overlapping ways with which to categorize cryptosystems, for the purpose of this work we will limit ourselves to some categorizations that inform the development and contents of this thesis. One of the most basic distinctions among cryptosystems is that between blockcipher and stream ciphers.

2.1.1 Blockciphers

Blockcipher apply their encryption and decryption processes on indivisible units of data (blocks of data, hence the name). Not only that, but each block of data is encrypted utilizing the same secret key.

That is, if we consider a string of blocks of data such as:

$$x = x_1 x_2 x_3 ... x_m, \tag{2.1}$$

where each element x_i of the string (with $1 \leq i \leq m$) informs a block of data, the encryption process is performed element to element (block to block) with the same secret key.

$$y = e_k(x) = e_k(x_1)e_k(x_2)e_k(x_3)...e_k(x_m) \tag{2.2}$$

At the same time, each block of data is a string of bits of the same length ($n \geq 1$). The value of n depends on the type of cryptosystem.

Two basic block ciphers are substitution ciphers and permutation ciphers. As its name implies, substitution ciphers substitute the value of a block of data by another possible value according to some ruling. For bit strings of length n, where $|\mathcal{P}| = |\mathcal{C}| = 2^n$, a substitution cipher is a bijective mapping such that:

$$\pi_S : \{0,1\}^n \leftrightarrow \{0,1\}^n. \tag{2.3}$$

Table 2.1 exemplifies a very simple substitution scheme for a bit string of length $n = 2$.

While for the set of all possible bit strings of length n there are a total of $2^n!$ possible substitutions, in later sections it will be seen that not all substitutions are secure.

00	10
01	00
10	11
11	01

Table 2.1: One of 24 possible substitutions for strings of 2 bits

Permutation ciphers, on the other hand, modify the positions of bits on a string. That is, as opposed to a substitution of the whole word, an n-bit string permutation shuffles the bits according to some rule. Permutations are, again, bijective mappings:

$$\pi_P : \{0,1\}^n \leftrightarrow \{0,1\}^n. \tag{2.4}$$

However, in this case, for an n-bit string there are a total of $n!$ possible permutations. Table 2.2 exemplifies one possible permutation of an 8-bit string.

While it is apparent that neither of the above block ciphers make use of a secret key composed of bits, it will be shown in later sections how permutation

x	0	1	2	3	4	5	6	7
$\pi_P(x)$	2	3	1	5	6	7	4	0

Table 2.2: One of 8! possible permutations for strings of 8 bits. The numbers represent the position of the bits within the string.

and substitution functions are core components of various cryptosystems in use to this day.

2.1.2 Stream Ciphers

The other type of cryptosystems previously mentioned are stream ciphers. In stream ciphers, the unit of encrypted data block are strings of bits of length n=1. That is to say, each bit is encrypted separately. Not only that, but the encryption key for each separate bit, as opposed to block ciphers, is a distinct bit element from what is typically called the keystream, denoted as z. That is:

$$y = e_z(x) = e_{z1}(x1)e_{z2}(x2)...e_{zm}(xm) \tag{2.5}$$

In most stream ciphers, the encryption process is simply an XORing between one bit of the plaintext (x) and one bit of the keystream (k):

$$y_1 = e_{z1}(x_1) = x_1 \oplus k_1 \tag{2.6}$$

The encryption, as such, is a very simple process. Thus, the principal function of the cryptosystem is to produce a secure stream of bits that compose the keystream from an initial state, or seed. This initial state is defined by the secret key. The stream cipher, based on the initial state and a governing algorithm produces bits of the keystream with which the plaintext is XORed, bit by bit.

This keystream generation is a matter of the current state of the stream cipher (z_t) and a function, called the keystream generator ($g(\cdot)$). Thus, each cycle, or unit of time, the value of the keystream is updated based on the previous state:

$$z_{t+1} = g(z_t) \tag{2.7}$$

These types of cryptosystems, similar in function to a Moore Finite State Machine, are called synchronous stream ciphers. Among the possible different synchronous stream ciphers, one of the better studied keystream generators are Linear-Feedback shift-registers.

Consider an n-bit register whose contents represents the state of the stream cipher. Each cycle, the state of the register is updated according to a function:

$$
\begin{aligned}
z_4(t+1) &= z_1(t) \oplus z_2(t) \\
z_3(t+1) &= z_4(t) \\
z_2(t+1) &= z_3(t) \\
z_1(t+1) &= z_2(t)
\end{aligned} \tag{2.8}
$$

This recursive linear relationship can be represented as a matrix linear transformation

$$
Z[T+1] = \boldsymbol{A} \cdot Z[T] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \end{bmatrix} \tag{2.9}
$$

Clearly, in this case the seed or secret key would be the value of the bits z_1 through z_4 at time $t = 0$.

In this particular case the keystream is generated according to the following expression:

$$
z(t+1) = z_1(t) \tag{2.10}
$$

While Linear Feedback stream ciphers are widely studied, or precisely because they are widely studied, they implement poor stream ciphers in as much as they are insecure.

In general, any system that attempts to perform some sort of encryption operation in a linear fashion can easily be broken, given the ease with which computers can manipulate matrices. That includes operations such as substitution and permutations.

For this reason, cryptosystems rely heavily on non-linear operations in order to avoid the ease with which linear systems of equations can be solved. This implies that neither the permutations, substitutions, or feedback elements in keystream generators should be designed so that they can be represented as linear operations on their inputs. This is a necessary but not sufficient condition for security.

While it is beyond of the scope of this text to extensively derive or present notions of security in cryptosystems, some basic principles are necessary.

2.2 Security in Cryptosystems

One of the most fundamental principles in cryptosystem design or security is the Kerckhoff's principle [12], which states that any cryptosystem is to be designed assuming that the entirety of the algorithm that it implements is going to be known. Based on this principle, all cryptosystems in use nowadays are public, and the proposal of new cryptosystems is submitted to extensive public scrutiny

so as to determine whether some manner exists by which the system can be broken.

Thus, the fundamental element that allows secure communication is the fact that an unintended eavesdropper does not posses the key that was used to encrypt the message.

Notwithstanding, the question remains whether the eavesdropper can become an attacker and derive information from the ciphertext, assuming that they posses all pertinent information regarding the structure and functionality of the system.

From this notion arises the idea of perfect secrecy, derived by Shannon in 1949 [13]. While a longer derivation can be found in the Annexes (10.1), the important notions can be summarized here.

Given a plaintext set, a ciphertext set and key space, a perfectly secret cryptosytem is one in which the following equality holds:

$$P[X = x|Y = y] = P[X = x] \tag{2.11}$$

Where X represents the set of possible plaintexts and Y the set of possible ciphertext, and $P[\cdot]$ represents the probability that each random variable has of adopting a specific value.

This equality implies that having access to the ciphertext does not convey an attacker with any information regarding the original plaintext.

Under such circumstances, the conditions of perfect secrecy is met when $P[K = k_i] = P[Y = y]$. That is, the structure and probability distribution of the keyspace is such that it follows that of the ciphertext space.

In the case where the order of each set $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{K}|$, where the cryptosystem is a bijection such that there is no collision between plaintext elements (that is, $e_{k1}(x) = e_{k2}(x)$ only if $k_1 = k_2$), this implies that $P[K = k_i] = \frac{1}{|\mathcal{K}|}$ when $P[Y = y] = \frac{1}{|\mathcal{Y}|}$, the discrete distribution with the highest entropy.

Thus, the notion of perfect secrecy imposes a structure upon the plaintext, ciphertext and keyspace sets, as well as upon their probability distribution.

2.2.1 The piling-up lemma and linear cryptanalysis

Consider two bits representing two independent random variables (X_1 and X_2) that can adopt values 1 or 0 with probabilities:

$$P[X_1 = 1] = p_1, P[X_1 = 0] = 1 - p_1 \tag{2.12}$$

$$P[X_2 = 1] = p_2, P[X_2 = 0] = 1 - p_2, \tag{2.13}$$

and define the bias ϵ of such probability distribution as:

$$p_1 = \frac{1}{2} - \epsilon_1 \tag{2.14}$$

$$(1 - p_1) = \frac{1}{2} + \epsilon_1. \tag{2.15}$$

With $-1/2 \leq \epsilon \leq 1/2$, where ϵ represents the "amount of probability" with which the random variable X_1 deviates from $1/2$.

Given the independence of the random variables, the joint probability distributions of X_1 and X_2 can be expressed :

$$\begin{aligned} P[X1 = 0, X2 = 0] &= (1 - p_1) \cdot (1 - p_2) \\ P[X1 = 0, X2 = 1] &= (1 - p_1) \cdot p_2 \\ P[X1 = 1, X2 = 0] &= p_1 \cdot (1 - p_2) \\ P[X1 = 1, X2 = 1] &= p_1 \cdot p_2 \end{aligned}$$

Consider now a new random variable that can be expressed as:

$$Z = X_1 \oplus X_2. \tag{2.16}$$

That is, Z is a random variable that arises as the XORing of X_1 and X_2. A truth table for an XOR function can be seen in table 2.3

X_1	X_2	Z
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.3: XOR truth table

The probability distribution of Z can be obtained by noting that:

$$\begin{aligned} P[Z = 1] &= P[X1 = 0, X2 = 1] + P[X1 = 1, X2 = 0] \\ P[Z = 0] &= P[X1 = 0, X2 = 0] + P[X1 = 1, X2 = 1] \end{aligned} \tag{2.17}$$

Expressing the above equations as a function of the bias of X_1 and X_2 we obtain:

$$P[Z = 1] = P[X_1 \oplus X_2 = 1] = \frac{1}{2} - 2\epsilon_1\epsilon_2 \tag{2.18}$$

$$P[Z = 0] = P[X_1 \oplus X_2 = 0] = \frac{1}{2} + 2\epsilon_1\epsilon_2 \tag{2.19}$$

The piling-up lemma defines an expression for the probability of random variables of this form for an arbitrary number of instances ($Z = X_1 \oplus X_2 \oplus ... \oplus X_n$):

$$P[X_1 \oplus X_2 \oplus ... \oplus X_n = 0] = \frac{1}{2} + 2^{k-1} \prod_{i=1}^{k} \epsilon_i \tag{2.20}$$

In perfectly biased random variables, the value of ϵ is equal to 0, and thus the bias of a collection of random variables that include some unbiased one, is also equal to zero ($\prod_{i=1}^{k} \epsilon_i = 0$)

With this, we have sufficient background to derive the importance of the piling-up lemma.

Consider the substitution presented in Table 2.2, repeated here in a more convenient way and recall that substitutions are a core component of some cryptosystems.

X_1	X_2	Y_1	Y_2
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	1

Table 2.4: One of 24 possible substitutions for strings of 2 bits. The X1 and X2 bits represent the inputs of the substitution function, while Y1 and Y2, its outputs.

Out of this particular substitution we can form random variables of the form:

$$(a_1 \cdot X_1) \oplus (a_2 \cdot X_2) \oplus (a_3 \cdot Y_1) \oplus (a_4 \cdot Y_2) = 0 \tag{2.21}$$

Where the different a_i are constant coefficients that can adopt values of either 1 or 0, essentially selecting whether one of the random variables has an effect.

Consider the case $X_1 \oplus Y_1 = 0$. We can find the bias of this expression by noting the times it evaluates to 0 out of this particular substitution. It is easy to see that $P[X_1 \oplus Y_1 = 0] = 1/2$, denoting an unbiased random variable.

However, if we analyze the random variable resulting from $X_1 \oplus Y_2 = 0$, it can be seen that $P[X_1 \oplus Y_2 = 0] = 1$.

Consider a very simple cryptosystem with secret key $K = \{k_1, k_2\}$ (a two bit key) and an encryption operation such that:

$$Y = \pi_s(P \oplus K) \tag{2.22}$$

That is, the encryption process first XORs in a bitwise manner the secret key with two bits of plaintext ($P = \{p_1, p_2\}$). The result of this operation is then the input of substitution function presented in Table 2.4 and the output ($Y = \{y_1, y_2\}$) is the ciphertext.

Assume that we do not know the value of the secret key, but wish to obtain it. Note that:

$$X_1 \oplus Y_2 = P_1 \oplus K_1 \oplus Y_2, \tag{2.23}$$

and assume that we have control over the plaintext and access to the ciphertext, once computed. We can then make a guess over the value of k_1 and

run the cryptosystem for all its possible values and calculate the probability $P[P_1 \oplus K_1 \oplus Y_2 = 0]$.

In this particular case, guessing the correct key would yield $P[P_1 \oplus K_1 \oplus Y_2 = 0] = 1$, while guessing the incorrect key would yield $P[P_1 \oplus K_1 \oplus Y_2 = 0] = 0$ (they essentially contain the same information).

While this example is so simple so as to be trivial, it serves to illustrate some important notions. Firstly, as previously stated, it can be seen that not all types of substitution are secure, even if they are purely non-linear. These types of attacks are referred to as linear-cryptanalysis [14], because they make linear approximations of non-linear substitution functions based on the probability distributions of random variables formed by its input-output variables.

From this also arises the importance of designing substitution functions that are unbiased for all the possible bit combinations. In fact, in all the work presented in further sections, it will always be assumed that all bits follow a discrete uniform probability distribution with unbiased probabilities which, to an extent, also follows the principle of perfect secrecy introduced by Shannon.

In essence, the conditions of perfect secrecy impose a given structure to the plaintext, ciphertext and keyspace elements, while the dangers of linear cryptanalysis impose conditions on the probability distribution of intermediate values.

At the same time, linear cryptanalysis attacks also illustrate an important concept in what is the bulk of this thesis. Namely, that of a testing function. As it will be seen in the following sections, Power Analysis Attacks of cryptosystems require of a statistical model against which to contrast "guesses" of sub-bits of the secret key. Essentially, a distinguishing function is required.

Chapter 3

Side Channel Attacks & Countermeasures

Cryptographic functions can be designed, implemented and analyzed entirely from a mathematical perspective. Their security can be tested against diverse methods of attack and their vulnerabilities addressed without ever leaving the world of abstract notions. However, the moment they are implemented in physical system, some considerations appear that cannot be previously addressed.

Physical implementations of cryptosytems are subjected to the natural laws of physics. As such, they can produce mechanical vibrations, dissipate heat, emanate electromagnetic waves, present timing constraints in their execution, or present distinct dynamic or static power consumption.

These physical magnitudes can, and have been proven to leak cryptographic information of system implementations. These leaks, at the same time, can be exploited under distinct circumstances to indirectly break physical implementations of cryptosystems through their measurement.

The diverse physical magnitudes that can leak cryptographic information are referred to as Side-Channels, and attacks on cryptosystems engaged through their measurement are called Side-Channel Attacks (SCA). On this thesis we are going to focus on a subset of SCA called Power Analysis Attacks (PAA).

In Power Analysis Attacks, the Side-Channel utilized to obtain information of the cryptosystem is the power consumption of the cryptosystem during its encryption operation. The objective of these attacks is to obtain sufficient information "encoded" in the power trace of the cryptosystem that allows the derivation of the secret key being used by the system.

For all intents and purposes, we can assume that cryptographic systems are mostly implemented in either ASIC form, an FPGA or through a software implementation in a microcontroller. As such, we are going to treat these circuits mostly as being implemented in CMOS technology. All models (or all those models known to the author) utilized to attack cryptosystems through PAA stem from these notions.

One of the first formalized mentions of PAA was [3], where the authors observed that some cryptosystems implementing modular exponentiation algorithms take different times to produce an operation depending on the value of the exponent. The authors noted that, by measuring the power consumption of the circuit during the encryption operation, they could identify the time window and distinguish different values of the exponent based on the time of execution. These types of Power Analsysis Attacks are called Timing Attacks for apparent reasons.

Shortly thereafter the authors introduced Simple Power Analysis Attacks (SPA) and, most importantly, Differential Power Analysis Attacks (DPA) [4]. Differential Power Analysis Attacks proved to be a relatively simple and inexpensive, albeit somewhat intrusive method of deriving sub-bits of the secret key of block cryptosystems, allowing the breaking of a cryptosystem with relative ease in ways never addressed before. They will be described momentarily, but some important notions should be addressed first.

3.1 Finite State Machines and Intermediate Variables

Encryption algorithms, when described mathematically, present an order to their operations. However, in their description there is no sense of timing or time consumed; simply the order in which the different functions should be applied. On the contrary, in physical implementations, the order must be maintained and considerations to the timing of their executions respected.

Thus, most encryption operations present variables that are the result of previous evaluations of a function and that, at the same time, serve as the inputs of further functions. These are typically called intermediate variables.

In a way, implementations of encryption algorithms can be thought of as a black box of sorts. Its inputs are the plaintext and secret key, and its output is the ciphertext. The algorithm takes a finite, albeit not immediate time to execute and during its execution the black box processes intermediate values that are not readily accessed.

Carrying this line of thought a bit further, implementations of encrypting algorithms can be considered Finite State Machines (FSM) in which, once primed with the inputs (plaintext and secret key), the algorithm is executed transitioning from state to state. In line with the black box model, the intermediate values never serve as an output and only when the ciphertext is computed does the machine present it.

It is clear that, in physical implementations, these intermediate variables must be stored, both as a result of an operation and to serve as the input of the next step of the process. Thus, we can assume that, continuing with the assumption of circuits implemented in digital technology, these variables are stored in temporary flip-flops. For example, in the AES encryption algorithm, these storing elements are represented by the state matrix, while the Trivium

stream cipher [15], being a shift-register, is in and of itself a storing element.

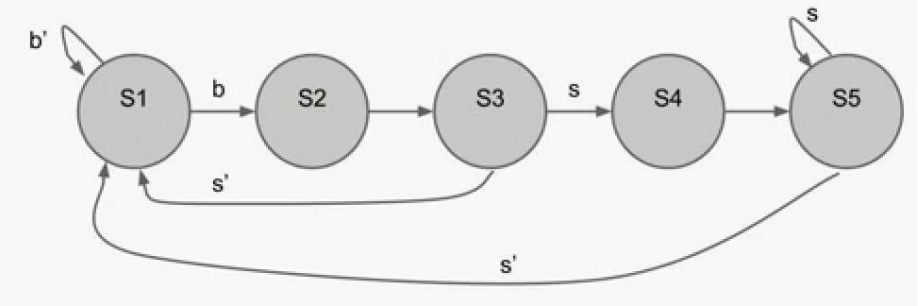

Figure 3.1: FSM diagram

Consider now an intermediate variable U. In a digital system, this intermediate variable will be a string of bits of length n ($U \in \{0,1\}^n$). The value of this intermediate variable U_j at state S_j is a function of the same intermediate variable and the previous state:

$$U_{Sj} = f(U_{S_{j-1}}, S_{j-1}) \tag{3.1}$$

Where $f(\cdot,\cdot)$ is a function that computes part of the encrypting algorithm. However, at the very first state (S_1), U_1 depends exclusively on the input plaintext and the secret key ($U_{s1} = f(x,k)$). Given the Kerckhoff's principle, the encryptying algorithm is known, so every state transition represents a function that is known to the public. Thus, albeit indirectly, the value of U at every possible state is also a function of the plaintext and the secret key, no matter how complicated the relation can be ($U_{Sj} = f_j(f_{j-1}(...f_1(x,k)...))$.

In a typical cryptosystem, these intermediate variables are not readily accessed and cannot be directly observed. However, consider a function $\mathcal{L} : U \to \mathcal{R}$ that maps a string of bits to a physical magnitude. This function, called a leakage function, informs a side-channel.

Going back to equation 3.1, whatever the function is that informs the updated value of U, the datapath must respect the setup time of the storing elements and thus the evaluation of the function $f(U_{j-1}, S_{j-1})$ must precede the clock edge. Thus, the input of the storing element "sees" the result before it can store it (or so it should be without timing violations).

What is meant by "seeing" the result is that, in digital circuits, the evaluation of a boolean function is going to present either a 1 (V_{dd}) or a 0 (GND). As such, the equivalent input capacitance of the register, formed by the parasitic output capacitances of the driving element, along with the gate capacitances of the input, must be charged or discharged.

The energy consumed by the charging of a capacitor in an RC network can be shown to be (including resistive losses):

$$E = C_{in} V_{dd}^2 \tag{3.2}$$

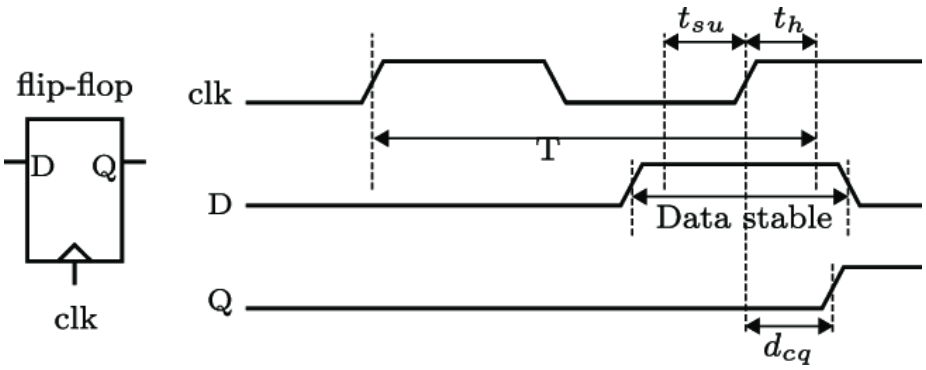

Figure 3.2: Setup and hold Timing

And the average power during a single charge cycle can be expressed as:

$$P = C_{in} \cdot V_{dd}^2 \cdot f_{clk} \cdot \alpha \tag{3.3}$$

In this model of dynamic power consumption, the factor α, called the activity factor, is the estimated probability that a transition from 0 to 1 will take place. However, in the particular case of cryptographic function, it is noteworthy to consider that α is actually a boolean function of intermediate variables:

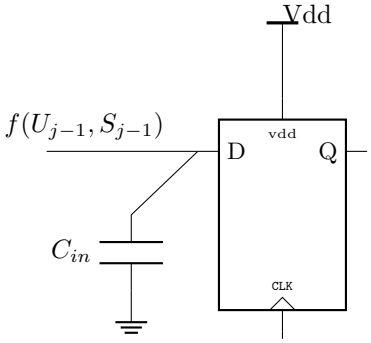

Figure 3.3: Schematic representation of the input of a D flip-flop

$$\alpha = f(U_{j-1}, S_{j-1}) \tag{3.4}$$

And, since intermediate variables are functions of the plaintext and the secret key, equation (3.3) can be expressed as:

$$P = C_{in} \cdot V_{dd}^2 \cdot f_{clk} \cdot f(x, k) \tag{3.5}$$

Resulting in a model in which the dynamic power consumed by the system is dependent on the evaluation of a boolean function of intermediate variables; that

is to say, the plaintext and the secret key. Thus, power consumption informs a leakage function of cryptographic information.

3.2 Differential Power Analysis Attacks

In their seminal paper introducing Differential Power Analysis attacks (DPA)[4], Kocher et al. present the concept of exploiting the power trace of a cryptographic system as a source of information. By introducing a shunt resistor between the power pin and the voltage source of the system, it becomes possible to measure the voltage drop or the current flow across the resistive element, obtaining a power trace of a cryptosystem during the execution of its encrypting algorithm.

Simple Power Analysis (SPA) attacks rely on the interpretation of the visual representation of the power trace to infer information on cryptographic functions, such as when they begin or end, how long they take to execute, etc. DPA attacks, on the other hand, are somewhat more involved in their execution, but also more powerful.

Some assumptions must be made to address the manner of the attack. Firstly, it is implied that an attacker has access to the physical device, can perform measurements on it, and has control over the plaintext that is fed into the cryptosystem, for as many executions as they deem necessary. This informs a model of a strong attacker, and it is going to be maintained during the rest of this work.

The attack is performed by carrying out a total of N experiments. That is, N random plaintexts are introduced into the cryptosystem, and the power trace for the execution of the algorithm measured.

For each measurement, the power trace is modelled as follows:

$$P(t) = P_I(t) + P_a(t) + P_{na}(t) \tag{3.6}$$

Where $P_I(t)$ stands for "Power of Interest", P_a for "Algorithmic Power" and P_{na} for "Non-algorithmic Power". This nomenclature is not necessarily extended in the field of side-channels, but it is representative of each of the terms and is adopted here for clarity.

$P_a(t)$ and $P_{na}(t)$ represent the power consumed by other parts of the circuit, elaborated further in the text. The term $P_I(t)$ arises from the notions introduced in subsection 3.1. That is, we are assuming that $P_I(t)$ is of the form:

$$P_I(t) = C_{in} \cdot V_{dd}^2 \cdot f_{clk} \cdot f(x, k) \tag{3.7}$$

Where $f(x, k)$ is a function of sub-bits of plaintext and sub-bits of the secret key. The function, boolean in nature, and known to the attacker, evaluates into either a 1 or a 0. This function, called the function or bit of interest, is executed at a certain time during the encryption process τ, or time of evaluation.

Thus, equation 3.7 can be somewhat abstracted as:

$$P_I(t) = \begin{cases} C_{in} \cdot V_{dd}^2 \cdot f_{clk} \cdot f(x,k) & \text{when } t = \tau \\ 0 & \text{when } t \neq \tau \end{cases} \tag{3.8}$$

Thus indicating that $P_I(t)$ evaluates to 0 at any time other than the time of evaluation. This is consistent with the notion that the function of interest $f(x,k)$ only consumes power at a given point in time.

Given that the attacker knows the plaintext used in each experiment, they can calculate the result of the function $f(x,k)$ by guessing the secret key. As such, for each possible key guess k_i, the attacker can partition the power traces into two groups:

$$\begin{aligned} P_{1_{ki}}(t) &= \{P(t) | f(x,k_i) = 1\} \\ P_{0_{ki}}(t) &= \{P(t) | f(x,k_i) = 0\} \end{aligned} \tag{3.9}$$

Based on the notions introduced in section 2.2.1, we can assume that the function $f(x,k)$ is unbiased regardless of the secret key and plaintext introduced. Thus, each partition contains approximately $r_1 \approx r_0 \approx \frac{N}{2}$ power traces independently of the correctness of the secret key, for a total of $r_1 + r_0 = N$ partitioned traces.

If the correct key is guessed, the different power traces will be appropriately partitioned. That is, $P_1(t)$ will contain all the power traces in which $f(x,k) = 1$, and $P_0(t)$ all the power traces in which $f(x,k) = 0$. However, again, assuming that the function $f(x,k)$ is unbiased, incorrect keys will partition each group such that they contain approximately the same number of traces that evaluate the function of interest to 1 and 0.

At this point, each partition is averaged and subtracted.

$$P_{\Delta_{ki}}(t) = \overline{P}_{1_{ki}}(t) - \overline{P}_{0_{ki}}(t) \tag{3.10}$$

Where:

$$\overline{P}_{1_{ki}}(t) = \frac{1}{r_1} \sum_{r_1} P_{1_{ki}}(t) \tag{3.11}$$

Expanding the averaged form of $P_\Delta(t)$ we make the following observations, dropping, for the moment, the sub-key nomenclature k_i:

$$P_\Delta(t) = (\overline{P}_{I_1} - \overline{P}_{I_0}) + (\overline{P}_{a1} - \overline{P}_{a0}) + (\overline{P}_{na1} - \overline{P}_{na0}) \tag{3.12}$$

The term P_{na} represent the summation of random sources of noise, such as Johnson-Nyquist noise, measurement noise, $1/f$ noise, etc. and, as such, are modeled as a Gaussian White Noise (GWN), with a given variance σ_{na}^2 and mean $\mu_{na} = 0$. Thus, it can be assumed that, for a sufficiently large number of traces N, both terms $\overline{P}_{na1} \approx \overline{P}_{na0} \approx \mu_{na} = 0$, and thus vanish from equation 3.12.

The terms P_a refer to algorithmic noise. Algorithmic noise represents the power consumed by the evaluation of cryptographic functions that are not the function of interest in the attack.

As an example consider an attack on an 8-bit cryptosystem. The attack is performed on a single bit, so at every point in time there are at least 7 bits of the cryptosystem that introduce algorithmic noise. That is, the evaluation of these 7 bits, which we are going to assume take place in parallel, consume power according to an equation similar to that of 3.7:

$$P_a = V_{dd}^2 \cdot f_{clk} \cdot \sum_{j=1}^{7} (C_{in_j} \cdot f_j(x,k)) \tag{3.13}$$

Since the plaintext is chosen at random, and we can assume that the cryptosystem is unbiased, each $f_j(x,k)$ is, in turn, a discrete uniform random variable that can adopt either a value of 1 or 0 with equal probability. The expected value of P_a at the time of evaluation is then, approximately:

$$E[P_a] = C_{eq} \cdot V_{dd}^2 \cdot f_{clk} \cdot \frac{7}{2} \tag{3.14}$$

Given that this notion holds regardless of how the partition of the power traces is made for a single bit, for a sufficiently large number of experiments we have that $\overline{P}_{a1} \approx \overline{P}_{a0} \approx E[P(a)]$. Thus, the second term on equation 3.12 also vanishes. This is valid for any clock period within the execution of the algorithm, and is not limited to the time of evaluation τ of the bit of interest.

Finally, we can assume that, for incorrect keys, the partitions P_1 and P_0 produce an equal distribution of evaluations to 1 and 0 of the function of interest in either partition, while a correct key will place all the evaluations that result in a 1 in P_1, and all the evaluations that result in a 0 in P_0.

Thus, for incorrect keys:

$$(\overline{P}_{I_1} - \overline{P}_{I_0}) \approx 0 \tag{3.15}$$

While, for correct keys:

$$(\overline{P}_{I_1} - \overline{P}_{I_0}) \approx \epsilon \tag{3.16}$$

This difference ϵ arises under the assumption that the circuit consumes a distinct amount of power when it charges the input capacitances than when it discharges them.

As such, for correct keys:

$$P\Delta(t) = \begin{cases} \epsilon & \text{when } t = \tau \\ 0 & \text{when } t \neq \tau \end{cases} \tag{3.17}$$

Thus, the correct key guesses would produce spikes in the visual representation of $P\Delta(t)$ at the time of evaluation of the bit of interest, identifying the secret key.

In [5], the authors further develop the statistical model of DPA attacks providing a figure for the Signal-to-Noise Ratio (SNR) of $P\Delta(t)$ at the time of evaluation:

$$SNR \approx \frac{\sqrt{N} \cdot \epsilon}{\sqrt{4\sigma_{na}^2 + n \cdot \epsilon^2}} \tag{3.18}$$

Where n is the number of bits that introduce algorithmic noise and N is the number of averaged traces.

The authors of [5] also extend the statistical model of DPA to include attacks that target multiple bits at the same time, instead of just one bit of interest. This presents some benefits, in that the intensity of the signal produced by multi-bit attacks is increased by a factor of d, the number of bits targeted.

Thus, equation 3.18 becomes:

$$SNR \approx \frac{\sqrt{N} \cdot d \cdot \epsilon}{\sqrt{4\sigma_{na}^2 + n \cdot \epsilon^2}} \tag{3.19}$$

However, multi-bit DPA attacks, as outlined in the same article, present some limitations, in that some of the traces measured cannot be used in partitions, reducing the total number of measurements available. Thus, a trade-off between signal intensity and number of available traces arises.

While these models are theoretically and, to a given extent, empirically sound, DPA were observed to exhibit behaviours that were not sufficiently captured through these models. Specifically, two phenomena were observed: That, at times other than the time of evaluation, "peaks" were observed in the $P\Delta(t)$ traces, and that these peaks also occurred when evaluating incorrect keys.

These ghost peaks, as they are referred to in the literature, were studied in different articles. In [16] the authors noted that the existence of these ghost peaks could be a consequence of hidden correlations, either in time, algorithmic correlation to incorrect keys, or both.

The initial models presented on DPA assume that the correct secret key, and only the correct secret key, is statistically correlated to the power consumption of the algorithm. Not only that, but that this correlation is exclusively present at the time of evaluation. In a way, this treats the algorithmic noise of the cryptographic system as a white noise, in the sense that the expected value of the autocorrelation function of the power trace $E[P\Delta(t) \cdot P\Delta(t-\tau)]$ is zero for any lag other than zero.

A series of articles ([17][18][19]) demonstrated how, depending on the S-Boxes of substitution and permutation networks, incorrect keys can exhibit higher correlations and differential peaks. These correlations are algorithm-dependent.

The existence of temporal correlations could be explained, among other factors, by the fact that DPA attacks measure dynamic power and, as such, transitions between states. Since transitions depend on the initial value of the state and the final value of the state, temporal correlations can arise between the transition and all the previous states that inform it.

To address these issues, Brier et. al. [20] developed a new form of statistical analysis that improves upon the limitations of Differential Power Analysis Attacks: Correlation Power Analysis Attacks.

3.3 Correlation Power Analysis Attacks

Correlation Power Analysis (CPA) attacks are performed similarly to DPA. There is a function of interest, evaluated at a precise time, dependent on sub-bits of the plaintext and sub-bit of the secret key. A large number of random plaintexts are introduced into the cryptosystem and, for each encryption process, the power consumed by the circuit is measured and stored.

The differences stem from the manner in which the statistical analysis is performed. CPA attacks rely on the correlation between power consumed and data processed, numerically established through the calculation of the Pearson Correlation Coefficient (PCC).

The PCC, typically denoted with ρ, explores the degree to which two random variables can be expressed linearly through each other (e.g.: $Y = a \cdot X + b$, with a and b being constant coefficients). Being a normalized function, it can present values between $-1 \leq \rho \leq 1$, with the extreme cases of -1 and 1 denoting perfect correlation, and $\rho = 0$ indicating the absence of any correlation (but not necessarily independence).

Equation 3.20 depicts how to calculate the PCC between two random variables, where $Cov(\cdot, \cdot)$ denotes the Covariance function, and $Var(\cdot)$ the variance function.

$$\rho_{x,y} = \frac{Cov(x, y)}{\sqrt{Var(x)}\sqrt{Var(y)}} \tag{3.20}$$

Regarding the manner in which the PCC is utilized to determine whether a secret key in a cryptosystem is the correct one, suppose again that we wish to perform an attack on an 8-bit cryptosystem. Consider, at the same time, an attack directly on 8 bits. This is something that CPA attacks, as opposed to DPA, can attempt without the limitations on multi-bit attacks introduced in previous sections.

From previous sections, we know that:

$$P = C_{eq} \cdot V_{dd}^2 \cdot f_{clk} \cdot \sum_{j=1}^{8} f_j(x, k) \tag{3.21}$$

And, again, assume that each instance of $f_j(x, k)$ is a uniform discrete random variable that can adopt values of either 1 or 0 with equal probability. That is, $P[f(x, k) = 0] = P[f(x, k) = 1] = 1/2$ for every possible plaintext value of x. This condition arises from the consideration that the internal states of the cryptosystem are unbiased and not susceptible to linear cryptanalysis attacks.

The sum of the values of a boolean array or word of bits is known as the Hamming Weight, the number of 1's in such an array. As such, it can be seen that equation 3.21 can be expressed as:

$$P = C_{eq} \cdot V_{dd}^2 \cdot f_{clk} \cdot HW_{f(x,k)} \tag{3.22}$$

And it can be seen that, at the time of evaluation, the power consumed by the cryptographic circuit is linearly dependent on the value of the Hamming Weight function, which depends on the plaintext and the secret key.

Under such circumstances it is possible to calculate the PCC between the power consumed during the different experiments and the Hamming Weight of each experiment.

The PCC becomes a key distinguisher, in that the correct key, in the absence of algorithmic and non-algorithmic noise, will present the highest possible value of the PCC, while all incorrect keys will have lower values. That means that, for a sufficiently large number of experiments, having the correct key would produce Hamming Weight values of the vector of attacked bits that would showcase a perfectly linear correlation with the power consumed. Incorrect keys, however, would present smaller correlation values.

This model is somewhat simplified for power analysis attacks that measure dynamic power, since it bases its power metrics on states, rather than transitions. Transitions between similar states ($1 \rightarrow 1$ or $0 \rightarrow 0$) are expected not to consume power. However, unlike DPA attacks, CPA have the capability to model the effect of transitions appropriately by considering the bit-wise Hamming Distance (HD) between two transitioning states. If the previous state is not known, it must be approximated, such that:

$$P(t) \approx C_{eq} \cdot V_{dd}^2 \cdot f_{clk} \cdot HD(x,k) \tag{3.23}$$

The power model used in CPA attacks can include sources of noise, as described in previous sections. Thus, equation 3.22 can be expressed considering algorithmic and non-algorithmic sources of noise as:

$$P = C_{eq} \cdot V_{dd}^2 \cdot f_{clk} \cdot HW_{f(x,k)} + P_a + P_{na} \tag{3.24}$$

If for every plaintext used sufficient experiments are performed and averaged, the same conclusions can be reached regarding the means of the different sources of noise as described in previous sections:

$$E[P] = C_{eq} \cdot V_{dd}^2 \cdot f_{clk} \cdot HW_{f(x,k)} + E[P_a] + E[P_{na}] \tag{3.25}$$

If the number of experiments is large enough, the expected value of the sources of noise will tend to their mean value, remaining constant across experiments, even for different plaintexts. The Hamming Weight, however, will remain a function of the plaintext and the secret key. Thus, noise can be discriminated simply through trace averaging without the need of subtracting traces, as in the case of DPA attacks. This reduces the magnitude of noise to be averaged, particularly in the case of additive white gaussian noise.

At the same time, CPA attacks can include the effect of state transitions. Thus, they can curtail the emergence of "ghost peaks", facilitating the discrimination of false keys.

Not only that, but CPA multi-bit attacks do not have to discard any traces, given that the Hamming Weight is well structured for all possible plaintexts; that is, assuming all bits of the cryptosystem follow a discrete uniform unbiased distribution, with each bit being independent from each other. Thus, multi-bit attacks are much simpler to execute utilizing CPA than a DPA approach.

Further considerations regarding CPA attacks will be made in subsequent sections, where they are employed as the primary metric to determine the effectiveness of the developed countermeasures.

While a wider variety of statistical methods of analysis to derive the secret key beyond DPA and CPA attacks exist, they won't be detailed here as they are outside the scope of this work. Nonetheless, some further reading can be found here [21]

3.4 Countermeasures to PAA

A countermeasure is any means, be them a software or hardware implementation, that attempts to hinder or outright make it impossible to obtain the secret key of a cryptosystem through the measurement of power traces. What follows is a non-comprehensive list of some of the countermeasures developed in the last two decades. Most of the countermeasures presented will be just summarily mentioned, with appropriate references, as they do not fully intersect with the work developed during this thesis.

Timing Countermeasures

One of the first countermeasures to PAA ever proposed are those that can be categorized as Timing Countermeasures [3]. We have seen in previous sections that both in DPA and CPA attacks, it is important to be able to ascertain the time of evaluation of the functions of interest. In any case, the different power traces obtained during measurements should be temporarily aligned for the attacks to be successful; that is, the attacker should be able to "synchronize" the different power traces so that all the times of evaluation coincide within a single time reference frame. Timing Countermeasures attempt to randomize the time of evaluation by introducing random halting periods of execution or dummy operating instructions. In essence, they introduce noise in the time domain. They are somewhat effective but can be rendered ineffective by increasing the number of samples taken [22].

Masking

One of the most effective countermeasures proposed, that has spurred a great volume of research, are masking countermeasures [7]. Masking countermeasures attempt to decorrelate the data processed by the cryptosystem from the power

consumed by randomly "masking" the evaluation of intermediate variables computed during the encryption process.

We can assume that, at a given time during the evaluation of a Boolean function part of cryptographic primitive, an intermediate variable is to be processed and/or stored in the system.

However, consider a bit-wise mask composed of the minimum amount of bits required to perform masking effectively: two bits.

Before the original intermediate value u_o is stored, a function is applied to it, considering the first bit of the mask (x_{m1}), resulting in a masked value:

$$u_m = f(u_o, x_{m1}) \tag{3.26}$$

Now consider that when the algorithm must present its output, a second function is applied to the masked intermediate value, along with the second bit of the mask (x_{m2}):

$$u_o = g(u_m, x_{m2}) \tag{3.27}$$

The functions and the bits of the mask (x_{m1} and x_{m2}) are chosen such that:

$$u_o = g(f(u_o, x_{m1}), x_{m2}) \tag{3.28}$$

Where the functions $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$ are applied at different moments in time, so that the cryptosystem stores only the masked intermediate variable.

At the same time, the values of x_{m1} and x_{m2} are chosen at random, in a way that makes equation 3.28 hold. Typically, $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$ are equal. That is, they represent the same function: the XOR function. Thus, equation 3.28 is actually:

$$u_o = u_o \oplus x_{m1} \oplus x_{m2} \tag{3.29}$$

Thus, if u_o is equal to 1, $x_{m1} \oplus x_{m2}$ must be equal to zero, for which there are two possible combinations, and if u_o is equal to 0, $x_{m1} \oplus x_{m2}$ must be equal to one; for which there are two further possible combinations.

In the case of a two-bit mask, one bit of the mask is chosen at random during each execution, and the other bit is chosen such that equation 3.29 holds.

The statistical effectiveness of masking countermeasures were initially studied in [23]. Further research in possible masking functions were introduced in [24].

Weaknesses to masking countermeasures were identified in what are called second-order PAA. More on their significance and how to perform such attacks can be seen in [25] [26] [27].

Further studies in masking countermeasures deal with the unexpected results of digital implementation of masked substitution boxes. While theoretically secure on paper, some implementations can lead to transitional glitches that render these implementations insecure. As a response, so called threshold implementations were presented [8].

Power Equalization Techniques

Power equalization techniques attempt to eliminate the disparities between circuit power consumption depending on data processed. Going back to equations 3.17, and 3.18, repeated here for clarity, it can be seen that the information leaked through the power consumed by the cryptosystem depends on the different power consumed (ϵ) when a function evaluates to 1 or 0 (or rather, transitions from $1 \to 0$ or $0 \to 1$).

$$P\Delta(t) = \begin{cases} \epsilon & \text{when } t = \tau \\ 0 & \text{when } t \neq \tau \end{cases} \tag{3.30}$$

$$SNR \approx \frac{\sqrt{N} \cdot \epsilon}{\sqrt{4\sigma_{na}^2 + n \cdot \epsilon^2}} \tag{3.31}$$

Thus, the information derived from the cryptographic attack is directly proportional to the magnitude of ϵ. Then, If it were possible to devise a circuit that would present the same amount of power consumption regardless of the value to which they evaluated, ϵ would approach a value of 0 and the system would cease to leak information.

This is the basis for such countermeasures.

Initially proposed as simply doubling the number of logic elements and complementing the original cryptosytem, such that the complement of a cryptographic primitive were also evaluated, thus evaluating always to both possible outputs, Tiri et al. [28] observed that complementary logic, or Hamming Weight balancing, is insufficient to mask power consumption in such a way.

As such, they proposed a logic style in which every logic cell must first precharge an equal capacitance before evaluating cryptographic functions. Thus, regardless of the logic transitions, the cell would always consume the same amount of power.

This technology, eventually named Dual-Rail pre-charge logic, was further developed in [29] and [30] to potentially reduce the need to tailor-design the logic cells, allowing their implementation with any available logic library.

Other similar approaches exist [31]. However, all such countermeasures present one mayor flaw; namely, intra-die variability of the capacitive elements to be pre-charged can still give rise to asymmetries in power consumed [32], requiring a heavily constrained manual design of the layout to avoid enhancing these variabilities. As such, while these approaches might difficult the performance of an attack, they do not fully protect against one.

Power Disconnection

Power disconnection countermeasures are based on the notion that, if the power injected into the cryptographic circuit is separated from the circuit itself, the attacker cannot directly read the actual power consumed by the circuit, but that of the intermediate power injector.

These countermeasures use voltage converters in some way or another, be them charge-pumps or LDO voltage regulators. Some countermeasures with varying degrees of success can be found in [33] and [34, 35, 36, 37, 38]. In this later series of articles the authors use charge-pumps in distinct phase configurations so as to both introduce noise and attempt to decorrelate the available power readings from the side-channel accessible by the attacker. Special mentions to the charge-pumps configuration presented in [39] and [40], where the authors design a charge-pump scheme such that they reach a perfectly decorrelated side-channel, able to sustain CPA attacks without the disclosure of the secret key.

Noise Enhancers

Noise enhancers are of particular interest to the development of this thesis. Particularly, countermeasures based on Dynamic Voltage/Frequency Scaling (DVFS), firstly introduced in [41].

Dynamic Voltage/Frequency scaling utilizes notions initially derived to optimize power consumption in digital circuits, where depending on the computational load it is possible to choose a pair of values in a power voltage - clock frequency dyad of operational points to save dynamic power (see equation 3.3). This allows saving power when the computational load is not critical by decreasing the operational voltage and frequency, or enhancing the speed of operation by increasing both the supply voltage and the clock frequency [42].

The authors in [41] proposed the use of DVFS as a means to decorrelate the power consumed in cryptographic systems from the data processed by it. This decorrelation is achieved by introducing random noise into the power profile in the form of voltage and frequency variations. In fact, consider again equation 3.3, with two extra components related to these potential variations:

$$P = C_{in} \cdot (V_{dd} + \Delta V)^2 \cdot (f_{clk} + \Delta f) \cdot \alpha, \tag{3.32}$$

where the variables ΔV and Δf are random variables. Under these circumstances, the power traces would present both magnitude and phase noise that would diminish the SNR metrics and, as such, the number of measurements required to successfully perform PAA would increase, as a greater amount of traces would be needed to average the increased noise out.

However, the proposal made in [41] presented some limitations that were addressed in [43]; namely, the original article restricted itself to simulations of the countermeasure that were not originally implemented. At the same time, the authors of [43] noted that, in a system in which the voltage-frequency values were paired, observing the change in frequencies could directly informed the change of voltage values and, as such, the modified voltages could be derived from this information, giving the opportunity to computationally undo the changes.

The authors of [43] then proposed that, instead of coupling the possible voltage-frequency pairs, by fixing the frequency and only allowing the voltage to adopt random values, the countermeasure would be more secure.

Further analysis of the effectiveness of the countermeasure is presented in [44], where the authors derive the minimum increase in traces required to disclose the secret key depending on the number of voltage-frequency pairs available. However, no mention of the functional Dynamic Range of such possible pairs is mentioned. That is, digital circuits will present a floor and possibly a ceiling regarding the minimum (maximum) voltage and frequency, beyond which the circuit loses functionality. This imposes a hard limit on the maximum efficacy of the countermeasure.

3.5 Conclusions

In this chapter we have introduced the notion of Side-Channel Attacks based on the study of power consumption of encryption algorithms during their execution, as well as some notions regarding the use of countermeasures to protect against them.

The conceptual introduction of side-channel states and the leakage of information is formalized through an analogy with finite state machines, in which the state of the cryptosystem informs its internal variables, which present distinct physical magnitudes.

Attacks that rely on obtaining this information make use of statistical methods, viable given the probability distributions that are imposed on the cryptosystem during its design. DPA was one of the first proposed methods derived to perform power analysis attacks. They rely on the adequate partitioning of the power traces measured according to guesses on the secret key.

While powerful, DPA techniques present some issues regarding the correct identification of the secret key. Given the assumptions made regarding temporal and algorithmic correlations, which the technique cannot address, the identification of the secret key can be relatively obscured given the presence of ghost peaks. At the same time, DPA attacks on multiple bits at once can become complex, as the correct partinioning must be coupled with the appropriate discarding of traces that cannot be used.

Correlation Power analysis, on the other hand, can address all of these issues and thus become a somewhat more powerful and simpler key distinguisher. As such, during the development of this thesis they will be considered the golden-standard and be used accordingly.

Chapter 4

Leakage Power Analysis Attacks, Countermeasures & FDSOI technology

4.1 Technology Scaling and Leakage Currents as a Side-Channel

While until recently most of the power consumed by CMOS digital circuits has stemmed from the current surged experienced by circuit transitions, the need for ever more complex systems and designs, with an ever increasing growth in the number of transistors present in a given die has forced the pursue of transistors of smaller size.

With transistors manufactured in smaller nodes it is possible to manufacture systems of greater complexity within the same or smaller dies. However, this reduction in transistor size, when coupled with an increased population of CMOS devices is translated in a steep increase in the power consumption incurred by these newer systems, were everything to remain as is.

It is clear that the dynamic power consumption of a tightly packed design is going to be proportional the number of digital elements that see their state altered from one period to the other. We can then quote, yet again, equation 3.3, with a certain modification to account for the number of such elements present:

$$P \propto N \cdot C_{in} \cdot V_{dd}^2 \cdot f_{clk} \cdot \alpha, \tag{4.1}$$

where N represents the approximate total number of digital elements.

In the interest of efficiency, one would desire to decrease the total power consumption to be as small as possible, without having to sacrifice the minimum number of digital elements required to maintain the design within its desired functionalities.

As such, two possibilities arise. Either the clock frequency is decreased, or the magnitude of the voltage source is decreased, with the possibility of combining both options as has been seen in in the case of DVFS.

It might not be wholly desirable to decrease the clock frequency much, under the presumption that an increase transistor count implies a design that envisions greater computational complexity, which would be further penalized by a slower clock.

At the same time, the power consumption is directly proportional to the square of the voltage source, so any reduction in voltage would significantly impact the reduction on power consumption (though with diminishing returns).

As transistors decrease in length, it is also necessary to reduce the magnitude of the voltage source, not only for power saving reasons, but because of functionality. Given that shorter channels endure higher electric fields, it is necessary to decrease the voltage seen at the drain so as to not surpass the breakdown point.

The ability of CMOS transistors to source or conduct current is proportional to powers of their overdrive voltage ($V_{gs} - V_{th}$). In digital circuits, particularly, the gate-source voltage can be directly considered the input voltage V_{dd}. Thus, a reduction in the magnitude of the voltage source will heavily impact the maximum frequency and even the functionality of transistors unless the threshold voltage is reduced. That is, unless the threshold voltage is reduced at the manufacturing level, the overdrive voltage decreases with V_{dd}.

However, both the reduction of the threshold voltage, as well as a reduction in channel length, present unintended consequences. Chief among which, is the increase in static power consumption, produced through an increase in leakage current, specially those that arise between drain and source.

Thus, as transistors progress further into nanometric scales, the static or leakage power consumption of circuits has increased, becoming a significant portion of the overall power consumption of CMOS circuits.

Because of this increase in the impact of leakage power in the overall power consumption of a circuit, it became of interest to explore whether static power inadvertently leaks cryptographic information and whether it can be utilized as a side-channel to perform power analysis attacks.

Initial works were produced by different authors exploiting mainly the leakage consumption of combinational logic gates with different input vectors [45], [46], [47]. In 2010, Alioto et al. [9] observed that sequential logic elements (that is, flip-flops), exhibited distinct leakage current profiles depending on the data they were storing (either a 1 or a 0) and, with that, introduced a power model of leakage power consumption of register arrays depending on intermediate cryptographic variables.

That is, consider n flip-flops that store intermediate variables of a cryptographic function. Assuming, for the moment, that there is no variability to take into account, flip-flops that store a 1 consume, when in steady-state, a leakage power of I_1. Alternatively, flip-flops that store a 0 consume a magnitude I_0 of leakage power. We define the Hamming Weight (HW) of the register array as the total amount of 1's present or stored in the array. Thus, the total leakage

consumption of such an array can be expressed as the sum of the leakage current of flip-flops that store a 1, and the sum of leakage current of flip-flops that store a 0.

$$I_{leak} = n \cdot I_0 + HW \cdot (I_1 - I_0) \tag{4.2}$$

Defining $(I_1 - I_0) = \epsilon$, the difference between the leakage current consumed by flip-flops storing a 1 and a 0 we finally get the expression that serves as a power model for leakage power analysis attacks on register slices:

$$I_{leak} = n \cdot I_0 + \epsilon \cdot HW \tag{4.3}$$

This power model, as opposed to the power models based on dynamic power consumption, require certain assumptions that somewhat constraint the analysis. Firstly, we must assume, and it is shown both in [9] and in [48] that, contrary to Dynamic power consumption, leakage current necessitates of a certain settling time before the circuit can reach its steady-state, at which point measurements will be stable. This implies that the attacker must hope that the standard-cell components of the technological node being attacked either settle before the next clock edge, or a stronger attacker must be presumed, one which is able to control the clock.

At the same time, the intensity of the signal of interest, namely, ϵ, is much smaller than those of dynamic power, in which the current that surges to charge input and parasitic capacitances are of much higher intensity. This implies that, under the same noise conditions, several additional experiments are required to be able to average the noise out and obtain significant information.

The coupling of these two facts (control over the clock or a slow clock, and a higher number of required samples) make the performance of leakage power analysis attacks somewhat more cumbersome.

Other elements to take into consideration are the fact that leakage currents of CMOS circuits are particularly susceptible to variations in temperature.

Assume, for the moment, that the component I_0 is a function of temperature, while the term ϵ remains somewhat constant to temperature variations. Fluctuations in temperature given environmental factors and heat dissipation within the circuit would introduce a level of randomness into equation 4.3.

$$I_{leak}(T) = n \cdot I_0(T) + \epsilon \cdot HW \tag{4.4}$$

As such, the magnitude I_0 would become a random variable which would introduce a certain biased noise represented by a variance $\sigma^2_{I_0}$. Thus, this further noise would either need to be averaged out or the experiments should be performed under thermally controlled conditions. All these considerations have been addressed in [49] and [50].

In reality, though, the absolute difference between $|\epsilon| = I_1 - I_0$ does not remain constant with increasing temperatures. It has been seen [51] that there is an exponential increase in the signal intensity with increasing temperatures,

while noise has been observed to increase, mostly, linearly. Thus, attacks performed under thermally controlled conditions at high temperatures (e.g., $80°C$) require a smaller amount of traces to perform successful attacks.

Another element to take into account is the fact that measurements of leakage current consumption are to be performed with a DC coupled oscilloscope or measurement instrument. Dynamic Power consumption, given its rapid variation or transitions, can be easily AC coupled without much loss of information at lower frequencies. Thus, much of the lower frequency noise components that arise within the cryptosystem itself and the measurement systems are filtered away, reducing the overall noise floor of the measurements. Since this process cannot be performed for stable measurements of leakage current, a DC coupled measurement is required, accumulating the entire noise spectrum. Because of this, the noise floor is going to be higher. As such, measurements of leakage power consumption are not only smaller in signal intensity, but noise tends to be higher [51]

Nonetheless, power analysis attacks that utilize leakage current as a side channel present some advantages against certain implementations of masking countermeasures in which the masks are simultaneously loaded onto register arrays.

That is, consider a two-share mask that follows one of two possible equations: $b \oplus m_1 \oplus m_2 = b$ or $S(b \oplus m_1) \oplus m_2 = S(b)$, with $S(\cdot)$ being a substitution function, and b, m_1, and m_2 being single bits (see Section 3.4).

Suppose that the masked plaintext $(b \oplus m_1)$ and the second share of the mask (m_2) are loaded simultaneously on two distinct registers.

We have previously seen that at least one of the mask bits is random, and the other bit must evaluate to a value that satisfies the equalities presented above. We are going to assume that the equation that must be satisfied responds to $b \oplus m_1 \oplus m_2 = b$.

We are interested in evaluating the Hamming Weight of the following:

$$HW(b, m_1, m_2) = (b \oplus m_1) + m_2 \tag{4.5}$$

Thus, in the case where $b = 1$, we have:

$$\begin{aligned} HW(b, m_1, m_2) &= 1 \oplus 0 + 0 = 1 \\ HW(b, m_1, m_2) &= 1 \oplus 1 + 1 = 1 \end{aligned} \tag{4.6}$$

And in the case where $b = 0$ we have:

$$\begin{aligned} HW(b, m_1, m_2) &= 0 \oplus 0 + 0 = 0 \\ HW(b, m_1, m_2) &= 0 \oplus 1 + 1 = 2 \end{aligned} \tag{4.7}$$

Assuming that m_1 evaluates with a discrete uniform equiprobable distribution ($P[m_1 = 1] = P[m_1 = 0] = 1/2$), it can be seen that the expected value of the Hamming Weight when $b = 1$ is:

$$E[HW|b=1]=1 \tag{4.8}$$

At the same time, the expected value of the HW when $b=0$ is:

$$E[HW|b=0]=1 \tag{4.9}$$

So, in principle, on multiple experiments where noise averaging is applied, the Hamming Weights would be indistinguishable from one another and no information would be obtained.

However, if the leakage current magnitudes were squared, the expected value of the squared hamming weight would change. That is:

$$\begin{aligned} E[HW^2|b=1]&=1 \\ E[HW^2|b=0]&=2 \end{aligned} \tag{4.10}$$

Thus, for an attack on n bits, the squaring of the leakage current measured at time of evaluation and, as a consequence the squaring of the HW, linearly transforms the contribution of the Hamming Weight to the overall power consumption when multiple measurements are averaged. That is:

$$E[HW^2]=HW+2\cdot(n-HW), \tag{4.11}$$

where the first term (HW) represents the contribution of all bits of the n-bit register that evaluate to $b=1$, and the second term ($2\cdot(n-HW)$) is the contribution of all bits of the register where $b=0$.

Thus, by squaring the magnitude of the leakage current it is possible to attack masking implementations in a simpler fashion than what dynamic power analysis attacks allow [52][49][53].

It has also been observed that it is possible to synergically perform leakage and dynamic power analysis attacks to extract the secret key [54].

4.2 Countermeasures to LPA attacks

Countermeasures to Leakage Power Analysis Attacks can be again categorized according to their effect. Thus, as Power Equalization Techniques we have articles by Zhu et al., [55], where the authors attempt to design logic cells that consume the same amount of leakage current regardless of the input vector or state in which the cell finds itself. Similar approaches have more recently been made by Fadaeinia et al. in [56] and [57].

Another category of countermeasures is that of Noise Enhancers. Zhu et al. proposed the use of Ring Oscillators to introduce random, uncorrelated leakage current noise into the circuit, decreasing the SNR by increasing the noise floor [58]. Yu et al, on the other hand, have proposed the use of their charge pump schemes as a source of uncorrelated noise [59].

Another proposal by Yu et al. is to introduce variations into the power supply voltage in a way that correlates to a potential false key. Since leakage currents are exponentially dependent on the source voltage value, it is possible to alter the leakage current profile by slightly modifying the voltage. Doing it in a way, such that the increase/decrease of the voltage magnitude is linearly related to the evaluation of a false secret key plus plaintext, the attacker can be dumbfounded into extracting a false key [60]. They called this approach False key-controlled aggressive voltage scaling (FKCAVS).

However, in [61] the authors demonstrate that by using principal component analysis of the correlation figures obtained during an attack on circuits implementing FKCAVS, it was possible to undo the effect of the false key on the correlation parameters, obtaining the correct secret key.

4.3 FDSOI Technology in Cryptosystems

At the beginning of this chapter several potential effects of transistors' node shortening have been described, and how short channel effects can impact and increase the leakage current consumption of CMOS technology presented.

Fully Depleted Silicon-On-Insulator (FDSOI) technology, given their structure, curtails some of the short-channel effects, in particular Drain-Induced Barrier Lowering (DIBL) and accompanying punchthrough effects [62]. The proximity of the Drain and Source in shorter nodes, as well as their extension towards the bulk, facilitates the formation of a depletion region for each junction too large for comfort. If the channel is not properly doped [63], applying a voltage to the drain can create a wide enough depletion region in both of the junctions such that they overlap, forming a path for current flow regardless of the bias applied at the gate. This phenomena is dependent on both the effective channel length and the depth of the drain and source junctions [10].

FDSOI technology solves some of these issues by limiting the depth achievable by the drain and source junctions. The effective bulk and the channel of the transistor are separated by a thin layer of insulating material that limits the maximum depth of the channel and, with it, that of the junctions (Fig. 4.1).

At the same time, since the bulk and the channel are electrically isolated, a wider range of bulk biasing can be used to influence the threshold voltage of CMOS transistors without impacting circuit functionality.

Subthreshold currents are heavily influenced by the threshold voltage value of their transistor. In fact, it has been seen [10] that the threshold voltage of FDSOI transistors varies linearly with the application of a body bias. Thus, it is to be expected that the static current consumption of FDSOI transistors is an exponential function of body bias:

$$\begin{aligned} V_{th} &= V_{th0} + k \cdot V_{bb} \\ I_{leak} &\propto e^{(b \cdot V_{th}(V_{bb}))} \end{aligned} \tag{4.12}$$

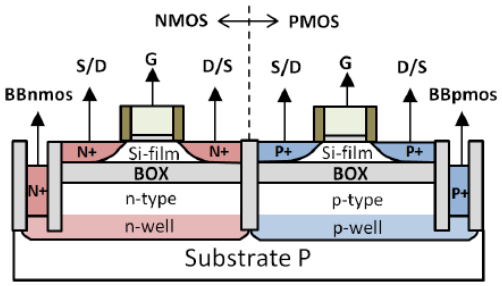

Figure 4.1: Cross-sectional view of a flip-well CMOS FDSOI pair (Figure extracted from [1]).

Where k and b are parameter constants.

FDSOI technology as presently conceived presents two types of transistors depending on the doping of their wells. Flip-well transistors, as depicted in Fig. 4.1, see their threshold voltage decreased with increasing absolute values of body bias. Flip-Well, NMOS transistors can be biased positively, while flip-well PMOS transistors require a negative voltage applied at their bulk terminal if bias is to be applied. This is of particular interest for high-performance, low-power applications, where a lower voltage supply can be offset by lowering the threshold voltage as required simply by charging the bulk region of the transistors of interest. This type of body biasing is typically referred as forward body bias.

However, transistors without flip-wells, in which the bulk of N-transistors is P-doped, and that of P-transistors are N-doped, behave the other way around. That is, the polarities are reversed: NMOS bulk is negatively biased, PMOS bulk is positively biased, and increasing the absolute value of body bias increases the threshold voltage. While this biasing might decrease the current sourcing capabilities of transistors and, as a consequence, decrease the gain of analog systems and maximum operational frequency of digital implementations, low-latency, portable or remote circuits might benefit greatly from it in stand-by modes to extend their life cycle.

With this, it would be possible to modify the static leakage profile of cryptosystems implemented in FDSOI technology through the modification of their body bias. In order to bias FDSOI transistor wells, a back bias generator is required. A back-bias generator is any device or implementation that can generate a range of desired voltages at the back-gates of FDSOI transistors, which are modelled as capacitive loads in series with a generally small current source.

This would require both a body bias scheme and a circuit capable of driving the body bias wells. Various possible implementations have been proposed as body bias generating circuits:

- Push-pull networks [64][65] [66]

- Charge-pumps [67] [66][1]
- OTAs, either in buffer configuration or as a comparator [68] [69] [67]

Where in all the above references, a charge-pump was utilized to generate the negative body bias.

Thus, FDSOI technologies present an opportunity to protect against LPA attacks.

This thesis then seeks to explore how the employ of FDSOI technology, with its capability to dynamically alter the leakage profile of CMOS circuits, can be used to thwart leakage power analysis attacks. To the best of the author's knowledge, at the beginning of this thesis only one such approach had been attempted [70].

Chapter 5

Random Body Bias as a countermeasure to Leakage Power Analysis Attacks

In the previous chapters we have offered a brief introduction into the fundamentals of cryptography, power consumption of cryptographic circuits as a side-channel, and the basic notions of Power Analysis Attacks, as well as a brief introduction into leakage currents and FDSOI technology. The totality of these concepts should prove enough of a stepping stone to develop an understanding on how body biasing schemes could be used as a countermeasure against Leakage Power Analysis Attacks.

In this chapter, we first explore the relationship between leakage current of standard cell flip flops, the bit value they store, and the body bias they present. We do so through electrical simulations that allow us to gain insight into the circuits' working, enough to be able to formalize an analysis and develop a statistical power model.

With this, a first countermeasure to LPAA is presented, mathematically modelled, and its efficiency tested against montecarlo simulations of virtual Correlation Power Analysis Attacks. The countermeasure is based on the random application of a body bias value at the beginning of each encryption process.

5.1 Initial Investigation & Modelling

We begin our approach in attempting to establish an understanding on the effect of body bias on the leakage current profile of flip-flops. Thus, we forego the exploration of countermeasures tailored for attacks on combinational logic. However, it is important to note that much of the countermeasures developed during this thesis could also be applied to such circuits.

The first step is attempting to obtain a sufficiently accurate model of how

the leakage current of flip-flops behaves as a function of body bias. In order to do so, a test-bench circuit is implemented in the Virtuoso Analog Design Environment (ADE).

Two D-flip-flops, in a 28nm FDSOI technology are used as the Devices Under Test (DUT). The devices are powered at $V_{dd} = 1$ V, the nominal voltage for the technology. The D input of the two devices is each connected to an ideal voltage source, V_{in1}, and V_{in0}, with constant voltage values equal to 1 V, and 0 V respectively.

Each flip-flop is also connected to a clock source, designed to produce a single clock pulse. After the clock pulse, one flip-flop will store a logic value of 1, while the other will store a logic value of 0.

In addition, two ideal voltage sources are used to implement the body bias well drivers. The particular flip-flops initially studied are forward body bias, flip-well transistors (Fig. 4.1), which means that N-transistors are biased with a positive voltage (V_{bbn}), while P-Transistors are biased with a negative voltage (V_{bbp}). Each CMOS logic cell has, then, two terminals through which the body bias can be modified.

The reset terminal of each of the flip-flop is connected to an ideal voltage source tying it to the inactive state. The output of the flip-flops is connected to ideal capacitors of 3 fF, a figure in accordance to a low fan out in the technology.

The simulation is configured such that the current flowing onto each flip-flop from its voltage source (V_{dd}) is read after a single clock pulse takes places, loading the appropriate value onte the register.

A parametric sweep is performed on top of the transient simulation, altering the body bias value. In this initial assessment, the positive and negative body bias values are kept equal in the absolute sense. That is, $V_{bbn} = -V_{bbp} = |V_{bb}|$. This is done partly to simplify the analysis and partly to guarantee that each driving cell has equal pull-up and pull-down capabilities. After each simulation, the values of I_0 and I_1 are stored, and ϵ is calculated as $\epsilon = I_1 - I_0$.

The resulting curves can be seen in Fig. 5.1

It can be seen that all three curves are monotonically increasing functions of $|V_{bb}|$, with $I_1(V_{bb}) > I_0(V_{bb})$ for all possible values of body bias within the range.

Recall from Chapter 4 and [9] that the total leakage current consumed by a register array of n bits can be expressed as a function of I_1, I_0, ϵ, and the HW of the bits stored in the array.

$$I_{leak}(HW) = n \cdot I_0 + \epsilon \cdot HW \tag{5.1}$$

From the curves of Fig. 5.1 it can be seen that the total leakage current consumed by a register array (Equation 4.3), initially a linear univariate function of the Hamming Weight, now becomes a bivariate function of both the HW and the Body Bias as well. For an initial assessment, we consider a symmetric body bias, where $V_{bbn} = -V_{bbp} = |V_{bb}|$.

$$I_{leak}(HW, |V_{bb}|) = n \cdot I_0(|V_{bb}|) + \epsilon(|V_{bb}|) \cdot HW \tag{5.2}$$

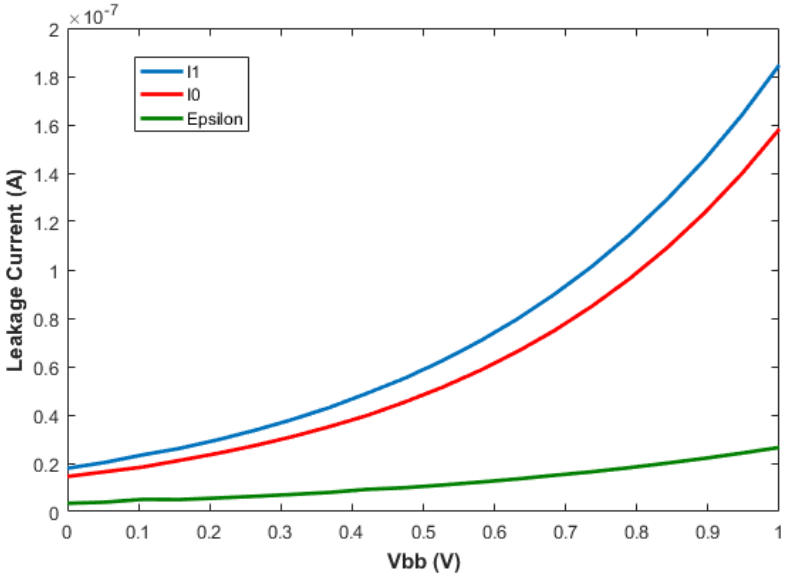

Figure 5.1: Leakage current consumption of a standard-cell D flip-flops as a function of absolute body bias. While this is just an example of a particular flip-flop, it is representative of the behaviour observed.

5.2 CPA on Dummy Cryptosystem. Empirical approach

We wish to explore the effect that a varying body bias has on the capability of the Pearson Correlation Coefficient to distinguish between the correct key and incorrect keys in a cryptosystem.

As such, a dummy cryptosystem is implemented in the Virtuoso ADE. The cryptosystem is formed by an 8-bit register array that stores the result of a dummy encryptying function. The encryptying function is formed by the bit-wise XORing between the plaintext, an 8-bit array, and the secret key, a fixed, deterministic 8-bit array (Fig. 5.2).

We base this dummy cryptotystem in the fashion of what is to be expected for typical block ciphers such as the AES [71]. Generalizing block ciphers, we expect some kind of key encryption of the plaintext (represented by the XOR function), followed by a non-linear substitution process. This non-linear substitution process, typically referred to as an S-Box, can be omitted in our assessment, given that their results are isomorphic to a vector of all the XORed plaintexts for a given key. At the same time, S-Boxes are generally implemented either through LUT's or combinational logic, which we are at the moment not interested in studying.

As it stands, $|V_{bb}|$ is a continuous function. However, in order to realistically

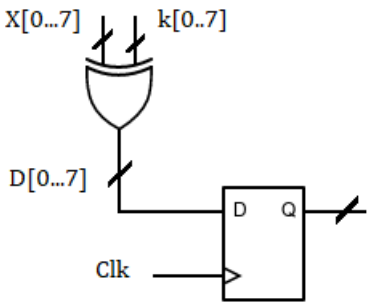

Figure 5.2: Dummy cryptosystem implementing a bitwise XORing function between the plaintext ($X[0..7]$) and a secret key ($k[0..7]$)

modify the body bias of the cryptosystem, we have to take into consideration the type of available circuits that can drive the wells of the back gates of transistors. We are going to work under the assumption that, whatever the body bias generator employed to reach a fixed value of body bias, be it a DAC, a linear regulator or a Charge Pump, the circuit can only modify the body bias in discrete increments of voltage. We could also expect that these discrete increments of body bias, which we will call ΔV_{bb}, can present some noise that make the circuit slightly deviate from its expected voltage goal. This will, however, not be taken into consideration in the following analysis.

In this initial idealized approach, the body bias generator is emulated with the circuit depicted in Fig. 5.3.

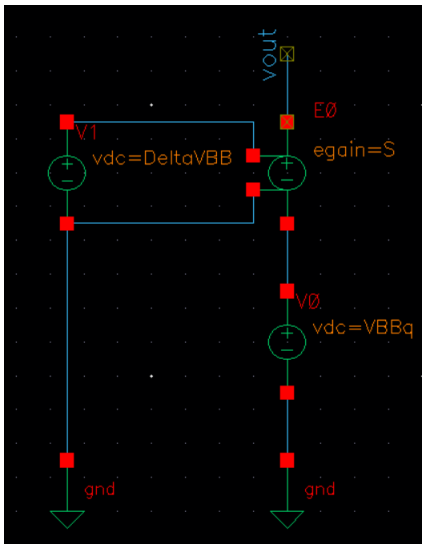

Figure 5.3: Idealized discrete body bias generator

The function implemented by this circuit can be expressed as:

$$V_{bb}(S) = VBB_q + S \cdot \Delta V_{bb} \tag{5.3}$$

Where VBB_q represents the quiescent point of the body bias, ΔV_{bb} the step increment with which the body bias can change, and S is discrete random variable that can only adopt integer values, determining the total increment of body bias with respect from VBB_q.

This produces a positive body bias for NMOS transistors. The negative body bias for PMOS transistors is generated by multiplying the resulting $V_{bb}(S)$ with a gain of -1 with an ideal voltage controlled voltage source.

For an initial approach, VBB_q is set at 0 V, ΔV_{bb} at 20 mV, and S is made to adopt any integer value that guarantees functionality within the limits of these particular FDSOI technology. The minimum and maximum absolute values of body bias in this technology are, respectively, $Vbb_{min} = 0$ V and $Vbb_{MAX} = 1$ V. Thus, under these conditions, S can adopt any integer value between $s \in [\frac{Vbb_{min}}{\Delta V_{bb}}, \frac{Vbb_{MAX}}{\Delta V_{bb}}] = [0, 50]$, with a probability of $P[S = s] = \frac{1}{51}$ for any s.

In order to be able to perform a virtual Correlation Power Analysis Attack with leakage current as a side-channel on the dummy cryptosystem depicted in Fig. 5.2, the following steps are taken:

- The dummy cryptosystem is implemented in the Virtuoso ADE environment under a transient simulation. Similarly to the testbench used to obtain the curves in 5.1, a vector of input plaintexts $X[0..7]$ is set at the beginning of the simulation. A clock pulse then loads the resulting XORing between the plaintext and the secret key into the register array. After some time, the leakage current of all of the flip-flops is noted.

- With 8 bits, there are 256 possible plaintexts, ranging from 0 to 255 in decimal notation. Thus, 256 transient simulations are run, one for each possible plaintext.

- The secret key is set to 170_{10} and kept the same during all the experiments.

- At the end of the 256 experiments, a vector containing 256 leakage current values and the plaintext used is obtained, one for each run. This leakage current vector represents the magnitude of the leakage current consumed exclusively by the registers. Neither the XOR gates, the inverters connected to the clock, or any other element is considered, thus obtaining a representation of a system that is noiseless.

- A Hamming Weight matrix is produced in Matlab. Each entry in a column is the result of calculating the Hamming Weight of the 8-bits resulting from the XORing of the plaintext i (representing the rows) and a prospective secret key j (representing the columns). Since there are 256 plaintexts and, with 8 bits, 256 possible keys, the matrix is a square matrix of dimension

256×256. Each entry of the matrix is represented by the following function, where each column represents the HW between the 256 plaintexts and a prospective secret key:

$$a_{i,j} = HW(i \oplus j) \tag{5.4}$$

- The Pearson Correlation coefficient (PCC) is calculated between the vector of leakage currents obtained from the 256 experiments and each column of the HW matrix.
- This procedure is repeated two times. In one set of experiments, the body bias is kept constant, at 0 V during the 256 plaintext experiments. In the other set of 256 transient runs, at the beginning of each simulation a random value is chosen for S, such that the body bias becomes a random variable. The value of S is chosen at random following a discrete uniform probability distribution with the probabilities described above, within the interval of possible values of $[0, 50]$.

The correlation between the vector of leakage currents and the vectors of Hamming Weight dependent on the secret key are calculated utilizing the PCC in Matlab. The PCC can be calculated analytically using the expression:

$$\rho_{I_{leak},HW} = \frac{Cov(I_{leak}, HW)}{\sqrt{Var(I_{leak})} \cdot \sqrt{Var(HW)}} \tag{5.5}$$

Where:

$$Cov(I_{leak}, HW) = E[I_{leak} \cdot HW] - E[I_{leak}] \cdot E[HW] \tag{5.6}$$

The use of this analytical expression is valid when the probability distribution functions of the random variables in play are known. At this point, however, are not analytically deriving their distributions, but using simulated samples. In this case, the sample correlation is calculated as:

$$\rho_{I_leak,HW} = \frac{1}{N-1} \sum_{i=0}^{N} \frac{(I_{leak_i} - \hat{I}_{leak})}{\sqrt{Var(I_{leak})}} \cdot \frac{(HW_i - \hat{HW})}{\sqrt{Var(HW)}} \tag{5.7}$$

Where the variances are also calculated from the samples.

The results for the PCC between the leakage current and the HW matrix for the case in which the body bias is kept constant, and for the case in which the body bias is randomized can be seen in Figs. 5.4 and 5.5.

It can be seen that, when no random body bias is applied, the secret key can be readily identified. However, when a random body bias is applied, not only does the maximum PCC decrease significantly, the correct secret key cannot be properly determined.

Two conclusions can be extracted from the results depicted in the figures 5.4 and 5.4. From Fig. 5.4 it can be seen that, when the register array is analyzed under no noise conditions, without algorithmic noise introduced by bits not

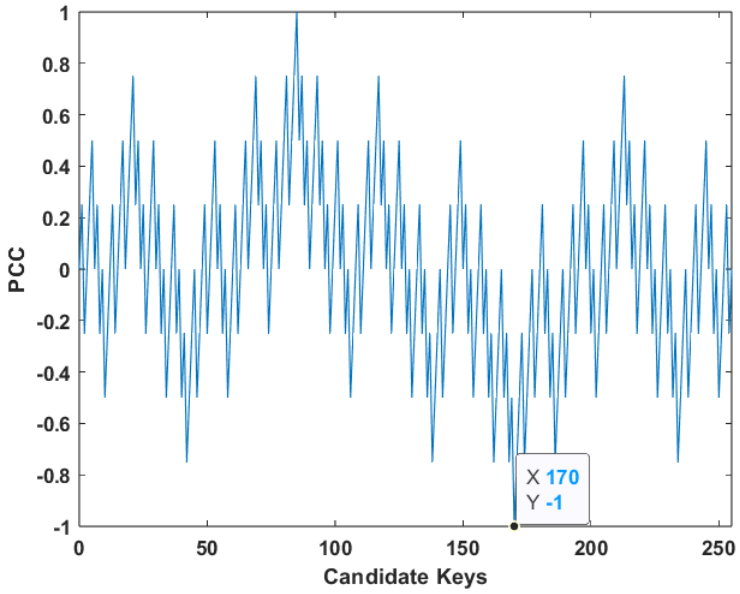

Figure 5.4: PCC between the leakage current of a register array and the HW calculated with all possible keys for a fixed body bias value $|V_{bbn}| = 0$ V

pertinent to the attack, or non-algorithmic noise (the only current evaluated is that of the registers, and there are no other sources of noise), there is a perfect correlation ($PCC = 1$) between the Hamming Weight and the leakage current consumed by the array. This means that the leakage current of the register array can be perfectly expressed as a linear function of the Hamming Weight:

$$PCC = 1 \rightarrow I_{leak}(HW) = a + b \cdot HW \tag{5.8}$$

This result is consistent with the expected behaviour of an ideal register array, as informed by equation 4.3:

$$I_{leak}(HW) = n \cdot I_0 + \epsilon \cdot HW \tag{5.9}$$

On the other hand, Fig. 5.5 indicates that this perfect correlation is severely diminished, to the point where the secret key cannot be identified.

Two other factors arise from these simulations. The first one is that the results presented in Fig. 5.5 stem from a single experiment of 256 transients. S, on the other hand, can adopt 51 distinct values. Being a random variable, this number is too small to guarantee a representative distribution of S. Thus, the results shown in Fig. 5.5 are only one of many possible realizations.

Another factor is that empirical results would not allow us to readily predict the influence of different parameters (VBB_q, ΔV_{bb}, and S) on the ability of a

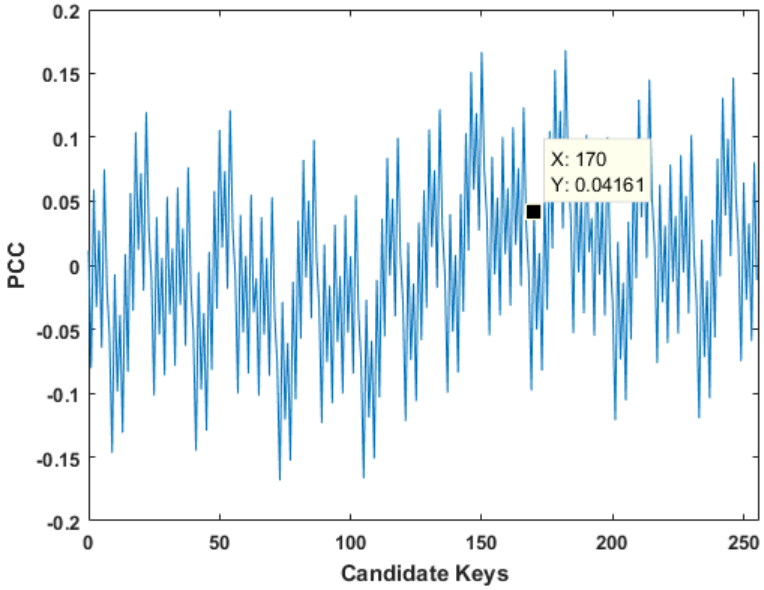

Figure 5.5: PCC between the leakage current of a register array and the HW calculated with for all possible keys, in the presence of a random body bias

potential random body bias countermeasure to hinder the identification of the secret key. At the same time, despite the dummy cryptosystem being a simple circuit, each run of 256 experiments takes a long time to simulate, which would make the realisation of multiple electrical simulation unfeasible in the pursue of trends.

Thus, in order to obtain a model of the effects of the different parameters on the ability of a random body bias to hinder the key acquisition we cannot rely exclusively on empirical results. Given that when appropriately evaluated, the leakage currents of the simulated circuits present a perfect correlation, numerical models would, in principle, present a one-to-one correspondence with electrical simulations.

5.3 Countermeasure Modelling

5.3.1 PCC without countermeasure

In order to determine the effect of the countermeasure, we first need to establish analytically the value of the PCC observed when no countermeasure is applied.

Consider equation 4.3 with no application of a random body bias.

We wish to calculate the PCC between the leakage current and the hamming weight assuming that, without a random body bias, both I_0 and ϵ are constants.

The PCC is calculated using equation 5.6 assuming that the correct secret key is chosen. Thus, we expect the secret key to present the highest possible value of the PCC.

The distribution of the Hamming Weight arises from the fact that the plaintext in the dummy cryptosystem is chosen at random. This is not exactly the case when an exhaustive simulation is considered, where all the possible values of the plaintext are used, but considering all 256 possible values of the plaintext, each bit ends up having a well defined distribution.

That is, consider a vector of 8 bits representing the plaintext of the dummy cryptosystem, $[x_0, x_1, ..., x_7]$, with x_0 being the Least Significant Bit (LSB). In an exhaustive simulation, all 2^8 possible values are considered. The probability distribution of each bit can be calculated as:

$$P[x_i = 1] = P[x_i = 0] = \frac{2^{8-1}}{2^8} = \frac{1}{2} \qquad \forall i, 0 \leq i \leq 7 \tag{5.10}$$

That is, each bit has an unbiased uniform discrete probability distribution. We assume, at the same time, that each bit of the plaintext is independent from all others.

The results stored in the register array arise from the XORing of the plaintext with the secret key. However, the XOR operation between a random variable with equiprobable distribution (a bit of the plaintext) with a constant (a bit of the secret key) does not alter the probability distribution of the output.

Thus, we can define the Hamming Weight as a random variable representing the sum of all bits of the plaintext for a given experiment:

$$HW = x_0 + x_1 + ... + x_7 \tag{5.11}$$

Given the linearity of expectation, we can calculate the expected value of the Hamming Weight as:

$$E[HW] = E[x_0 + x_1 + ... + x_7] = E[x_0] + E[x_1] + ... + E[x_7] \tag{5.12}$$

Since each bit is independent from each other and follows an unbiased uniform probability distribution, we have:

$$E[HW] = \sum_{i=0}^{7} E[x_i] = \sum_{i=0}^{7} 1 \cdot P[x_i = 1] + 0 \cdot P[x_i = 0] \tag{5.13}$$

And with these, the expected value of the Hamming Weight for an n-bit array of unbiased uniformly distributed bits is:

$$E[HW_n] = \frac{n}{2} \tag{5.14}$$

The variance of the Hamming Weight can be similarly calculated. Under the assumption that each bit of the plaintext is independent from all the others, we have that:

$$\begin{gathered} Cov(x_i, x_j) = 0 \\ \forall i, j, i \neq j, 0 \leq i, j \leq 7 \end{gathered} \tag{5.15}$$

Thus:

$$Var(HW) = Var(x_0 + x_1 + ... + x_7) = Var(x_0) + Var(x_1) + ... + Var(x_7) \tag{5.16}$$

For an n-bit register array it can be shown:

$$Var(HW_n) = \sum_{i=0}^{n-1} Var(x_i) = \frac{n}{4} \tag{5.17}$$

With these, we have $E[HW]$ and $Var(HW)$. To calculate the PCC we next need to compute $E[I_{leak} \cdot HW]$:

$$\begin{aligned} E[I_{leak} \cdot HW] &= E[(n \cdot I_0 + \epsilon \cdot HW) \cdot HW] = \\ & E[n \cdot I_0 \cdot HW + \epsilon \cdot HW^2] = \\ & E[n \cdot I_0 \cdot HW] + E[\epsilon \cdot HW^2] \end{aligned} \tag{5.18}$$

Given that under no countermeasure conditions n, I_0, and ϵ are constants:

$$E[I_{leak} \cdot HW] = n \cdot I_0 \cdot E[HW] + \epsilon \cdot E[HW^2] \tag{5.19}$$

We now compute $E[I_{leak}] \cdot E[HW]$, which given the above derivations can be shown to be:

$$E[I_{leak}] \cdot E[HW] = n \cdot I_0 \cdot E[HW] + \epsilon \cdot E[HW]^2 \tag{5.20}$$

Finally, substituting equations 5.19 and 5.20 in the expression of the covariance (Equation 5.6), we arrive at:

$$Cov(I_{leak}, HW) = \epsilon \cdot (E[HW^2] - E[HW]^2) \tag{5.21}$$

And noting that:

$$E[HW^2] - E[HW]^2 = Var(HW) = \sigma_{HW}^2 \tag{5.22}$$

$$Cov(I_{leak}, HW) = \epsilon \cdot \sigma_{HW}^2 \tag{5.23}$$

The variance of I_{leak} can be easily calculated as:

$$Var(I_{leak}) = Var(n \cdot I_0 + \epsilon \cdot HW) = \epsilon^2 \cdot \sigma^2_{HW} \tag{5.24}$$

All these lends itself to the calculation of the PCC between the Leakage Current of the dummy cryptosystem and the Hamming Weight.

$$\rho_{I_{leak},HW} = \frac{\epsilon \cdot \sigma^2_{HW}}{\sqrt{\epsilon^2 \cdot \sigma^2_{HW}} \cdot \sqrt{\sigma^2_{HW}}} = 1 \tag{5.25}$$

This result is consistent with that of Fig. 5.4 and makes evident the fact that, under noiseless, ideal conditions the leakage current of a register array is a linear function of the Hamming Weight, if the correct key is chosen. All other incorrect keys will present a smaller value of the PCC. In fact, it can be shown ([20]) that, with an attack on an n-bit key, with n' correct key bits and m incorrect key bits, the PCC between the leakage current consumption of a register array and the Hamming Weight calculated with this partially incorrect key is:

$$\rho_{Ileak,HW_{wrong}} = \rho_{Ileak,HW_{correct}} \cdot \frac{(n' - m)}{n} \tag{5.26}$$
$$n = n' + m$$

This is also consistent with Fig. 5.4.

Thus, the PCC serves as a discriminant between the correct secret key and the rest. As such, we will evaluate the effect of the countermeasure on the resulting PCC.

5.3.2 Random body bias scheme

We would like to determine the effect of using a random value of body bias as countermeasure on the PCC, which we are employing as a key discriminant.

In order to do that, we need to be able to parameterize the magnitudes of I_0 and ϵ as a function of body bias. Once we have obtained such a relation, we can express I_0 and ϵ as functions of a random variable (i.e. V_{bb})

We begin by obtaining an expression of the different variables of interest (I_1, I_0, and ϵ) as a function of body bias (Fig. 5.1). Each of the leakage current curves obtained during the parametric sweep described at the beginning of this chapter are fitted with the help of Matlab into a function of absolute value of body bias.

All three curves presented in Fig. 5.1 can be fitted with an R^2 of 1 into exponential functions of body bias of the following form:

$$f(|V_{bb}|) = a \cdot e^{b \cdot |V_{bb}|} \tag{5.27}$$

Where a and b are constants for a given temperature. At the moment, all simulations are performed at 27^{o} C.

As such, the leakage current consumed by a register array stops being exclusively a function of the HW, and becomes a bivariate function:

$$I_{leak}(HW, |V_{bb}|) = n \cdot I_0(|V_{bb}|) + \epsilon(|V_{bb}|) \cdot HW \tag{5.28}$$

Furthermore, this equation can now be expressed as:

$$I_{leak}(HW, |V_{bb}|) = n \cdot a_0 \cdot e^{b_0 \cdot |V_{bb}|} + a_\epsilon \cdot e^{b_\epsilon \cdot |V_{bb}|} \cdot HW \tag{5.29}$$

At this point we drop the $|V_{bb}|$ notation for convenience.

The next step is defining the behavior of V_{bb} as a random variable. This has already been hinted at in section 5.2. At this point we justify our assumptions.

In our body biasing scheme, the body bias is going to be governed by a discrete random variable S having a uniform probability distribution, such that:

$$V_{bb}(S) = V_{BBq} + S \cdot \Delta V_{bb} \tag{5.30}$$

Where S is a random variable that follows a uniform probability distribution. We define it such that S can adopt any integer value between $[-s_{max}, s_{max}]$, where s_{max} is the parameter of the countermeasure over which we are going to have control.

We set V_{BBq} at the middle of the range of allowable body bias values. In this particular technology, with $V_{bb_{MAX}} = 1$ V, and $V_{bb_{min}} = 0$ V this corresponds to $V_{BBq} = 0.5V$.

From this it can be seen that:

$$\Delta V_{bb} = \frac{V_{bb_{MAX}} - V_{BBq}}{s_{max}} \tag{5.31}$$

Which means that the step increase with which the body bias changes is going to be a function of the maximum number of steps of body bias.

With all these elements into consideration, we can calculate the different moments of S and $V_{bb}(S)$:

$$E[S] = 0$$

$$E[V_{bb}(S)] = V_{BBq}$$

$$Var(S) = \frac{s_{max} \cdot (s_{max} + 1)}{3}$$

$$Var(V_{bb}(S)) = \Delta V_{bb}^2 \cdot Var(S) \tag{5.32}$$

This leaves $V_{bb}(S)$ with a well defined distribution.

Finally, we can express equation 5.29 parameterized as a function of S:

$$I_{leak}(HW, S) = n \cdot a_0 \cdot e^{b_0 \cdot (V_{BBq} + S \cdot \Delta V_{bb})} + a_\epsilon \cdot e^{b_\epsilon \cdot (V_{BBq} + S \cdot \Delta V_{bb})} \cdot HW \tag{5.33}$$

With this, we have an expression governing the leakage current consumption of a register array as a function of a random body bias, parameterized with a random variable S over which the designer has control.

The idea behind the countermeasure is that, at the beginning of each encryption process, a value of S is chosen at random and independently. To obtain a notion of what is happening, consider first the ideal case exemplified by figure 5.4, when no countermeasure is applied.

If the secret key is known, every measurement of I_{leak} versus Hamming Weight for different values of plaintext X will fall in a straight line, as defined by Equation (4.3), giving a perfect correlation with a PCC of 1 (Equation 5.25). However, an incorrect key would produce values of I_{leak} outside of such curve, reducing the linear correlation between variables according to equation 5.26 (Fig. 5.6).

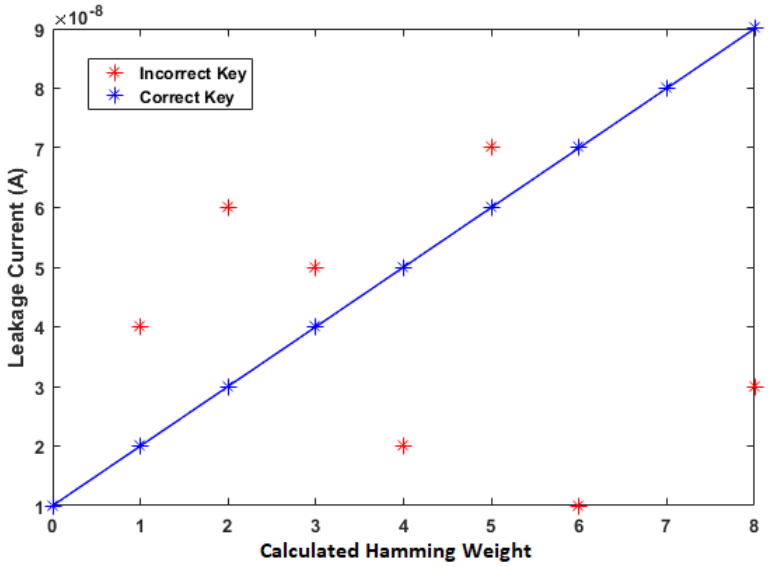

Figure 5.6: Measured leakage current vs Hamming Weight of different plaintext X in an 8-bit register. The solid line depicts equation (4.3) for a given register, while the markers represent measured current values for different plaintexts displayed according to the calculated Hamming Weight with a correct key (Blue) and an incorrect key (Red).

However, if we can consider the possibility that I_0, so far a constant, can become a random variable, and that the value of ϵ varies negligibly in comparison so as to be able to consider it constant, we obtain a collection of curves:

$$
\begin{aligned}
I_{Leak1}(X,k) &= n \cdot I_{01} + \epsilon \cdot HW \\
I_{Leak2}(X,k) &= n \cdot I_{02} + \epsilon \cdot HW \\
&... \\
I_{Leakn}(X,k) &= n \cdot I_{0n} + \epsilon \cdot HW
\end{aligned} \tag{5.34}
$$

Where I_0 is now a discrete random variable that can adopt values $\{I_{01}, I_{02}, ..., I_{0n}\}$ with probability $P[I_0 = I_{0i}] = p_i$, for $1 \leq i \leq n$.

Provided that I_0 and HW are independent variables and their variances are well defined, the PCC between the register's leakage current and the Hamming Weight now becomes:

$$
\rho_{I_{leak},HW} = \frac{\epsilon \cdot \sigma_{HW}}{\sqrt{(n^2 \cdot \sigma_{I0}^2 + \epsilon^2 \cdot \sigma_{HW}^2)}} < 1 \tag{5.35}
$$

Assuming that the attacker has no means of accessing the value I_0, this means that for a sufficiently large number of possible values of I_0, having the secret key can be indistinguishable from having an incorrect key, as the values of I_{leak} do not fall in a single curve.

This is exemplified in Fig. 5.7, where a collection of such curves is shown. Under a correct key assumption, the markers represent the correct evaluation of the Hamming Weight for each possible plaintext. However, it can be seen that much of the linear relation is lost.

5.3.3 Random Body Bias Analysis

Having a mathematical model of the body bias scheme and a conceptual understanding of how the scheme can reduce the PCC and, as a consequence, harden the ease with which the secret key can be extracted by performing LPA, we can derive an expression of the PCC when the countermeasure is applied. With these, we can observe trends as a function of the countermeasure parameters, as well as determine an overall efficiency or level of protection achieved.

In order to do so, we must calculate the PCC between the Hamming Weight of the dummy cryptosystem that we have thus far used, and the leakage current consumption of the cryptosystem as a function of a random body bias, parameterized with S. That is:

$$
\rho_{I_{Leak},HW} = \frac{Cov(n \cdot I_0(S) + \epsilon(S) \cdot HW, HW)}{\sqrt{Var(n \cdot I_0(S) + \epsilon(S) \cdot HW)}\sqrt{HW}}
$$

The probability distribution of the HW remains the same and so do its moments (Equations 5.14 and 5.17 for an attack on n-bits). Thus, we calculate the PCC as detailed in section 5.3.1. The whole derivation can be found in the Annex (10.2)

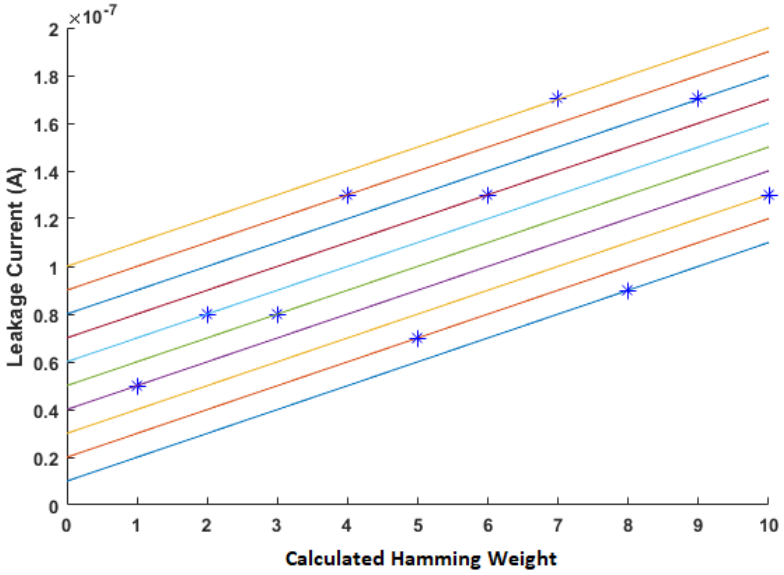

Figure 5.7: Collection of register leakage current vs Hamming Weight curves. Each line represents a possible realization of Equation (4.3) for different values of I_0. The markers represent measured current values for different plaintexts displayed according to the calculated Hamming Weight with a correct key

With all these considerations and developments we can finally express the PCC between the leakage current consumption of a register array and its Hamming Weight in the presence of a random body bias countermeasure as a function of:

- Technological parameters a_ϵ, b_ϵ, a_{I0}, and b_{I0}
- Countermeasure parameters V_{BBq}, ΔV_{bb}, and s_{max}

$$\rho_{I_{leak},HW} = \frac{\mu_\epsilon \sigma_{HW}}{\sqrt{p_1 + p_2 + p_3}}$$

$$p_1 = \sigma_{HW}^2 \sigma_\epsilon^2 + \mu_\epsilon^2 \sigma_{HW}^2 + \mu_{HW}^2 \sigma_\epsilon^2$$

$$p_2 = n^2 \cdot \sigma_{I0}^2$$

$$p_3 = 2 \cdot n \cdot \mu_{HW} \cdot Cov(\epsilon(S), I_0(S)) \tag{5.36}$$

Results

We begin our analysis by plotting the PCC under a variety of conditions, under the assumption that a lower PCC results in better protection against leakage power analysis attacks.

In all our analysis we use the following:

- The technological parameters a_ϵ, b_ϵ, a_{I0}, and b_{I0} are extracted using matlab's fitting tools for reverse well, fordward body bias D flip-flops simulated at 27 ºC
- V_{BBq} is set at the mid point of the maximum and minimum allowable voltages for the technology ($V_{bbMAX} = 1$ V and $V_{bbmin} = 0$ V, respectively). Thus, $V_{BBq} = 0.5$ V. This permits the body bias to span the entire Dynamic Range symmetrically
- We assume an attack on the dummy cryptosystem depicted in Fig. 5.2 under ideal conditions, where only the leakage current of the register array is taken into consideration, without any source of noise present.

With these, we first plot the PCC as a function of s_{max}, noting that:

$$\Delta V_{bb} = \frac{V_{bbMAX} - V_{BBq}}{s_{max}} \tag{5.37}$$

From Fig. 5.8 we can derive some conclusions. Firstly, the application of a random body bias does reduce the PCC. Also, the rate of reduction as a function of s_{max} depends on the size of the step. In this figure, the step size is fixed, such that a larger s_{max} brings the maximum body bias value closer to the maximum permitted by the technology.

However, when the entire Dynamic Range is used, that is, when the body bias can swing between V_{bbMAX} and V_{bbmin}, the PCC is the same for all the plotted possibilities, and approximately equal to $\rho = 0.048$, regardless of the size of the step.

Thus, it is clear that the wider the span within the Dynamic Range (DR) of the body bias used, the better the countermeasure performs. It is then of interest to study the behaviour of the countermeasure assuming that the whole DR is used, and establishing a trade-off between s_{max} and ΔV_{bb}, such that:

$$\Delta V_{bb} = \frac{DR}{2 \cdot s_{max}} \tag{5.38}$$

Where the Dynamic Range is fixed and defined as $DR = V_{bbMAX} - V_{bbmin}$.

Figure 5.9 show what happens when the DR is fixed but s_{max} and ΔV_{bb} depend on each other, such that the higher s_{max} the smaller ΔV_{bb} and vice versa.

It can be seen that, for small values of s_{max} (and large of ΔV_{bb}) the PCC is at its lowest. As s_{max} increases and ΔV_{bb} decreases, there is a small increase in the PCC, which begins to stabilize at $s_{max} \approx 20$, settling at $\rho \approx 0.048$, as

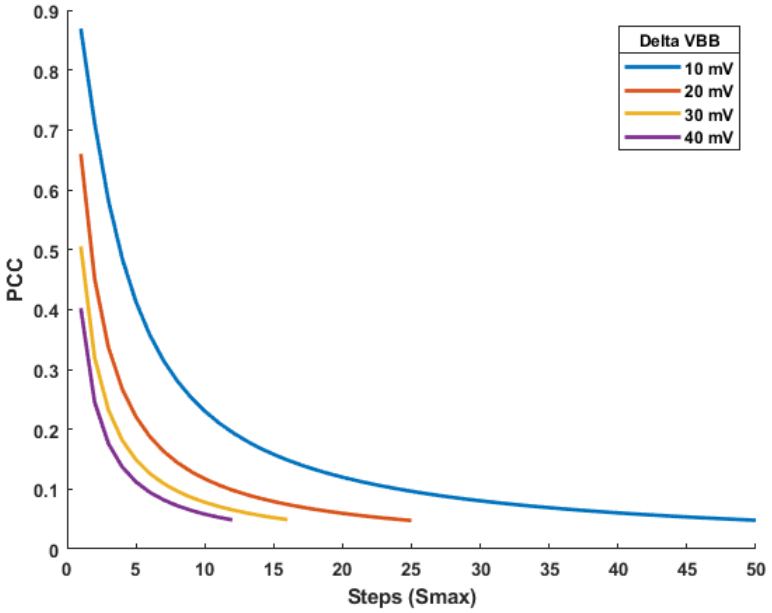

Figure 5.8: PCC between the leakage current and the Hamming Weight of an 8-bit register array in the presence of a random body bias as a function of the maximum number of steps

previously seen. This value is consistent with that observed in Fig. 5.5, where the PCC between the HW and the leakage current consumption of the dummy cryptosystem under attack for the correct key is approximately 0.042. The disparity between the theoretical result and the simulated one can be attributed to the fact that the realization of S does not perfectly reproduce a uniform distribution and so, some amount of noise in the PCC is expected.

Figure 5.9 also depicts the entropy of S as a function of s_{max}, where it is defined as:

$$H(S) = log_2(\frac{1}{2s_{max}+1}) \tag{5.39}$$

We choose to present these results (PCC and entropy) in the same graph to bring attention to the fact that, even if the PCC is at its lowest when s_{max} is small, there is a trade-off with the entropy of the random variable S. Thus, if S leaked information in a particular implementation of the countermeasure, it would yield the most information for small values of s_{max}, which might easily undo the effect of the countermeasure.

On the other hand, as s_{max} increases, the PCC stabilizes at relatively small values, while the entropy of S continues to rise. It might be of interest, then,

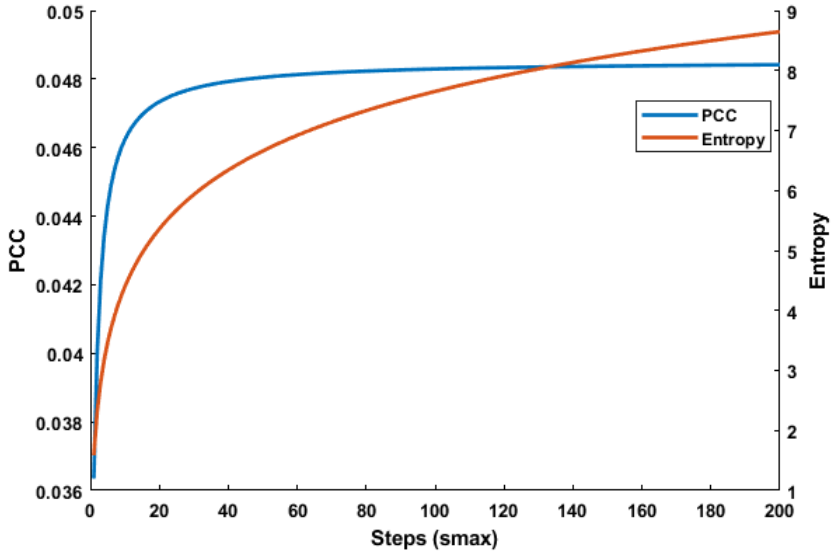

Figure 5.9: PCC between the leakage current and the Hamming Weight of an 8-bit register array in the presence of a random body bias, with a fixed maximum Dynamic Range, as a function of the s_{max}

when envisioning an implementation, to choose a value of s_{max} between 20 and 50. As s_{max} increases, ΔV_{bb} decreases, so designing a body bias generator with high granularity might be difficult while not offering much more protection, and still presenting sufficient entropy of S.

5.3.4 Random Body Bias: Trace Averaging

The metrics and plots derived in the previous sections inform us of the effect of the countermeasure on the PCC, but not on the overall difficulty to extract the secret key. While it is understood that a diminished PCC would hinder the extraction of the secret key, we need to be able to determine what happens under more realistic attack scenarios.

A random body bias acts as a noise enhancer, decorrelating the Hamming Weight and the leakage current consumption between different plaintexts by modyifing the floor and, to some extent, the slope of the leakage consumed by a register array. This varying leakage current, in effect, becomes noise. As such, this countermeasure is sensitive to trace averaging that can diminish the noise introduced.

Thus, it is of interest to study and quantify how trace averaging affects the capability of the countermeasure to hide the secret key.

The noise figures introduced by the countermeasure are a result of the variance of the leakage current consumption informed by the random body bias.

That is, those of $\sigma^2_{I_0}$, σ^2_ϵ (the variance of $I_0(S)$ and $\epsilon(S)$) and their covariance.

Under trace averaging conditions, for every plaintext x_i, the experiment or encryption process is going to be repeated N times. Thus, we will have access to N realizations of the random variable S, while the plaintext and, therefore, the Hamming Weight, remains the same.

If for each experiment the magnitude of I_{leak} is measured, we can then average the total number of values of I_{leak}.

$$\hat{I}_{leak}(HW, \hat{S}) = \frac{1}{N} \sum_{i=0}^{N-1} I_{leak}(HW, S_i) \tag{5.40}$$

As such, as N tends to infinity, it can be expected that I_{leak} will converge to its expected value, $I_{leak}(HW, E[S])$. This means that, following the notions introduced by figures 5.6 and 5.7, I_{leak} would cease to represent a collection of curves determined by the random body bias, and coalesce to a single curve. So, it can be expected that, under a correlation power analysis attack, the PCC tends to 1 with increasing number of averaged traces N.

Under such averaging conditions, we can expect the variances representing the noise to diminished with N. An approximation can be made regarding the reduction of the variance, such that:

$$\sigma^2_\epsilon \to \frac{\sigma^2_\epsilon}{N}$$

$$\sigma^2_{I_0} \to \frac{\sigma^2_{I_0}}{N}$$

$$Cov(\epsilon(S), I_0(S)) \to \frac{Cov(\epsilon(S), I_0(S)}{N} \tag{5.41}$$

Which follows from the fact that, for the sum of N independent random variables with equal variance:

$$Var(\frac{1}{N} \sum_{i=0}^{N-1} x_i) = \frac{1}{N^2} Var(\sum_{i=0}^{N-1} x_i) = \frac{N}{N^2} \sigma^2_x \tag{5.42}$$

Substituting the reduced variances into equation 5.36 we can recalculate the PCC between the Hamming Weight and the leakage current consumption of the register array in the presence of the countermeasure when trace averaging is taken into account.

The results can be seen in Table 5.1 and Fig. 5.10. Figure 5.10 represents the value of the PCC as a function of s_{max} for different values of N, while table 5.1 depicts the value of the PCC for various values of N when $s_{max} = 25$.

These results clearly show an increase of the PCC when trace averaging is applied, with the notion that an attacker only needs to increase the number of experiments performed to extract the secret key. Thus, the countermeasure hinders an attack, but does not necessarily impede it.

Table 5.1: PCC for maximum dynamic range of the Body Bias for different number of averaging encryption processes when $s_{max} = 25$

N	PCC
1	0.0476
10	0.1489
100	0.4300
1000	0.8331

We would like, however, to be able to somewhat quantify how much harder it is to extract the secret key when the countermeasure is applied, something that cannot be readily established only through the value of the PCC.

In order to do so, we rely on a frequentist approach, signifying the number of traces required to be able to extract the secret key with high probability. To be able to obtain this "empirical" figure we would need to simulate the dummy cryptosystem under study 256 times (one for each possible value of the plaintext of 8 bits), multiplied by the number of averages N. This procedure must be repeated multiple times to obtain sufficient CPA attempts that allow us to determine the probability of extracting the secret key for a given set of averages.

The simulation time of all these can be prohibitively high for elevated numbers of averages. Thus, given the deterministic nature of the leakage current Equation (5.1) under noiseless conditions, as demonstrated by the perfect correlation seen in Fig. 5.4, we opt to numerically simulate the analog behavior of the dummy cryptosystem. The numerical simulation of the analog behavior of the system allows us to simulate a CPA attack where each leakage current measurement, for every plaintext, is repeated N times and then averaged. This can be done given that the technological parameters of the registers under study are known (the constants in Equation 5.33), and thus these CPA's provide comparable results to the ones that would be obtained through electrical simulations in the Virtuoso ADE.

In order to carry out simulated Correlation Power Analysis Attacks, we define the the leakage current equation in Matlab (Equation 5.33), with the constant parameters a_ϵ, a_{I_0}, b_ϵ, and b_{I_0} extracted at 27 ºC, for flipped well, LVT, Forward body bias D flip-flops. We assume an attack on 8-bits, as in the dummy cryptosystem considered so far, so $n = 8$. We consider the maximum dynamic range of the body bias, so $DR = V_{bbMAX} - V_{bbmin} = 1$ V, with the quiescent point of the body bias set at the middle of the DR, with $V_{BBq} = 0.5$ V. s_{max} is set at 25, and the body bias step ΔV_{bb} is defined according to equation 5.38.

In these numerical simulations we fixed the secret key, again, at 170_{10}. In the simulations, we exhaust the plaintext values. That is, we iterate through each plaintext value. For each plaintext $X = i$, $0_{10} \leq i \leq 255_{10}$, we calculate

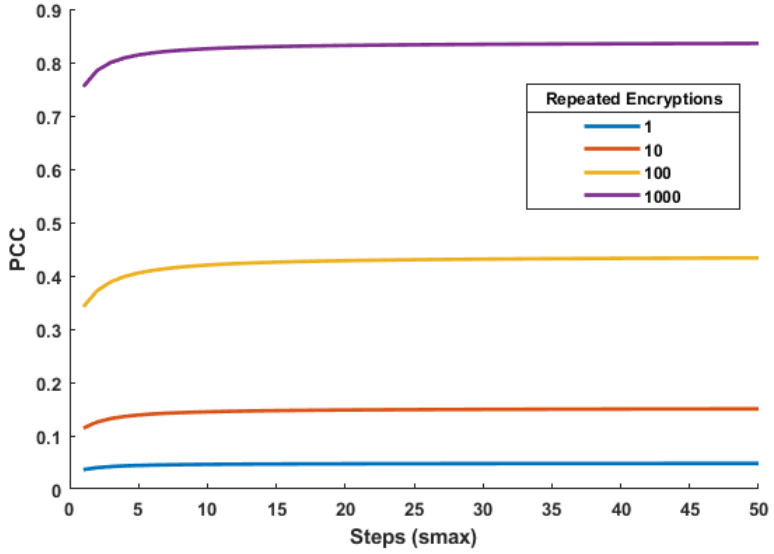

Figure 5.10: Effect of averaging on the PCC between the Leakage Current and the Hamming Weight of the register array in the presence of the proposed countermeasure with a fixed, maximum body bias DR of 1 V. It can be seen how averaging undermines the effect of the countermeasure for large number of repeated encryption processes.

the hamming weight of the value stored in the virtual register array as:

$$HW_i = \sum_{j=0}^{n-1} x_j \oplus k_j \tag{5.43}$$

Where j represents the bit index of the plaintext and the secret key, and the HW is the result of the sum of the bits stored at the array after performing a bitwise xoring of the plaintext and secret key.

At the same time, for each of the simulations, a random realization of $S = s$ with the distribution and limits above described is produced, simulating the countermeasure. Thus, for each simulation we obtain a realization of the following equation:

$$I_{leak}(HW_i, S = s) = n \cdot a_0 \cdot e^{b_0 \cdot (V_{BBq} + s \cdot \Delta V_{bb})} + a_\epsilon \cdot e^{b_\epsilon \cdot (V_{BBq} + s \cdot \Delta V_{bb})} \cdot HW_i \tag{5.44}$$

With trace averaging, every plaintext value is repeated N, each with an independent realization of S, such that a leakage current value is obtained for

each one:

$$\begin{gathered} I_{l1,i}(HW_i, s_1) \\ I_{l2,i}(HW_i, s_2) \\ \ldots \\ I_{l_N,i}(HW_i, s_N) \end{gathered} \tag{5.45}$$

Then, the result is averaged:

$$\hat{I}_{leak_i}(HW_i, \hat{S}) = \frac{1}{N} \sum_{j=1}^{N} I_{leak_{j,i}}(HW_i, s_j) \tag{5.46}$$

The value of Equation 5.46 is stored for each plaintext, for a total of 256 values. Finally, the PCC between the vector of 256 averaged leakage current values and the calculated Hamming Weight according to the input plaintext for every candidate key is evaluated as described in section 5.2 , thus numerically simulating a CPA attack in the presence of the countermeasure with noise averaging.

Four cases are contemplated: no averaging, and 10, 100 and 1000 repeated encryptions per input plaintext. The results can be seen in Figures 5.11 through 5.14.

It can be seen that, as the number of repeated encryptions increases, the CPA produces results that more correctly represent the simulated case when no countermeasure is applied (Fig. 5.4). Note also how the theoretical results as presented in Table 5.1, are very close to the results obtained through numerical simulations, as shown in Figures 5.11 through 5.14. Their difference is to be expected, as we are simulating population samples of the PCC, subjected to noise, which diminishes with increasing number N of averages.

To test the effect of increased PCC with averaging in the ability to correctly identify the secret key, a series of 100 distinct runs are then simulated for each collection of N averages. The amount of times the secret key presents the highest PCC is noted. The frequency with which the secret key is disclosed for different number of averaging encryption processes can be seen in Table 5.2.

Table 5.2: Success ratio of secret key identification in numerical simulations under different number of averaging samples

N	Success Rate
1	0
10	0.11
100	0.97
1000	1

Note that 100 repeated encryption processes appear to be enough to identify the secret key in most instances.

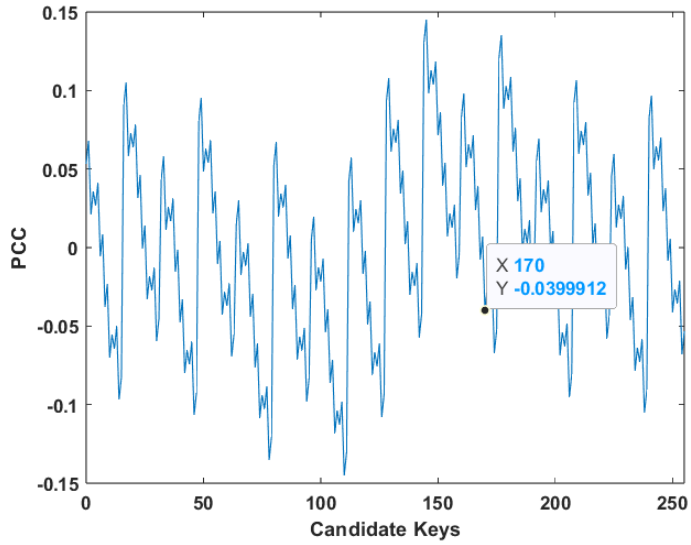

Figure 5.11: PCC between the numerically simulated leakage current and the Hamming Weight of the theoretical register array under attack for every candidate key in the presence of the proposed countermeasure. The results are obtained with one single encryption process per plaintext.

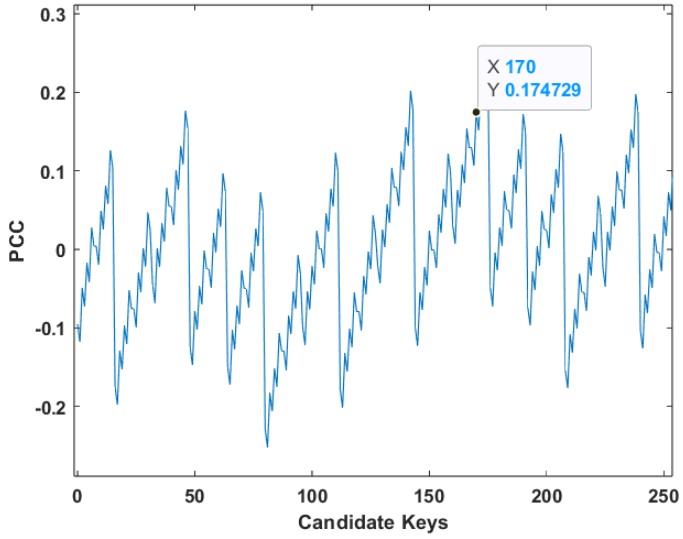

Figure 5.12: PCC of the numerically simulated and averaged leakage current and the Hamming Weight, with 10 encryption processes per plaintext.

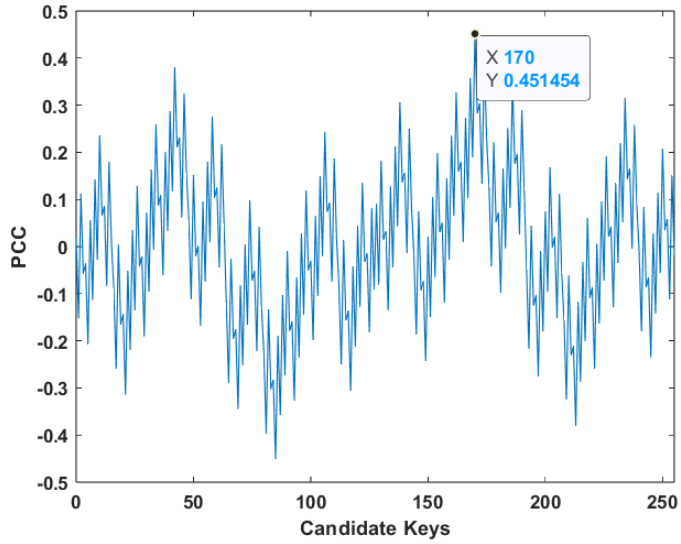

Figure 5.13: PCC of the numerically simulated and averaged leakage current and the Hamming Weight, with 100 encryption processes per plaintext.

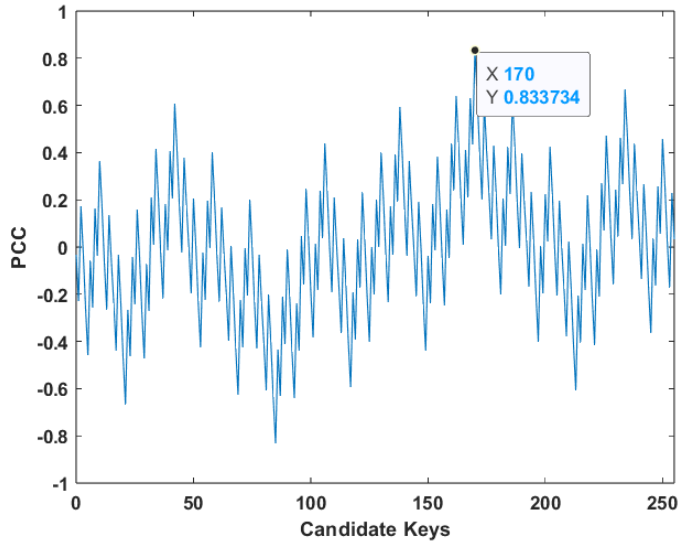

Figure 5.14: PCC of the numerically simulated and averaged leakage current and the Hamming Weight, with 1000 encryption processes per plaintext.

5.3.5 Algorithmic Noise

So far we have presented an analysis of the countermeasure and its simulation results as pertaining to a targeted register array of 8-bits. A more realistic scenario would deal with cryptosystems that presented a larger amount of bits, of which 8 would be under attack. For example, the AES cryptosystem can typically operate or perform encryptions on blocks of data (plaintext) of 128 bits of length, with round keys of the same length.

If only 8 of those bits are targeted, there are, at any point, 120 bits represented by a register array of the same length that are not pertinent to the attack. Nonetheless, these 120-bit register array consumes leakage current in a manner representative of all the analysis presented above.

Since the information conveyed by the leakage current consumption of these 120-bit register array does not elucidate information regarding the 8-bits of the secret key under attack, it is considered noise. And given that this noise is produced not by random physical elements (such as the brownian motion of electrons within conductors), but by the very structure of the encrypting algorithm, it is called algorithmic noise.

In a $n+m$-bit cryptosystem, with n bits under attack and m bits introducing algorithmic noise, the total leakage current consumption of the register array storing these $n+m$ bits can be expressed as:

$$I_{leak} = (n+m) \cdot I_0 + \epsilon \cdot (HW_n + HW_m) \tag{5.47}$$

Where HW_n refers to the Hamming Weight of the portion of bits under attack, while HW_m represents the Hamming Weight of the bits introducing algorithmnic noise. Under these circumstances, we assume that, much like in the analysis previously performed, each of the m bits follows an unbiased uniform probability distribution, and that the bits are independent from each other, so that the expected value and variance of HW_m are, respectively $E[HW_m] = \frac{m}{2}$ and $Var(HW_m) = \frac{m}{4}$.

With these, we can recalculate the PCC taking into account the effect of algorithmic noise. In the case in which no countermeasure is applied, where both ϵ and I_0 are assumed constant, it can be shown that (without any other sources of noise, and under a correct key assumption):

$$\rho_{Ileak,HW_n} = \frac{\epsilon \cdot \sigma_{HW_n}}{\sqrt{\epsilon^2(\sigma^2_{HW_n} + \sigma^2_{HW_m})}} = \sqrt{\frac{n}{n+m}} \tag{5.48}$$

From here, it is straightforward to calculate the PCC between the Hamming Weight of the n bits under attack and the leakage current consumed by a $n+m$ bit cryptosystem in the presence of the countermeasure described in previous sections.

We must simply take into account the changes introduced in the denominator of Equation (5.36) (the variance of the leakage current consumed by the register

array) by the m bits that introduce algorithmic noise. Again, we break down the variance of I_{leak} into three components $Var(I_{leak}) = p1 + p2 + p3$, with:

$$p1 = Var(\epsilon(S) \cdot (HW_n + HW_m))$$

$$p1 = \sigma_\epsilon^2(\sigma_{HW_n}^2 + \sigma_{HW_m}^2) + \mu_\epsilon^2(\sigma_{HW_n}^2 + \sigma_{HW_m}^2) + ...$$

$$+\sigma_\epsilon^2(\mu_{HW_n}^2 + \mu_{HW_m}^2) + 2\mu_{HW_n}\mu_{HW_m}\sigma_\epsilon^2$$

$$p2 = Var((n+m) \cdot I_0(S))$$

$$p2 = (n+m)^2 \cdot \sigma_{I_0}^2$$

$$p3 = 2 \cdot Cov(\epsilon(S) \cdot (HW_n + HW_m), (n+m) \cdot I_0(S))$$

$$p3 = 2 \cdot (n+m) \cdot (\mu_{HW_n} + \mu_{HW_m}) \cdot Cov(\epsilon(S), I_0(S)) \tag{5.49}$$

The same analysis performed for the countermeasure in the presence of noise averaging can be done under these new conditions. In fact, the same effect is expected for the different variances σ_ϵ^2, $\sigma_{I_0}^2$, and $Cov(\epsilon(S), I_0(S)))$ as in Equation 5.41; namely, their values are reduced by a factor of N, with N being the number of samples.

The same effect can also be expected for the variance of HW_m, which becomes $\sigma_{HW_m}^2/N$. In the particular case with algorithmic noise but no countermeasure applied, it can be shown that the PCC between the leakage current and the Hamming Weight of the bits of interest (Equation (5.48)) becomes:

$$\rho_{Ileak,HW_n} = \sqrt{\frac{n}{n + \frac{m}{N}}} \tag{5.50}$$

With these, we can again plot the PCC as a function of s_{max} and the number of averaged traces in the case of algorithmic noise, considering the same conditions defined in previous section (namely, $V_{BBq} = 0.5$ V, $DR = 1$ V, and $\Delta V_{bb} = \frac{DR}{2s_{max}}$). We consider a cryptosystem with a total of 128 bits, with $n = 8$ bits under attack and $m = 120$ bits of algorithmic noise.

The results can be observed in Fig. 5.15, where 250 times the same number of averaged samples are required as compared to Fig. 5.10 to obtain the same values of the PCC. Note that, in both cases, the total number of experiments equals those noted in the legend of the figure times 256.

Note that in the case of a cryptosystem with $n + m$ bits and an attack on n bits of interest, noise averaging is not obtained by repeating N encryption processes with exactly the same plaintext. Rather, we maintain the plaintext affecting the n bits constant during the N encryption processes, but for each of

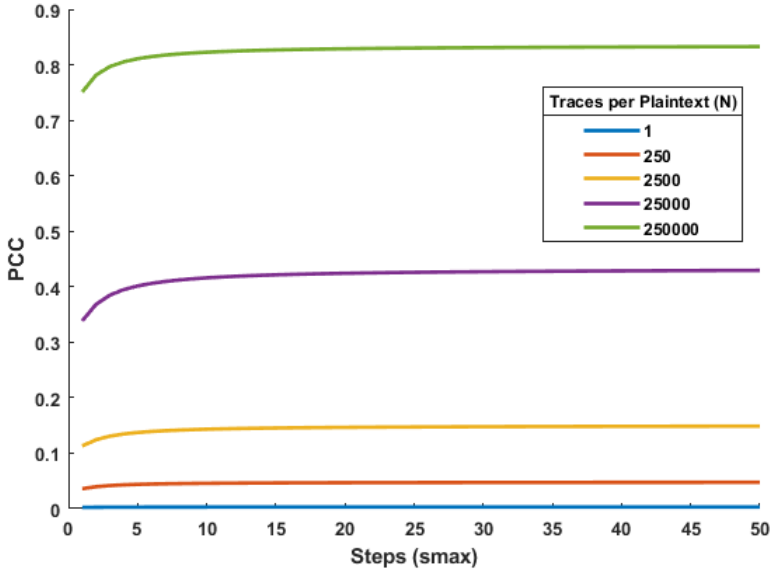

Figure 5.15: Effect of averaging on the PCC between the Leakage Current and the Hamming Weight of a 128-bit register array, with 8 bits under attack, in the presence of the proposed countermeasure with a fixed, maximum body bias DR of 1 V and different number N of traces per plaintext.

these encryption processes a random plaintext is chosen for the m remaining bits. With these modifications, numerical simulations of a CPA can be performed as described in section 5.3.4.

We first perform simulated CPA attacks on $n = 8$ bits of an 128-bit cryptosystem with no countermeasures applied, to have a controlled sample and better distinguish the effect of the countermeasure when algorithmic noise is also considered. We then simulated again a CPA applying the countermeasure. The results can be seen in Tables 5.3 and 5.4. The success rate noted in these tables is the ratio of the number of times the secret key is identified out of the 100 simulated attacks.

Both tables present the value of the PCC calculated according to equations 5.50 and 5.49 (column PCC-Theo) along one realization of the PCC obtained through a simulated CPA (Column PCC-CPA). The different values depicted in the columns PCC-CPA are, thus, subjected to noise and can vary between experiments, under the same conditions. However, it can be seen that, as N increases, the values get closer to those obtained from the evaluation of Equations (5.50) and (5.49), indicating that the model is representative of the expected behavior of the circuit.

Thus, it can be seen that the countermeasure significantly increases the number of samples required to extract the secret key, highly impeding the per-

Table 5.3: Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack without countermeasure for different number N of averaged traces per plaintext.

No Countermeasure			
	PCC		
N	Theo	CPA	Success Rate
1	0.25	0.17	0.61
300	0.976	0.977	1

Table 5.4: Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack with countermeasure for different number N of averaged traces per plaintext.

Countermeasure			
	PCC		
N	Theo	CPA	Success Rate
1	0.0030	0.0091	0
250	0.0470	0.0290	0.05
2,500	0.1472	0.1778	0.14
25,000	0.4328	0.4294	0.98
250,000	0.8301	0.8357	1

formance of leakage power analysis attacks on registers.

5.3.6 Other Conditions

So far our analysis has limited to the case of registers implemented with Low Threshold Voltage (LVT), flip-well, forward body bias (fbb) transistors at 27 ^{o}C. These transistors see their threshold voltage decreased as the absolute body bias (positive for NMOS, and negative for PMOS transistors) increases.

In this section we present results of the effect of the countermeasure on two distinct conditions:

- The same type of registers at 80 ^{o}C

- Using registers implemented with FDSOI, regular threshold voltage, reverse body bias transistors. These transistors see their threshold voltage *increased* with increasing values of $|V_{bb}|$. Thus, as opposed to registers

implemented with Forward bodybias transistors, the magnitudes of I_0, I_1, and ϵ are monotonically decreasing functions of $|V_{bb}|$ but, rather, decreasing.

In both cases, it is of interest to see how μ_ϵ, the intensity of the signal of interest, varies as compared to the variance of I_{leak}, the introduced noise that decorrelates the leakage consumption from the HW of interest.

For higher temperatures, we expect all the leakage current magnitudes to increase, but are interested in determining whether the signal of interest increases at a higher rate than the noise, and thus the attack becomes easier (fewer samples are required), or if the opposite happens.

For regular threshold voltage, reverse body bias registers, we expect the leakage current magnitudes to decrease, especially with increasing values of body bias. Thus, we aim to determine whether the signal of interest decreases faster than the noise under these conditions.

The results can be seen in Table 5.5 for the FBB, 80 ºC conditions and in Table 5.6 for the RBB, 27 ºC experiments, where similar calculations of the PCC and simulated CPA attacks are performed as in previous sections.

Table 5.5: Forward Body Bias Registers at 80 C -Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack with countermeasure for different number N of averaged traces per plaintext.

Countermeasure			
	PCC		
N	Theo	CPA	Success Rate
500	0.0468	0.0138	0.05
5,000	0.1467	0.1560	0.14
50,000	0.4246	0.5267	0.98
500,000	0.8290	0.8474	1

The results are worth discussing. On the one hand, there is an expectation and empirical understanding ([51]) that higher temperatures facilitate the performance of LPA attacks by enhancing the magnitudes of interest. However, it can be seen in Table 5.5 that, when the countermeasure is applied, the noise introduced surpasses the signal enhancement, further hindering the acquisition of the secret key. Thus, this countermeasure offers more protection with higher temperatures when registers implemented with LVT, FBB are used.

On the other hand, using registers implemented with Reverse body bias transistors seems to be detrimental to the efficiency of the countermeasure, as seen in Table 5.6 by the fact that fewer traces are required to disclose the secret key.

Table 5.6: Reverse Body Bias Registers at 27 C - Theoretical and numerical simulation results of the PCC, along the success ratio for 100 simulated attacks, for a 128 bit register array with 8 bits under attack with countermeasure for different number N of averaged traces per plaintext.

Countermeasure			
	PCC		
N	Theo	CPA	Success Rate
77	0.0467	0.0138	0.05
770	0.1462	0.1560	0.14
7,700	0.4234	0.5267	0.98
77,000	0.8280	0.8474	1

5.4 Conclusions

In this chapter, an initial investigation on the nature and relation between the body bias of D flip-flops implemented in CMOS FDSOI transistors and their leakage current consumption is made.

Through the information gathered, and based on the nature of Correlation Power Analysis attacks, what they target, and how they are effective in extracting the secret key of cryptosystems, a body bias scheme is proposed.

The body bias scheme introduces leakage current noise by producing a random floor level of static power consumption at the beginning of each encryption process. Through the modification of this floor, which effectively acts as inter-trace noise power, the correlation between the Hamming Weight of a register array and the static power it consumes is diminished. Thus, the possibility of performing CPA attacks based on static current as a side-channel is hindered.

This body bias schemes serves, effectively, as a countermeasure. The efficiency of the countermeasure is analyzed through the development of a statistical power model of the leakage consumption of a register array. The model is parameterized through the technological parameters of the flip-flops, and design variables of the countermeasure. The efficiency of the countermeasure is then compared under a variety of magnitudes for the design variables by calculating the PCC between the HW and power model.

Limitations to the capabilities to hinder CPA attacks are demonstrated, showcasing a floor to the minimum PCC. This is demonstrated to not be entirely secure, as showcased by the small entropy presented in these cases. A ceiling to the maximum PCC is also established, showcasing that beyond a certain point, further increasing the granularity of the body bias scheme barely offers more protection.

The statistical power model is further refined by the inclusion of sources of algorithmic and non-algorithmic noise, recalculating the PCC under such conditions, reaching drawing similar conclusions.

The possibility of undoing the effects of the countermeasure through repeated experimentation and trace averaging is also presented, along the changes produced in the PCC when different amount of samples are considered. To complete the picture, the efficiency of the countermeasure is then tested through numerical simulations of CPA attacks in the presence and absence of the body bias scheme, obtaining rough figures of the success rate of secret key identification. These metric is determined for the control setting, without the use of a countermeasure, and for the case of the countermeasure under varying numbers of trace averaging. The results showcase that, while the countermeasure does not fully protect a cryptosystem, it severely increases the number of experiments required to disclose the secret key with high probability.

The same procedure is then repeated for the case of the same type of registers (FBB, LVT), at increased temperatures, and for registers implemented with RBB, RVT transistors. In the case of FBB, LVT registers, an increased temperature proves beneficial to the efforts of the countermeasure, probably by the increase of noise that can be introduced through the body bias scheme. Registers implemented with RBB, RVT transistors, however, suffer what could be seen as a paradoxical effect, in that despite presenting lower magnitudes of leakage current, leakage power attacks are more successful against them. This could be explained through the same reasoning, an inability to introduce as much noise, given the increase of threshold voltage with increasing body bias.

Overall, the countermeasure seems effective in its capability to reduce the success of secret key acquisition under LPA attacks, forcing an attacker to increase the number of traces to be taken significantly.

The main limiting factor of this whole analysis is that it is exclusively presented based on simulations. However, the author believes that the models are sufficiently grounded so as to be reliable, at the very least, in the trends presented during the analysis. This methodology limitations are, however, still present, and further experimentation on physical implementations of the countermeasure could shed more light on the true effectiveness of the countermeasure.

Chapter 6

Current Balancing Body Bias

So far we have explored the effect that a random body bias applied at the beginning of each encryption process can have on the ease with which the secret key can be identified through leakage power analysis attacks.

The exploration of this potential countermeasure based on the nature of FD-SOI technology has proved somewhat successful, particularly for low threshold voltage, forward body bias transistors, more so as the temperature at which the attacks take place increases. The countermeasure does not appear to be as effective when regular threshold voltage, reverse body bias transistors are considered, possibly due to the fact that the leakage current noise that can be introduced with these devices is somewhat smaller.

All in all, the analysis presented does point to the countermeasure being effective under the appropriate circumstances.

In this chapter, we explore under which conditions the countermeasure can be nullified, in the particular cases of the AES and the Trivium stream cipher. However, the manner of the attack that undermines the initial countermeasure proposal can be generalized to other cryptosystems whenever these conditions are met. We contrast the effect of this potential vulnerability with the results of the original proposals for a random body bias countermeasure, determining the ease with which the secret key can be disclosed when the vulnerability is exploited.

The chapter goes on to explore another potential countermeasure based on an alternative body bias scheme that would protect against these vulnerabilities.

6.1 Vulnerabilities to Random Body Bias Countermeasure

The basis of the potential vulnerabilities that can undermine the countermeasure so far presented stem from the possibility of finding a known state within the encryption process.

The countermeasure capability of preventing the acquisition of the secret key originates from establishing a distinct and random value of the magnitude of the leakage current floor at the beginning of each encryption process.

However, if a known state can be derived and related to an unknown state *within* the same encryption process, it might be possible to perform a "normalization" of the random body bias for each encryption process.

That is, consider that we are able to perform two measurements of the leakage current of a cryptosystem at two different stages of the same encrypting process, t_1 and t_2.

Considering that these two measurements are taken within the same encryption process, the value of the body bias is the same for both of the measurements.

Consider that the value of the Hamming Weight stored in the register array of interest at time t_1 is known, while the value of the Hamming Weight at time t_2 is unknown and depends on the secret key.

The total leakage current consumption for these two instances, considering no sources of algorithmic or non-algorithmic noise, for an attack on n bits, can be expressed as:

$$\begin{aligned} I_{leak}(HW, V_{bb}, t_1) &= n \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot HW_{t1} \\ I_{leak}(HW, V_{bb}, t_2) &= n \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot HW_{t2} \end{aligned} \tag{6.1}$$

If the value of HW_{t1} is known for every possible plaintext, and HW_{t2} is the result of an encryption process directly following the bits that conform the result of HW_{t1}, it is possible to perform a bivariate attack in the form of a normalization of the noise floor by subtracting $I_{leak}(t_1)$ from $I_{leak}(t_2)$. That is:

$$I_{leak}(V_{bb}, t_2) - I_{leak}(V_{bb}, t_1) = \epsilon(V_{bb})(HW_{t2} - HW_{t1}) \tag{6.2}$$

Note how, in equation 6.2 the term $n \cdot I_0(V_{bb})$ has vanished, given that the body bias value remains the same within the same encryption process and is thus equal between the two equations. We have seen from the analysis presented in the previous chapter that this term represents the main element that introduces noise. As such, it is theoretically possible to greatly reduce the impact of the countermeasure under such conditions.

Because the statistical relation between HW_{t2} and HW_{t1}, and the fact that HW_{t1} must be known for every possible plaintext, the structure of the algorithm is not negligible when analyzing these vulnerabilities. As such, they are going to be explored in particular, as related to two widely used cryptosystems: the

AES cryptosystem, a block cipher; and the Trivium cryptosystem, a Non-Linear Feedback Shift Register (NLFSR) cryptosystem, a stream cipher.

How the random body bias countermeasure can be bypassed in each instance will be explained along the minimum considerations required regarding each cryptosystem.

As such, we begin our analysis on these vulnerabilities by introducing some notions regarding the AES cryptosystem.

6.1.1 The AES and Random Body Bias

The Advanced Encryption Standard, also known as the AES, is a block cipher capable of encrypting 128 bits of plaintext per run. It has various modes of operating that are not going to be described here in detail, as they are not pertinent to the discussion. However, more information on its functionality and development can be found in [72].

The AES is an example of a substitution and permutation block cipher, as succinctly detailed in Chapter 2. At the same time, it is what is referred to as an iterated block cipher, meaning that the encryption process is done through the repeated execution of the same algorithm several times. Each of these executions is called a round.

Each round uses what is called a round key. The AES accepts three possible key sizes, 128-, 192-, and 256-bit keys. Each key size determines the number of rounds used to encrypt the 128 bits of the plaintext. While not really pertinent to our discussion, we will be focusing on the 128-bit key case.

With a 128-bit key, the AES encryption process uses 10 rounds of encryptions, with 11 round-keys. It is important not to confound the 128-bit secret key with the 11 distinct round keys, despite the fact that they are all of the same length.

All 11 round keys are derived deterministically from the secret key through a publicly known algorithm. Through a series of shifts and substitutions the algorithm expands the secret key into 11 round keys. The process of obtaining the round keys from the secret key is called the key schedule, when referring to iterated block ciphers.

The central component of attack of the AES, of interest in our analysis of the vulnerabilities of the previously presented countermeasures, is the so-called state matrix. The state matrix can be conceptualized as a simple register array, a 4×4-byte memory element that stores the plaintext and the intermediate values processed by the algorithm after each round of encryption.

Each round of encryption in the AES is composed of the same processes (AddRoundKey, SubByte, ShiftRows, and MixColumns) in the following fashion:

- A series of bytes, be them the plaintext or intermediate variables, are loaded onto the state matrix.
- The content of the state matrix is XORed with the pertinent round key (AddRoundKey)

- Each byte resulting from this XORing is fed into a Substitution Box (S-Box) through the SubByte subroutine. The resulting bytes are again loaded onto the state matrix, substituting the previous contents.
- The contents of the state matrix are first shuffled (ShiftRows) and finally a linear transformation of the columns is applied (MixColumns), finalizing the round

For our analysis we need only focus on the first three processes; namely, the loading of the plaintext onto the state matrix, the XORing operation, and the substitution operation.

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}

Consider the state matrix depicted, loaded with the 16 bytes that conform the plaintext, at the very beginning of the encryption operation, where each cell contains 8 bits. Consider byte x_0, and assume its Hamming Weight at the moment the plaintext is loaded onto the matrix as HW_{t1}.

Consider then both the AddRoundKey and the SubBytes operation applied to byte x_0. Both operations can be expressed as:

$$x'_0 = S(x_0 \oplus k_0) \tag{6.3}$$

where k_0 denotes the portion of the secret key with which x_0 is bitwise xored, and the function $S(\cdot)$ represents the substitution operation. At this point x'_0 is stored in the state matrix, and its Hamming Height can be considered to be, following from the above discussion, HW_{t2}.

x'_0	x'_1	x'_2	x'_3
x'_4	x'_5	x'_6	x'_7
x'_8	x'_9	x'_{10}	x'_{11}
x'_{12}	x'_{13}	x'_{14}	x'_{15}

Given that control over the plaintext is presumed, the value of x_0 is known in every instance of an attack, and the relation that arises between HW_{t1} and HW_{t2} in the form of a subtraction is exclusively dependent on the portion of the round key of interest, under the assumption that disclosing the totality of a round key is equivalent to breaking the cryptosystem.

Under this condition, we can consider that, with an unknown round key, the subtraction of HW_{t1} from HW_{t2} gives rise to a new random variable, which we will call Z.

$$Z = HW_{t2} - HW_{t1} \tag{6.4}$$

And so, equation 6.2 can be expressed as:

$$I_{sub}(V_{bb}, Z) = \epsilon(V_{bb}) \cdot Z \tag{6.5}$$

Under these conditions, a correlation analysis can be performed on the magnitude I_{sub} that arises from a the bivariate analysis. If we were to calculate the PCC between the random variable Z and the subtracted leakage current consumption, without algorithmic or non-algorithmic noise, the result would be the following:

$$\rho_{I_{sub},Z} = \frac{\mu_\epsilon \cdot \sigma_Z}{\sqrt{\sigma_Z^2 \cdot \sigma^2\epsilon + \mu_\epsilon^2 \cdot \sigma_Z^2 + \mu_Z^2 \cdot \sigma_\epsilon^2}} \tag{6.6}$$

Where μ_ϵ and σ_ϵ^2 are the expected value and variance of $\epsilon(V_{bb})$ when a random body bias is applied and μ_Z and σ_Z^2 are the expected value and variance of Z.

We can take a step further and assume two properties from Z that arise from the fact that both HW_{t2} and HW_{t1} are random variables with assumed known expected value and variance, based on the cryptosystem properties. Namely, we are going to assume that the expected value and variance of Z are known, and establish computationally through a small simulation of the AES if our assumptions hold. In the mean time, we will consider the following:

$$E[Z] = E[HW_{t2} - HW_{t1}] = E[HW_{t2}] - E[HW_{t1}] = \frac{n}{2} - \frac{n}{2} = 0 \tag{6.7}$$

$$Cov(HW_{t2}, HW_{t1}) = 0 \tag{6.8}$$

$$Var(Z) = Var(HW_{t2} - HW_{t1}) = Var(HW_{t2}) + var(HW_{t1}) = 2 \cdot \frac{n}{4} \tag{6.9}$$

That is, we are assuming that each bit of the plaintext follows an unbiased uniform discrete proability distribution, with bits being independent from each other. We are also assuming that the cryptosystem does not introduce any form of bias and, thus, we can expect both instances of the Hamming Weight to present the moments depicted above and explained in the previous chapter. Finally, we are assuming that while the variables HW_{t2} and HW_{t1} are not independent, their correlation is 0 due to the effect of the AES substitution box. We expect this last assumption to be wrong, but it does not influence the following derivation.

With these, we can consider that $\mu_Z = 0$ and that $\sigma_Z^2 = \frac{n}{2}$, with n being the bits of the round key under attack, in all our analysis so far, 8.

This allows Equation 6.6 to be further simplified into:

$$\rho_{I_{sub},Z} = \frac{\mu_\epsilon}{\sqrt{\sigma_\epsilon^2 + \mu_\epsilon^2}} \tag{6.10}$$

Note that this equation already demonstrates the effectiveness of these bivariate attack to heavily undermine the effect of the random body bias. The principal element introducing noise was the term $n^2 \cdot \sigma^2_{I_0}$ which, for LVT, flip well transistors at 27ºC is three orders of magnitude greater than σ^2_ϵ.

In fact, we can plot Equation 6.10 and Equation 5.36 parameterizing the random body bias as a function of S, with $V_{bbq} = 0.5$ V and $DR = 1$ V to compare this disparity. The results can be seen in figure 6.1

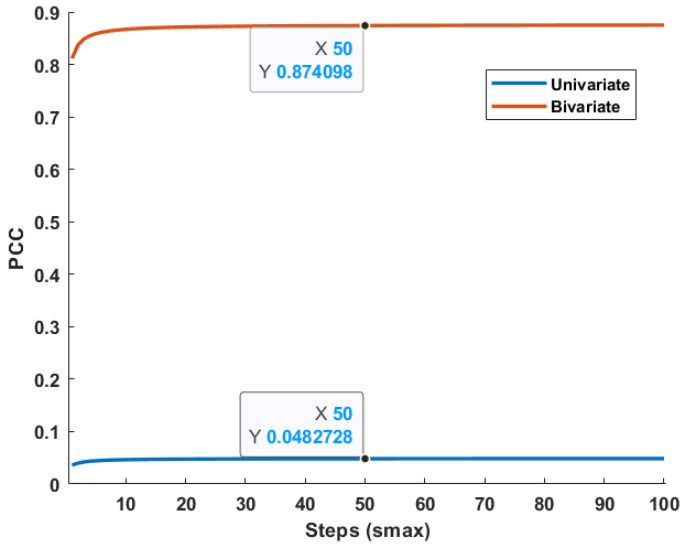

Figure 6.1: PCC between the univariate leakage magnitude (Blue) and its HW, as described in the previous chapter, and between the bivariate leakage magnitude I_{sub} (Orange) and Z as a function of s_{max}.

It can be seen that most of the effectiveness of the countermeasure is lost, and we expect that an attack can extract the secret key with ease without need to devote much effort to trace averaging.

We then assess our analysis regarding the moments of Z (μ_Z and σ^2_Z). In order to do so, we simulate the first three steps of the AES algorithm. Namely, the loading of the plaintext, its XORing with the round key, and the substitution through the S-Box. The value of the Hamming Weight is calculated for the plaintext and for the result arising from the substitution process. That is, following the nomenclature used:

$$\begin{aligned} HW_{t1} &= HW(x_0) \\ HW_{t2} &= HW(S(x_0 \oplus k)) \\ Z &= HW_{t2} - HW_{t1} \end{aligned} \tag{6.11}$$

This is done for all the possible values of the plaintext, ranging from $0 \leq x_0 \leq 255$, and for all possible keys $0 \leq k_0 \leq 255$. Thus, Z is a 256×256 matrix of Hamming Weight Values where the rows represent the values of the plaintext, while the columns represent the values of the round key. We can then calculate the mean and the variance of each column, obtaining the expected value and variance of Z for all possible values of the plaintext as a function of the round keys. The results can be seen in Figs. 6.2 and 6.3

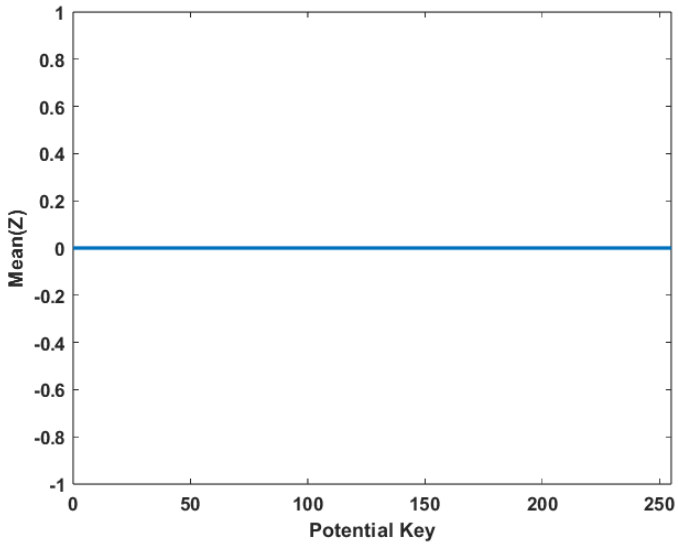

Figure 6.2: Mean value of the random variable Z for all possible plaintexts as a function of the round key

It can be seen that the expected value of Z is zero for all possible plaintexts and round keys, thus validating the initial analysis. The variance, however, presents differing values depending on the secret key, indicating some non-zero correlation between HW_{t1} and HW_{t2}. This, however, has no impact on the metric of the PCC and the ease with which the secret key can comparatively be extracted.

6.1.2 The Trivium and Random Body Bias

Trivium Functionality

The Trivium is a Non-Linear Feedback Shift Register that can be used to implement a Pseudo Random Number Generator or a stream cipher depending on the requirements of the application. It was initially designed under the premise of low power circuits (embedded systems, IoT...) that require an acceptable level of security while maintaining as small a power consumption as possible. It

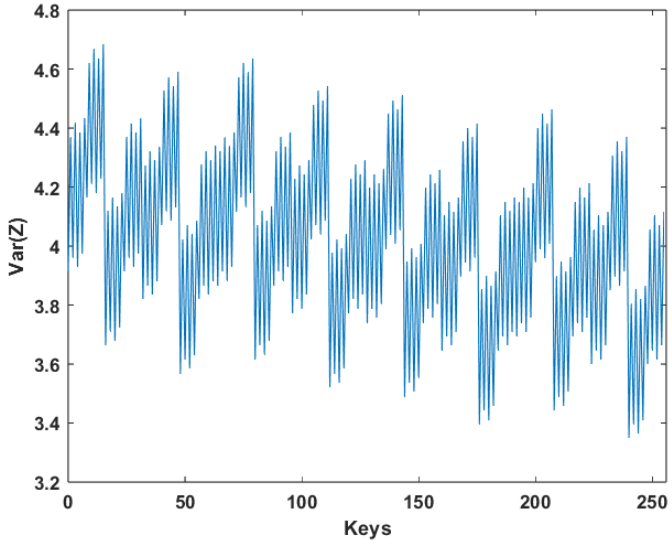

Figure 6.3: Variance of the random variable Z for all possible plaintexts as a function of the round key

has, thus, been described as a light-weight cryptosystem, able to be implemented with minimum standard cell count and area, with small power necessities. While a certain degree of its functionality will be discussed here shortly, its full design and implementation can be found in [15].

The Trivium is composed of three shift registers, which we will call X, Y and Z. Each of these registers contains 93, 84, and 111 bits respectively and serve as the memory element that stores the state of the intermediate variables to be attacked. Thus, the trivium contains a total of 288 bits, typically numbered as bit $s1$ to bit $s288$ (Fig. 6.4)

The Non-linear feedback elements implemented within the Trivium are carried out by the combinational functions $t1$, $t2$ and $t3$. While these functions are all equal in structure (see Fig. 6.5), they present as inputs different bits of the three shift registers. What these combinational functions have in common besides their composition is the fact that their output serves as the input to the first bit of each shift register. Thus, function $t1$ feeds into the first bit of register Y, labelled $s94$, $t2$ into the first bit of register Z ($s178$) and $t3$ feeds into register X ($s1$).

Before beginning its encryption process, the Trivium undergoes what is called its Initialization Phase. During its initialization phase, the secret key and Initialization Vector (IV) are firstly loaded onto the Trivium. Both the secret key and IV have a length of 80 bits. The secret key is loaded onto register X, from bits $s1$ to $s80$, while the IV is loaded onto register Y, from bit $s94$ to bit $s173$. The rest of the bits on these registers are initialized to 0. Register Z

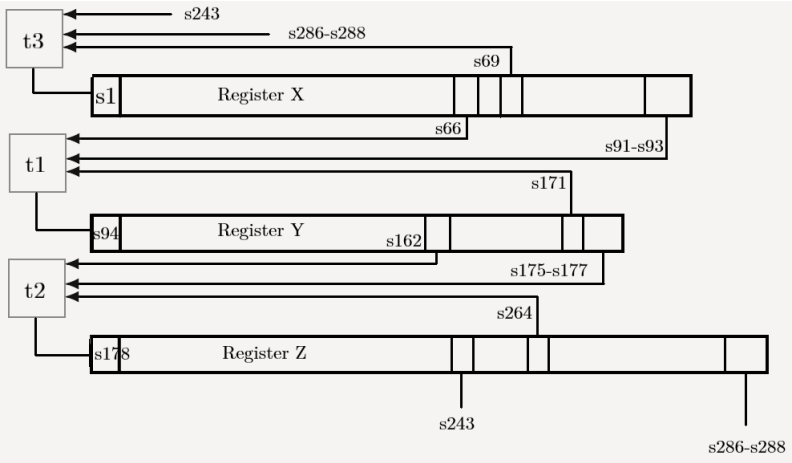

Figure 6.4: Schematic representation of the Trivium Stream Cipher

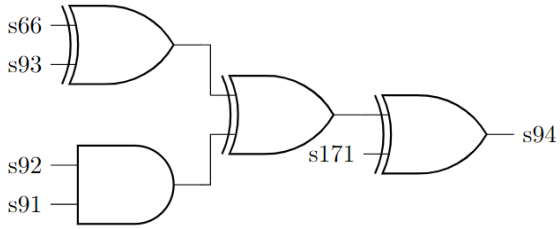

Figure 6.5: $t1$, the non-linear function under attack. The result from this function feeds into the first bit of register Y.

is, likewise, initialized to 0, except for bits $s286 - s288$, which are programmed with 1's.

After the pertinent bits have been loaded, the initialization phase continues by undertaking 4×288 periods of execution in which the trivium constantly feeds onto itself. After this number of periods, the Trivium begins producing the keystream with which the plaintext is XORed bit by bit.

CPA of the Trivium With leakage as a side-channel

Power analysis attacks on the Trivium take place during the first 80 periods of the initialization phase [73]. These attacks target bit $s94$, that is, the output of the function $t1$. This bit is targeted because the output of function $t1$ depends exclusively on bits of the secret key and 1 bit of the IV per period.

These attacks take place iteratively. That is, during different periods of operation within the initialization phase, different bits of the secret key can be attacked.

Typically, correlation power analysis attacks on the Trivium stream cipher

Table 6.1: Key Extraction Progression for different periods of the initialization phase. The pertinent bits of the Initialization Vector have been omitted

Key Extraction			
Absolute Periods	Relative Period (j)	Function	Key bits
1 to 12	j=0 to 11	k_{66-j}	66-55
28 to 37	j=0 to 9	$k_{66-j} + k_{65-j} \cdot k_{64-j} + k_{39-j}$	39-30
55 to 62	j=0 to 7	$k_{39-j} + k_{38-j} \cdot k_{37-j} + k_{12-j}$	5-12
70 to 80	j=0 to 10	$k_{24-j} + k_{23-j} \cdot k_{22-j} + k_{66-j}$	24-15
43 to 54	j=0 to 11	$k_{51-j} + k_{50-j} \cdot k_{49-j} + k_{24-j}$	51-40
16 to 27	j=0 to 11	$k_{78-j} + k_{77-j} \cdot k_{76-j} + k_{51-j}$	78-67
67 to 69	j=0 to 2	$k_{27-j} + k_{26-j} \cdot k_{25-j} + k_{69-j}$	27-25
40 to 42	j=0 to 2	$k_{54-j} + k_{53-j} \cdot k_{52-j} + k_{27-j}$	54-52
38 to 39	j=0 to 1	$k_{56-j} + k_{55-j} \cdot k_{54-j} + k_{29-j}$	29-28
14 to 15	j=0 to 1	$k_{80-j} + k_{79-j} \cdot k_{78-j} + k_{53-j}$	80-79
63 to 66	j=0 to 3	$k_{31-j} + k_{30-j} \cdot k_{29-j} + k_{4-j}$	4-1

have been carried out through the measurement of dynamic power consumption [73]. However, here we present two findings:

- How leakage current consumption of the trivium registers can be used to perform correlation attacks.
- And how these attacks can overcome the random body bias countermeasure introduced in the previous chapter.

In any case, all manner of attacks focus on the output of function $t1$. However, instead of assessing the power consumed by the charging or discharging of the input capacitance of register $s94$, we focus on the state of register Y and its leakage current consumption.

As an example, we can analyze the first 12 periods of the initialization phase. During these periods, the combinational block $t1$ evaluates according to the following function:

$$t1_j(K, IV) = S_{66-j} \oplus S_{171-j} = K_{66-j} \oplus IV_{78-j} \tag{6.12}$$

Where K_{66} and IV_{78} represent, respectively, the 66th bit of the secret key and the 78th bit of the Initialization vector, with j being the relative period of the Trivium, as depicted in Table 6.1.

Thus, given that register Y is a shift register, we can expect that, after m periods, with $0 \leq j \leq 11$, $m \leq MAX(j)$, the Hamming Weight stored within the register is going to be the sum of bits evaluated by function $t1$:

$$HW = \sum_{j=0}^{m-1} k_{66-j} \oplus IV_{78-j} \tag{6.13}$$

Ignoring, for the moment, the rest of bits already present or newly introduced into the Trivium state, equation (6.13), coupled with the equation of the leakage current consumed by a register array with a random body bias, forms a power model similar to those explored so far. That is, the leakage current consumed by the Trivium after m periods is:

$$I_{leak}(V_{bb}) = 288 \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot \sum_{j=0}^{m-1} k_{66-j} \oplus IV_{78-j} \tag{6.14}$$

Where 288 is the total number of bits of the Trivium. Thus, equation (6.14) can be used as the basis of a power model based on processed data by the algorithm as a function of the secret key.

With a power model such as this, it would be possible to perform Correlation Power analysis attacks in the manner described in previous sections. However, algorithmic noise would be an impediment. Thus, some slight considerations and modifications must be made.

Consider, firstly, that the Trivium does not produce algorithmic noise, that is, that the only bits that are evaluated or produced are the outputs of function t_1. This is evidently not true, as function t_2 and t_3 produce each one bit of what can be considered algorithmic noise every period of operation during the initialization phase. However, this assumption serves to illustrate the manner of the attack.

Consider, also, an attack on m bits of the secret key, beginning at period i, up to period $i + m$. During period i the Trivium finds itself in an unknown state, with its registers populated by bits of unknown value. We define the total Hamming Weight of the Trivium during this state as HW_i, which is an unknown random variable.

After m periods, the Trivium will find itself in a new state. Under the assumption of no algorithmic noise, the only new bits introduced in the trivium registers are those produced by function t_1, which depend exclusively on bits of the secret key and the IV. Thus, this state can be defined as:

$$HW_m = HW_i + \sum_{j=0}^{m-1} f_{i+j}(K, IV) \tag{6.15}$$

Where $f(K, IV)$ is any of the functions shown in Table 6.1. Thus, the total leakage consumption of the Trivium during period i and period $i + m$ can be expressed as:

$$I_{leak}(V_{bb})_i = 288 \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot HW_i$$

$$I_{leak}(V_{bb})_{i+m} = 288 \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot (HW_i + \sum_{j=0}^{m-1} f_{i+j}(K, IV)) \tag{6.16}$$

Given that both these states belong to the same overall execution of the algorithm, the value of the body bias, randomly adopted at the beginning of the encrypting process, is the same in both instances. Thus, as in the case of the AES, it is possible to obtain a bivariate power model by subtracting the leakage current consumed during period i from the leakage current consumed during period $i + m$. Performing this subtraction leaves the following expression:

$$I_{sub} = \epsilon(V_{bb}) \cdot \sum_{j=0}^{m-1} f_{i+j}(K, IV) \tag{6.17}$$

Where both the term $n \cdot I_0(V_{bb})$ and the contribution of the unknown initial state HW_i cancel out. With this new expression, it is again possible to use the PCC as a secret key discriminant, undoing the protective effect of the random body bias countermeasure.

Algorithmic noise

The Trivium, however, does produce algorithmic noise, and thus the above equation is not representative of the reality of its operation. For ease of simplification and a worst case scenario, we are going to consider that each bit of the Trivium is independent from each other, and follows an unbiased uniform probability distribution.

With this, we can consider, initially, five sources of noise:

- The edge of the shift registers, conforming three sources of noise, one for each of the three registers. That is, as bits reach the last flip-flop of each of the registers, they "disappear" the following period, generating Hamming Weight noise.
- Functions t_2 and t_3. These functions produce, each period, one bit of unknown value, thus accounting for one bit per period of noise each.

While it is clear that, under the assumption of bit independence with an unbiased distribution, the bits that reach the edge of the registers introduce noise readily arising from this distribution, it is worthwhile to examine the probability distribution of the bits produced by functions t_2 and t_3, as they don't necessarily produce an unbiased distribution.

We focus, then, on the probability distribution arising from the structure of the non-linear functions as seen in Fig. 6.5. It is easy to see that an XOR gate with equiprobable inputs produces equiprobable outputs. That is, $P[a \oplus b =$

$0] = P[a \oplus b = 1] = 1/2$. However, an AND gate has, under the same conditions, probability $P[Z = 1] = 1/4$ and $P[Z = 0] = 3/4$, where Z is the output. The output of this AND gate serves as the input for an XOR gate.

Table 6.2: Probability Distribution of second XOR gate

a b	$P[A] \cdot P[B] = P[Z]$	Z
0 0	$1/2 \cdot 3/4 = 3/8$	0
0 1	$1/2 \cdot 1/4 = 1/8$	1
1 0	$1/2 \cdot 3/4 = 3/8$	1
1 1	$1/2 \cdot 1/4 = 1/8$	0

Table 6.2 analyzes the probability distribution of the output bit of an XOR gate with inputs provided by another XOR gate (input a) and an AND gate (input b). It can be seen that the output is, again, unbiased ($P[Z = 0] = P[Z = 1] = \frac{1}{2}$). Thus, under the imposed probability distributions of the bits of the Trivium, the non-linear functions produce bits with a uniform discrete equiprobable distribution.

Thus, we can assume that all sources of noise produce bits with the same distribution.

Regarding the sources of noise, however, we can make some simplifications. We first note that registers X and Z contain more bits than periods are necessary to attack the entirety of the secret key. Therefore, regardless of which sub-key bits are under attack, the bits that "disappear" from these two registers are deterministic within each attack and do not introduce algorithmic noise.

This is not so for register Y. Depending on how many sub-key bits are attacked, IV bits initially loaded in position $s171$ reach the edge of the register after 6 periods, introducing noise. For simplification's sake, we are going to assume that, regardless of the number of key bits under attack, register Y always produces one bit of noise per period.

The inputs of functions $t3$ and $t2$ are, also, not always deterministic and, hence, also introduce noise.

Thus, there are three possible sources of noise, each producing m random bits, leaving Equation (6.17) post subtraction of HW_i as:

$$HW = \sum_{j=0}^{m-1} f_{i+j}(K, IV) \pm \sum_{r=1}^{3} HW_r \tag{6.18}$$

Where HW_r are random variables representing the Hamming Weight noise introduced, respectively, by the shift registers and the non-linear functions after m periods of activity.

The variance of the Hamming Weight of m independent bits following an equiprobable uniform probability distribution can be shown to be $\sigma^2_{HW} = \frac{m}{4}$.

Substituting Equation (6.18) into Equation (6.17) and calculating the PCC between I_{sub} and the Hamming Weight of the bits of interest, a worst case scenario of the ability to distinguishing the secret key can be found:

$$\rho_{I_{Sub},HW} = \frac{\sigma_{HW}}{\sqrt{\sigma_{HW}^2 + 3\sigma_{HW}^2}} = \sqrt{\frac{m}{m+3m}} = \frac{1}{2} \tag{6.19}$$

Thus, algorithmic noise can reduce the maximum PCC achievable, diminishing the ease with which the secret key can be identified.

Equation 6.19 represents the PCC when no contermeasure is applied. When a random body bias is taken into account, the equation becomes:

$$\rho_{I_{Sub},HW} = \frac{\mu_\epsilon}{2 \cdot \sqrt{\mu_\epsilon + \sigma_\epsilon^2 \cdot (4m+1)}} \tag{6.20}$$

Which, as can be seen, converges to the worst case scenario of equation 6.19 with trace averaging of the noise introduced by the countermeasure, much of which has already been eliminated through the bivariate subtraction. In fact, it can be seen in Fig. 6.6 that approximately 30 samples per plaintext suffice to average out the noise introduced by the random body bias countermeasure when the bivariate attack is contemplated. Figure 6.6 is obtained considering LVT, FBB transistors at 27 ºC.

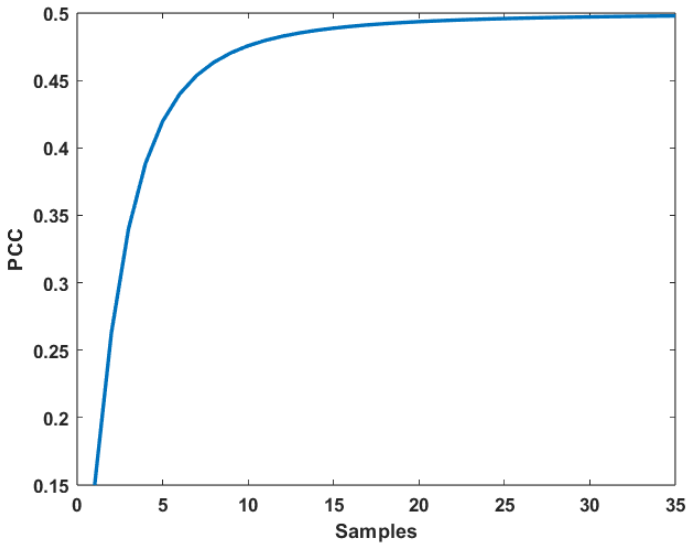

Figure 6.6: PCC between the magnitude I_{sub} and the Hamming Weight of interest in the Trivium as a function of samples per plaintext

6.2 Current Balancing Body Bias

With the considerations that have arisen regarding the vulnerabilities of the random body bias countermeasure we wish to explore whether another body bias scheme exists that can protect against leakage power analysis attack of register slices.

We propose a new countermeasure based on a simple observation. Throughout the text and the different analysis contemplated, it can be seen that the PCC between the leakage magnitude and the Hamming Weight of the bits of interest is directly proportional to the expected value of ϵ.

$$\rho_{I_{leak},HW} \propto \mu_{\epsilon} \tag{6.21}$$

That is, the disparity between the leakage current consumed by flip-flops storing a 1 and flip-flops storing a 0 is the main source of information leakage regarding the internal states of the cryptosystem when contemplating leakage power analysis attacks on register slices.

Thus, if a body bias scheme exists that can arbitrarily reduce the magnitude $\epsilon = |I_1 - I_0|$, while maintaining circuit functionality, this scheme could be used to hinder the extraction of the secret key, not by increasing the noise present in the cryptosystem, but by reducing the magnitude of the variable of interest.

We begin our assessment by designing a testbench for electrical simulations where, instead of considering that $V_{bbn} = -V_{bbp}$, we consider both of the potential body bias values to be independent from each other. As such, we perform transient analysis of the leakage current consumed by flip-flops storing a 1 and a 0 while performing independent and nested parametric sweeps of the magnitudes of V_{bbn} and V_{bbp}.

During this set of parametric sweeps we obtain a series of curves representing the leakage current consumed by a flip-flop when it is storing either possible value. In particular, we obtain the curves of ϵ as a function of both V_{bbn} and V_{bbp}. The results can be seen in Fig. 6.7.

It can be seen in Fig. 6.7 that there exists distinct pairs of body bias values in which ϵ becomes arbitrarily small, as depicted by the horizontal red line located at $\epsilon(V_{bbn}, V_{bbp}) = 0$ A. Thus, there exists a subset of values of the xy-plane defined by $(V_{bbn}) \times (V_{bbp})$ where $\epsilon \approx 0$.

In order to better interpret the results obtained from the simulation, the different curves are extracted with the help of Matlab fitting tools, and represented as a bivariate polynomial of degree n of the following form:

$$\epsilon(V_{bbn}, V_{bbp}) = c_{ij} \sum_{i=0}^{n} \sum_{j=0}^{n} V_{bbn}^{i} V_{bbp}^{j} \tag{6.22}$$
$$i + j \leq n$$

With the different c_{ij} terms being constant parameters of the polynomial and $n = 4$ sufficing to fit the curves with an $R^2 \approx 1$.

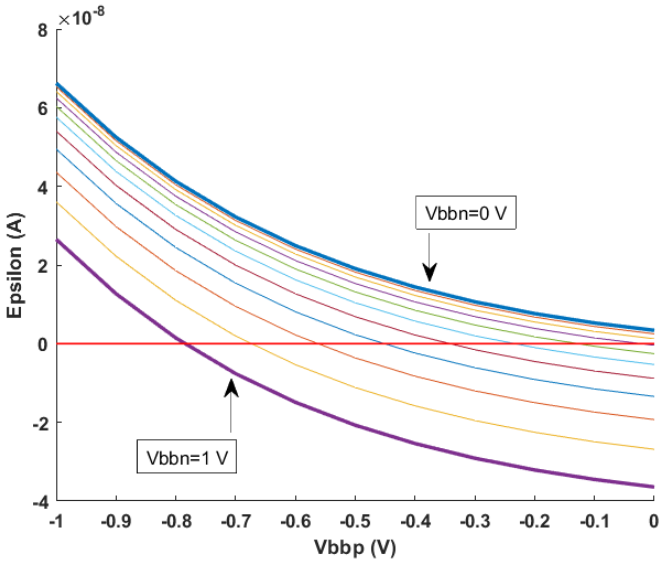

Figure 6.7: ϵ as a function of both V_{bbn} and V_{bbp} for LVT, flip-well transistors at 27 ^{o}C. Each curve represents a possible value of V_{bbn}, which ranges from 0 V to 1 V, in 0.1 V increments (from top to bottom). The horizontal red line indicates the points of zero crossing of ϵ.

We then implement an algorithm to solve this bivariate polynomial for the collection of pairs of points V_{bbn} and V_{bbp} such that $|\epsilon(V_{bbn}, V_{bbp})| \leq C$, where C being an arbitrary constant.

Setting $C = |1|$ nA, we obtain the following curves (Fig. 6.8).

We set the condition that $C = |1|$ nA somewhat arbitrarily, but mostly out of a conservative approach. At this point, without knowing exactly how such a body bias scheme would be implemented, we do not wish to assume that smaller values are attainable. After all, given that we are performing these analysis mostly through numerical simulations, one could be tempted to say that there exists values of body bias where $\epsilon = 0$ A, and conclude that a perfect countermeasure has been developed.

Nevertheless, Fig. 6.8 does show that pairs of values of body bias exist where the magnitude $\epsilon = |I_1 - I_0|$ does become arbitrarily small, while the flip-flops remain operational (they are able to store either value without issue).

At the same time, the two contour lines depicted in Fig. 6.8 can be parameterized as a function of each possible body bias. However, we focus on expressing the negative body bias, V_{bbp}, as a function of the positive body bias V_{bbn}.

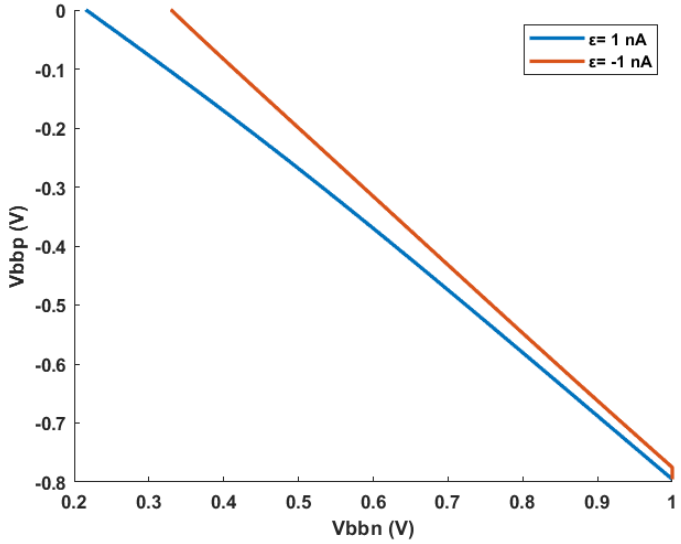

Figure 6.8: Reduced contour map of $\epsilon(V_{bbn}, V_{bbp})$. The lines represent the limits where $\epsilon = |1|$ nA, encasing an area where any possible combination of body bias values solves for $\epsilon \leq |1|$ nA.

$$V_{bbp}(V_{bbn}) = a_1 - b_1 \cdot V_{bbn}$$
$$V_{bbp}(V_{bbn}) = a_2 - b_2 \cdot V_{bbn} \tag{6.23}$$

Where the different a_i and b_i parameters are constants, depending on which of the two curves is chosen.

That is, the values of V_{bbp} that comply with the condition that $C = 1$ nA can be obtained through a linear expression of V_{bbn}.

This is important, as it allows the bivariate polynomial presented in Equation 6.22 to be reduced to an univariate polynomial that always solves for $\epsilon \approx |1|$ nA:

$$\epsilon(V_{bbn}) = c_{ij} \sum_{i=0}^{4} \sum_{j=0}^{4} V_{bbn}^{i} \cdot (a_1 - b_1 \cdot V_{bbn})^{j} \tag{6.24}$$
$$i + j \leq 4$$

This vastly simplifies numerical analysis.

With this initial assessment, we can propose another countermeasure based on a new body bias scheme, Current Balancing Random Body Bias (CBRBB).

For this, consider a cryptosystem that, at the beginning of each encryption process, chooses a value of the positive body bias at random and independently. After the random value of V_{bbn} is fixed, the value of the negative body bias, V_{bbp} is adjusted until the condition imposed is met. That is, until Equation 6.24 solves for $C \leq 1$ nA. In the limit of the condition, this is equivalent to solving Equation 6.2 once the value of V_{bbn} has been randomly selected.

We assume that V_{bbn} follows a discrete uniform distribution, again parameterized as a function of s_{max}, such that:

$$V_{bbn}(S) = V_{bbQ} + S \cdot \Delta V_{bb} \tag{6.25}$$

Where the different terms are defined as:

$$V_{bb_Q} = \frac{V_{bb_{min}} + V_{bb_{max}}}{2} \tag{6.26}$$

$$DR = V_{bb_{max}} - V_{bb_{min}} \tag{6.27}$$

$$\Delta V_{bb} = \frac{DR}{2s_{max}} \tag{6.28}$$

Note that depending on which of the two curves of Fig. 6.8, the values of $V_{bb_{min}}$ change, as not all the values V_{bbn} have solutions where $\epsilon(V_{bbn}) \leq 1$ nA. This can be seen in Fig. 6.9.

With these considerations we now have a model with which to determine the effectiveness of this proposed countermeasure.

To do so, we solve equations for the PCC between the Hamming Weight of the bits of interest and the bivariate leakage of the AES and Trivium (Equations 6.10 and 6.20) under noiseless assumption and in the presence of algorithmic and non-algorithmic noise. It is necessary to determine the expected value and variance of ϵ under these conditions. Given the model above derived, these can be determined numerically utilizing the following definitions.

$$Var(\epsilon(S)) = E[\epsilon(S)^2] - E[\epsilon(S)]^2$$

$$E[\epsilon(S)] = \frac{1}{2s_{max}+1} \sum_{i=-s_{max}}^{s_{max}} \epsilon(V_{bbn}(i), a_1 - b_1 \cdot V_{bbn}(i))$$

$$E[\epsilon(S)^2] = \frac{1}{2s_{max}+1} \sum_{i=-s_{max}}^{s_{max}} \epsilon^2(V_{bbn}(i), a_1 - b_1 \cdot V_{bbn}(i))$$

6.3 Results

In order to determine the effectiveness of the newly proposed countermeasure we take two approaches. Firstly, we calculate and plot the PCC of the bivariate power models of the AES and the Trivium presented above (Equations 6.10

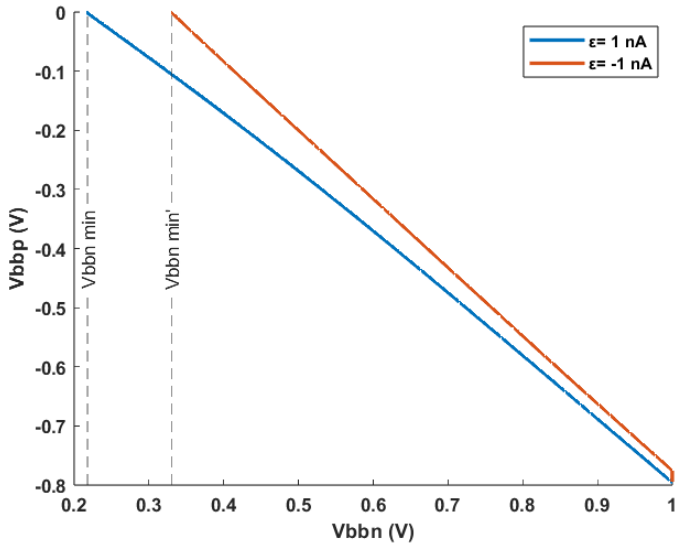

Figure 6.9: Reduced contour map of $\epsilon(V_{bbn}, V_{bbp})$ for registers implemented with LVT, FBB transistors at 27 ºC. The colored lines represent the limits where $\epsilon = |1|$ nA, encasing an area where any possible combination of body bias values solves for $\epsilon \leq |1|$ nA. The vertical lines showcase the inferior limits of V_{bbn} that can solve for the conditions

and 6.20, respectively). For each of these expressions, we compute the PCC contemplating a random body bias countermeasure. and the newly introduced current balancing random body bias countermeasure, comparing them.

We then go on to explore the effectiveness of the countermeasure under a variety of conditions, exploring the factors that affect the difficulty with which the secret key can be extracted.

Finally, we perform numerical simulations of parts of the AES and the Trivium under the new countermeasure to validate the theoretical results obtained through the different calculations of the PCC.

6.3.1 The AES and Current Balancing Body Bias

In previous sections we have seen that, under a bivariate power model of leakage consumption, the Pearson Correlation Coefficient between the magnitude I_{sub}, arising from the measurement and subtraction of the leakage current at two distinct temporal points, and the random variable Z, arising from the subtraction of the Hamming Weights of interest at those two point in time, can be expressed as:

$$\rho_{I_{sub},Z} = \frac{\mu_\epsilon}{\sqrt{\sigma_\epsilon^2 + \mu_\epsilon^2}} \tag{6.29}$$

This equation arises from the considerations of a a noiseless system under correct key assumptions. In order to distinguish the two countermeasures being compared, we will refer to the initial countermeasure, the Random Body Bias presented in Chapter 5, as Symmetric Random body bias, given that $V_{bbn} = -V_{bbp}$ at all times. The countermeasure presented in this chapter will be referred as Current Balancing (CB).

To compare the effectiveness of both countermeasures, we solve Equation 6.29 utilizing the conditions shown in Table 6.3. Equation 6.29 is solved for a variety of s_{max} values. The results can be seen in Fig.6.10.

Table 6.3: Countermeasure parameters

	Symmetric	CB
$V_{bbn,max}$	1 V	1 V
$V_{bbn,min}$	0 V	0.3 V
DR	1 V	0.7 V
V_{bbnQ}	0.5 V	0.65 V
n	8 bits	

It might seem at first that current balancing body bias presents significantly worse values (a higher PCC that can facilitate the acquisition of the secret key). This can be explained by noting that, under noiseless assumptions, the only source of noise is determined by the variance of ϵ and the variances of the Hamming Weights. Since in the Current Balancing body bias scheme the Dynamic Range of the body bias is limited, the variance of ϵ is smaller.

However, consider the case of a noisy system. That is, the leakage current equations at times t_1 and t_2 now present an additive White Gaussian Noise (WGN) component, B, with $\mu_b = 0$ and variance σ_b^2.

$$I_{leak}(HW, V_{bb}, t_1) = n \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot HW_{t1} + B$$
$$I_{leak}(HW, V_{bb}, t_2) = n \cdot I_0(V_{bb}) + \epsilon(V_{bb}) \cdot HW_{t2} + B$$

Under these conditions, the variance of $I_{sub} = I_{leak}(t_2) - I_{leak}(t_1)$ is modified to include the variance produced by the WGN, which, considering additivity, has a noise power of $2 \cdot \sigma_b^2$. As such, equation 6.29 is modified to include these sources of noise:

$$\rho_{I_{sub},Z} = \frac{\mu_\epsilon}{\sqrt{\sigma_\epsilon^2 + \mu_\epsilon^2 + 2\sigma_b^2}} \tag{6.30}$$

Figure 6.11 plots the results of the PCC for the Current Balancing and Symmetrical body bias schemes with a noise power of $\sigma_b^2 = -134$ dBW, the thermal

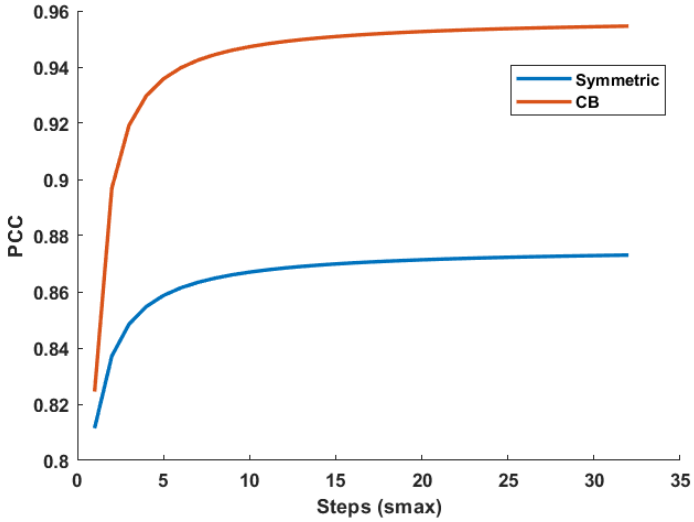

Figure 6.10: PCC between the Hamming Weight represented by the random variable Z and the bivariate leakage power model (Equation (6.5)) under noiseless conditions in the presence of a Symmetrical (blue) and a Current Balancing (orange) random body bias scheme, where ϵ is systematically kept at 1 nA

noise produced by a 1 ohm shunt resistor connected to a 1 V power supply, with measurements of up to a bandwidth of 10 MHz. This noise represents the pre-amplifications and pre-filtering measurements obtained in settings such as those described in [51]. Even though it is still somewhat arbitrary, it suffices, without loss of generality, for illustration purposes.

It can be seen that the PCC for the Current Balancing case is approximately one order of magnitude smaller than the symmetric body bias. Thus, the initial prediction of the countermeasure behavior is, at the very least, validated by numerical simulation.

By reducing the magnitude of μ_ϵ adopting a current balancing body bias, the signal of interest is 'covered' by sources of noise that we can expect to be present in a real system, such as thermal noise, $1/f$ noise, measurement noise, etc. Essentially, we are reducing the SNR by reducing the magnitude of the signal of interest by choosing an appropriate body bias point.

At the same time, if we fix the value of s_{max} to 32 and we plot the PCC against the number of averaged traces for the same noise conditions, we can expect to see an increase in the PCC as the noise is averaged out. However, the rate at which the PCC increases need not be the same for both body bias schemes.

Under noise averaging conditions, we can expect the PCC to be (with N being the number of averaged traces per plaintext):

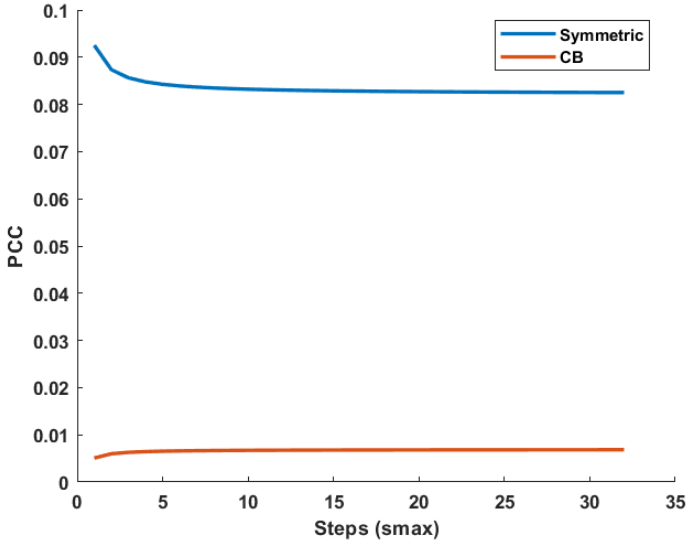

Figure 6.11: PCC between the Hamming Weight and the bivariate leakage power model (Equation 6.5) with non-algorithmic noise in the presence of a Symmetrical (blue) and a Current Balancing (orange) random body bias scheme, in the presence of non-algorithmic GW noise

$$\rho_{I_{sub},Z} = \frac{\mu_\epsilon}{\sqrt{\frac{\sigma_\epsilon^2}{N} + \mu_\epsilon^2 + 2\frac{\sigma_b^2}{N}}} \tag{6.31}$$

The results of noise averaging can be seen in Fig. 6.12. The figure shows that the Pearson Correlation Coefficient increases much more slowly when the Current Balancing body bias scheme is applied. That is, non-algorithmic noise severely dominates, and the current balancing countermeasure is much more resilient to noise averaging.

Algorithmic noise

We can further complete our analysis by contemplating the presence of algorithmic noise. So far, we have only taken into account the bits of interest ($n = 8$), and sources of non-algorithmic noise, in the form of GWN (B). However, attacks on the AES will have to contend with the presence of algorithmic bits.

Thus, a more realistic scenario considers an AES cryptosystem that processes $n+m$ bits, with n being the the bits of interest under attack, and m the rest of bits not pertinent to the attack that introduce algorithmic noise. In the AES, with $n = 8$, $m = 120$.

The bivariate power model under these conditions becomes:

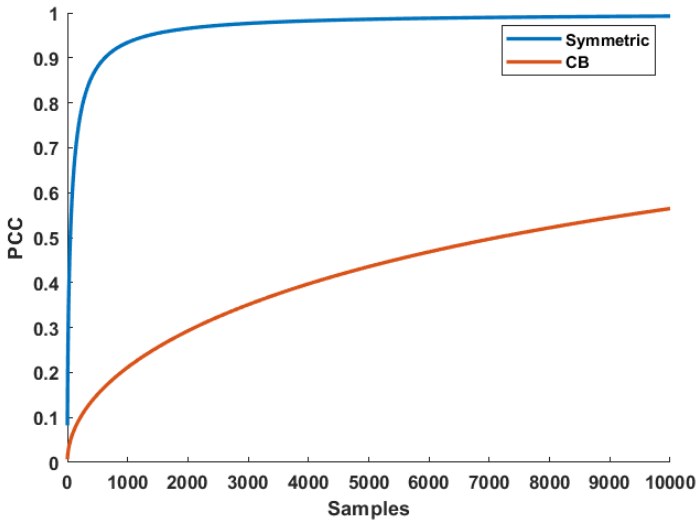

Figure 6.12: PCC between the Hamming Weight and the bivariate leakage power model (Equation 6.5) as a function of averaged number of traces in the presence of non-algorithmic GW noise with a Symmetrical (Blue) and a Current Balancing (Orange) random body bias scheme

$$I_{sub} = \epsilon(S) \cdot (Z_n + Z_m) + B \tag{6.32}$$

Where Z_n is defined in Equation (6.4) as the difference between the Hamming Weight of the bits of interest n after a round of encryption and before the round of encryption. Similarly, Z_m is defined as the difference between the Hamming Weight of the remaining m bits not pertinent to the attack after and before the same round of encryption.

$$Z_m = HW_{mt2} - HW_{mt1} \tag{6.33}$$

As in previous discussions, HW_{mt2} and HW_{mt1} might be uncorrelated but are not independent. In order to be able to treat them analytically, we make some assumptions regarding the distribution of Z_m.

$$\mu_{Zm} = 0 \tag{6.34}$$

$$\sigma^2_{Zm} = m/2 \tag{6.35}$$

With this, it can be shown that the PCC between the bivariate power model and Z_n is:

$$\rho_{I_{leak},Z_n} = \frac{\mu_\epsilon \sigma_{Z_n}}{\sqrt{(\sigma^2_{Z_n} + \sigma^2_{Z_m})(\mu^2_\epsilon + \sigma^2_\epsilon) + 2\sigma^2_b}} \tag{6.36}$$

From Fig. 6.3 we know that the variance of Z_n is not exactly $n/2$, but rather can adopt an approximate value between $n/2-0.6 \leq \sigma^2_{Z_n} \leq n/2+0.6$ depending on the secret key. The same can be said about $\sigma^2_{Z_m}$, where we can assume that it can adopt an approximate value between $m/2-15\cdot 0.6 \leq \sigma^2_{Z_n} \leq m/2+15\cdot 0.6$, where 15 are the number of bytes of algorithmic noise present in the AES. However, for ease of analysis, we will maintain the assumption of $\sigma^2_{Z_n} = n/2$ and $\sigma^2_{Z_m} = m/2$

We can plot Equation 6.36 as a function of non-algorithmic noise power σ^2_b, with different numbers of averaged traces N. We assume that, under trace averaging, the noise magnitudes of $\sigma^2_{Z_m}$, σ^2_ϵ, and σ^2_b are scaled by a factor of $1/N$.

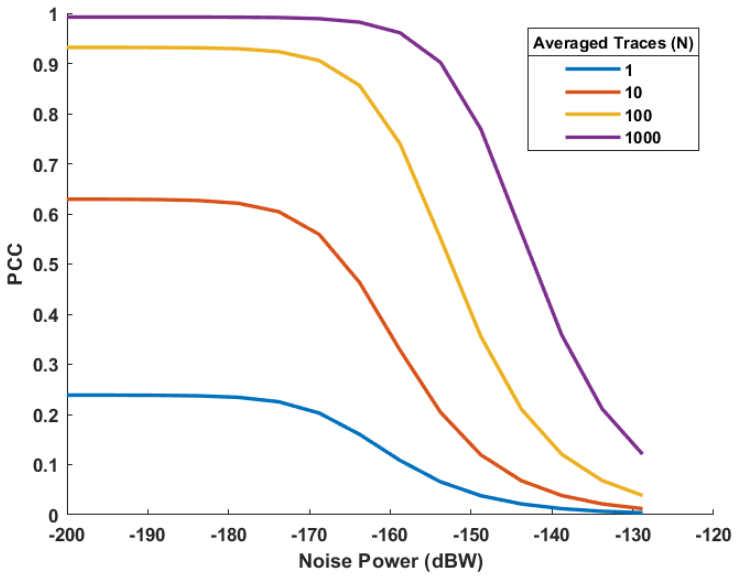

Figure 6.13: PCC between Z_n and the bivariate leakage power model with algorithmic and non-algorithmic noise (Equation 6.36) as a function of noise power under a current balancing boby bias, for different number of N averaged traces

It can be seen in Fig. 6.13 that the magnitude of the PCC under this conditions is heavily dependent on the magnitude of non-algorithmic noise power. For small noise powers (below -160 or -170 dBW depending on the number of averages) algorithmic noise dominates as a factor. In fact, it can be shown that for small magnitudes of gaussian noise power the countermeasure, under a bivariate attack, barely introduces noise. As such, the PCC between Z_n and

the leakage current can be approximated as:

$$\rho_{I_{leak},Z_n} = \frac{\sigma_{Z_n}}{\sqrt{(\sigma^2_{Z_n} + \frac{\sigma^2_{Z_m}}{N})}} = \sqrt{\frac{n}{n + \frac{m}{N}}} \tag{6.37}$$

Thus, only when non-algorithmic noise becomes comparatively high is current balancing body bias significantly effective.

We can further compare the effectiveness of the countermeasure considering both algorithmic and non-algorithmic noise by plotting the Equation 6.36 as a function of non-algorithmic noise power considering the Symmetric countermeasure and the Current Balancing countermeasure.

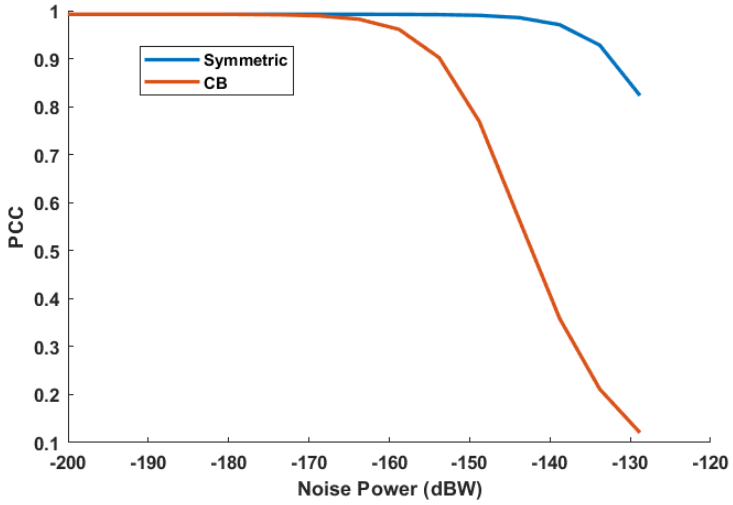

Figure 6.14: PCC between the Hamming Weight and the bivariate leakage power model with algorithmic and non-algorithmic noise (Equation 6.36), as a function of noise power under a a Symmetrical (blue) and a Current Balancing (orange) random body bias scheme for $N = 1000$ averaged traces.

It can be seen in Fig. 6.14 that with a Current Balancing body bias the PCC is much smaller for the same magnitude of noise even with the same amount of trace averaging.

Simulated CPA

We mount a numerically simulated CPA attack on a dummy cryptosystem that reflects the bivariate power model described above and summarized in Equation 6.32, under the countermeasure conditions established in Table 6.3.

In order to do so, we set a 128-bit secret key that represents a round key. The dummy cryptosystem comprises a round of encryption of the AES from the *MixColumns*, up to the *SubBytes* routine, without including the former [71].

That is, we consider that the whole state matrix after the $MixColumns$ routine is known to the attacker and directly consider this state as the input plaintext. Each of the 16 bytes of the plaintext are then XORed with their corresponding byte of the secret key. Each XORed byte is then fed to the AES S-Box and the result is again considered to be stored in the state matrix.

The attack is performed on $n = 8$ bits (1 byte). For each input plaintext i of interest (with $0 \leq i \leq 255$), N realizations of Equation (6.32) are numerically simulated. For each realization, the 15 remaining bytes of the plaintext are generated at random, each bit following a uniform probability distribution. The random variable S is also realized randomly following a discrete uniform distribution, constraint in such a way that a random body bias value that keeps $\epsilon \approx 1$ nA is obtained. Finally, for each of the N realizations, a white gaussian noise value following the distribution described above, for a noise power of -134 dBW is also produced.

The N realizations are then averaged:

$$\hat{I}_{leak}(Z_{ni}, \hat{Z}_m, \hat{S}) = \frac{1}{N} \sum_{j=1}^{N} \epsilon(s_j)(Z_{ni} + Z_{mj}) + B_j \tag{6.38}$$

Thus, a vector comprising 256 I_{leak} values, one for each possible plaintext is obtained. The PCC between this vector and the vector Z_n solved for each possible 8-bit secret key is calculated.

Table 6.4: PCC - Theoretical and Numerical Simulations of a 128 bit system in the absence of non-algorithmic noise

CB - No Gaussian noise			
	PCC		
N	Theo	CPA	Success Rate
1	0.239	0.196	0.6
10	0.630	0.645	1
100	0.932	0.933	1
1000	0.993	0.993	1

Tables 6.4 and 6.5 present the results of the PCC obtained for the secret key under attack for the simulated CPA and the theoretical value obtained through Equation (6.36) for different number N of averaged traces. It can be seen that, in both cases, as the number of averaged traces increase, the values of the PCC obtained through numerical simulations becomes closer to the theoretical values. The tables also include the success rate of secret key identification for 100 independent experiments.

At the same time, Table 6.6 presents the values of the expected value and variance of Z_m for increasing number of averaged traces. As N increases, the first and second moment of Z_m more closely resemble the theoretical values

Table 6.5: PCC - Theoretical and Numerical Simulations of a 128 bit system in the presence of non-algorithmic noise

CB - GWN -134 dBW			
	PCC		
N	Theo	CPA	Success Rate
1	0.010	-0.060	0
10	0.032	0.0325	0
100	0.100	0.0945	0
1000	0.302	0.188	0.7
5000	0.439	0.302	1

that had been previously assumed; namely, that $\mu_{Z_m} = 0$ and that $\sigma^2_{Z_m} = \frac{m}{2} = 60$, with m being the number of bits not under attack, 120 in this particular simulation.

Table 6.6: First and Second Moments of Z_m

Z_m Distribution		
N	μ_{Z_m}	$\sigma^2_{Z_m}$
10	4.400	25.82
100	-0.640	66.43
1000	0.250	62.20
5000	-0.018	59.16

The results show that in the presence of the countermeasure an approximately 5 fold increase is necessary to disclose the secret key with the same frequency.

6.3.2 The Trivium and Current Balancing Body Bias

We now develop the same theoretical and simulated analysis for the case of the Trivium.

We first calculate the PCC of a bivariate attack with algorithmic and non-algorithmic noise for the case of the Trivium.

$$\rho_{I_{Sub},HW} = \frac{\mu_\epsilon}{2 \cdot \sqrt{\mu_\epsilon + \sigma^2_\epsilon \cdot (4m+1) + 2\sigma^2_b}} \tag{6.39}$$

We calculate this metric for the case of a Symmetric Random Body Bias and a Current Balancing body bias under the conditions depicted in Table 6.3, with a noise power of -134 dBW. Again, in the presence of noise averaing, the

magnitudes σ_ϵ^2 and σ_b^2 are expected to scale with a factor $1/N$, with N being the number of averaging traces measured.

The results are shown in Figs. 6.15 and 6.16. It can be seen that the results are very similar in behaviour to the case of the bivariate power model of the AES. That is, for the same number of averaged traces, the PCC is much smaller in the case of CB body bias and this countermeasure is more susceptible (has a lower value of the PCC) to the presence of non-algorithmic noise. In this case, however, $n = 8$, and $m = 280$, for a total of $n + m = 288$, the total number of bits present in the Trivium.

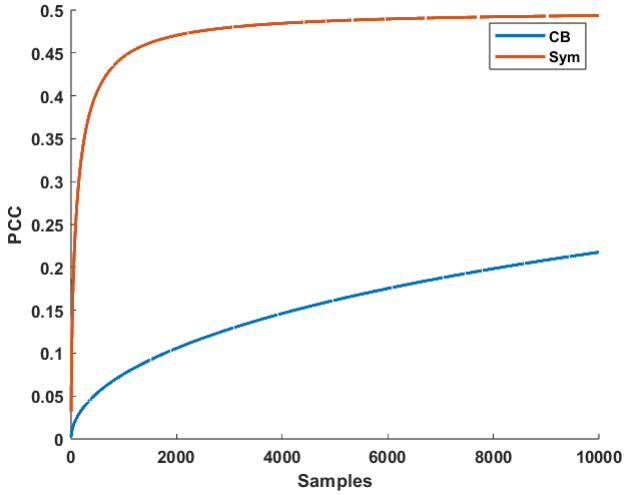

Figure 6.15: PCC between the Hamming Weight and the bivariate leakage power model of the Trivium with algorithmic and non-algorithmic noise (Equation 6.39), as a function of averaged samples under a Current Balancing (blue) and a Symmetrical (orange) random body bias scheme for a noise power of -134 dBW.

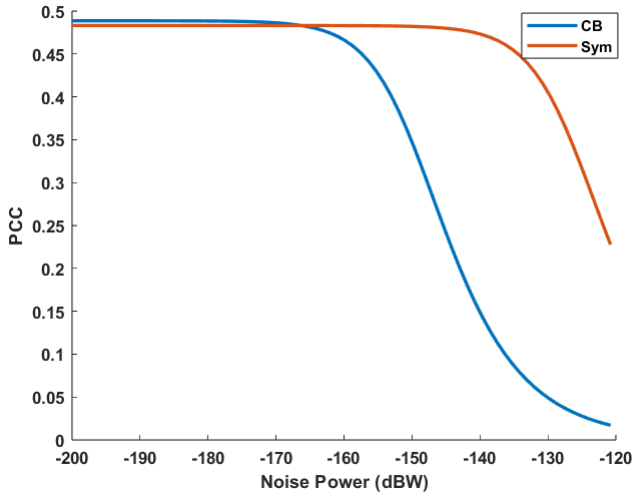

Figure 6.16: PCC between the Hamming Weight and the bivariate leakage power model of the Trivium with algorithmic and non-algorithmic noise (Equation 6.39), under a Current Balancing (blue) and a Symmetrical (orange) random body bias scheme for $N = 1000$ averaged traces.

Simulated CPA

We now perform a simulated CPA of the Trivium through numerical simulations of the cryptosystem implemented in a Simulink environment (Fig. 6.17).

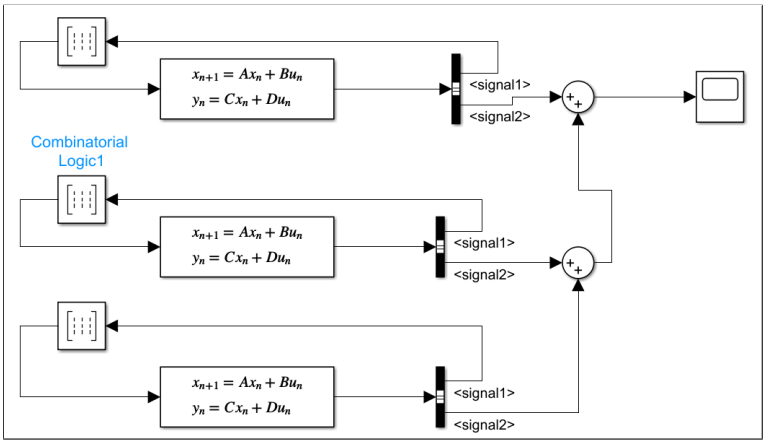

Figure 6.17: Block system representation of the Trivium implemented as a collection of discrete-time state space matrices. The scope saves the total Hamming Weight of the three registers.

Given the ease of Matlab to manipulate matrices, the shift registers of the Trivium are implemented as a collection of matrices forming a discrete-time state space. In a discrete-time state space system the state variable vector X is updated every clock period following a system of equations such as:

$$\begin{aligned} X[T+1] &= A \cdot X[T] + B \cdot \mu_{in}[T] \\ Y[T] &= C \cdot X[T] + D \cdot \mu_{in}[T] \end{aligned} \tag{6.40}$$

In our n bit shift register, the state variable X presents $n+1$ elements: the n bits that comprise the register, and their Hamming Weight (Equation (6.41)).

In order to implement a shift register, each element x_i for $i > 1$ adopts the value of element x_{i-1}, while variable x_1 updates according to the input μ_{in} (Equation 6.42)). The Hamming Weight of the shift register is the sum of all bits present in the register.

$$\boldsymbol{A} \cdot X[T] = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 \\ 1 & 1 & \dots & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \\ HW \end{bmatrix} \tag{6.41}$$

$$\boldsymbol{B}\mu_{in} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdot \mu_{in} \quad \boldsymbol{C}X[T] = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ HW \end{bmatrix} \tag{6.42}$$

The output $Y[T]$ of the state space system is the last bit of the shift register (x_n) and its HW, for monitoring purposes. Matrix D equals 0 in all instances.

The combinational non-linear functions of the Trivium (as seen in Fig. 6.5), whose outputs represent the variable μ_{in} for each shift register, are implemented using elements from the simulink boolean functions library.

By implementing a script to initialize the different state matrices it is then possible to load the secret key, desired IV and pertinent Trivium initial state. Once the system is prepared, it runs autonomously. To access the virtual side-channel representing leakage current, the Hamming Weight of the whole implementation is read each period of time.

We perform the simulated attacks as described in Section 6.1.2 in accordance with the procedures described in [73]. That is, for a given IV and an attack on m bits, the Hamming Weight output of the discrete-time state space is read at periods i and $i + m$ and subtracted.

$$HW = HW_i + HW_{i+m} \tag{6.43}$$

We then generate a leakage current virtual side-channel by calculating the equation:

$$I_{leak} = \epsilon(V_{bbn}(S)) \cdot HW + B \tag{6.44}$$

We do this by randomly generating a value for S at the beginning of the encryption process under the Dynamic Range constraints defined in Table 6.3. We then calculate the value of ϵ with the Symmetrical body bias or the Current Balancing body bias scheme.

Thus, depending on the countermeasure, V_{bbp} is obtained through two possible calculations:

$$\begin{aligned} V_{bbp} &= -V_{bbn} && \text{Symmetric} \\ V_{bbp} &= a - b \cdot V_{bbn} && \text{Current Balancing} \end{aligned}$$

As an example, we perform an attack on the first 8 periods of the initialization phase of the Trivium, attacking key bits k_{66} through k_{59}. For each attack, the Trivium is run 256 times. During each run, the pertinent bits of the IV are initialized to a value ranging from $0 \leq IV_{attack} \leq 255$. The rest of the IV is initialized to 0. Thus, the trivium is run $N \cdot 256$ times, where N is the numbers

Table 6.7: PCC - Theoretical and Numerical Simulations of a 128 bit system in the presence of non-algorithmic noise

CPA of Trivium		
	Success Rate	
N	Sym	CB
1000	1	.45
5000	1	0.77

of averaging samples and 256 is the number of all possible plaintexts in an 8-bit attack.

We consider in all cases a noise power of -134 dBW.

It can be seen in Table 6.7 that the CB countermeasure has a much smaller success rate of attack for the same number of samples, and that it is more resilient to trace averaging.

6.4 Temperature and Other Considerations

So far, temperature effects have been omitted. However, an important temperature effect worth discussing is the modification of the contour lines of ϵ as a function of temperature. Figure 6.8 shows the contour lines that met the imposed conditions for the registers under study at 27 C. However, as temperature increases, the contour maps of ϵ vary. Figure 6.18 shows the contour map $\epsilon(v_{bbn}, v_{bbp})$ at 75 C for the registers under study. It can be seen that, while presenting similar behaviour as the one at 27 C, the area that meets the criterion is reduced.

At the same time, registers with higher threshold voltage, implemented with non-flipped wells transistors driven through Reverse Body Bias (RBB), present much wider areas in their contour maps even at higher temperatures (Fig. 6.19 and 6.20). This means that it is much easier to obtain a body bias point where the conditions are met.

At the same time, note that if the countermeasure were ideal in form, that is, if the random body bias would always produce values of $\epsilon = |1|$ nA, the variance of ϵ, σ_ϵ^2, would be 0. Thus, a random body bias would not be required, only a system that could consistently reach the desired body bias point. This could free the design from true or pseudo-random number generators, which could heavily complicate the design.

At the same time, the results presented show that the countermeasure effectiveness is dependent on the magnitude of noise power present in the circuit or measuring system. While no such analysis has been done, the magnitude of ϵ must clearly also influence the effectiveness of the proposed scheme. In our analysis, we have restricted the conditions to a value of $\epsilon \approx 1$ nA. Without a designed and tested circuit, we do not know how much lower we can drive the

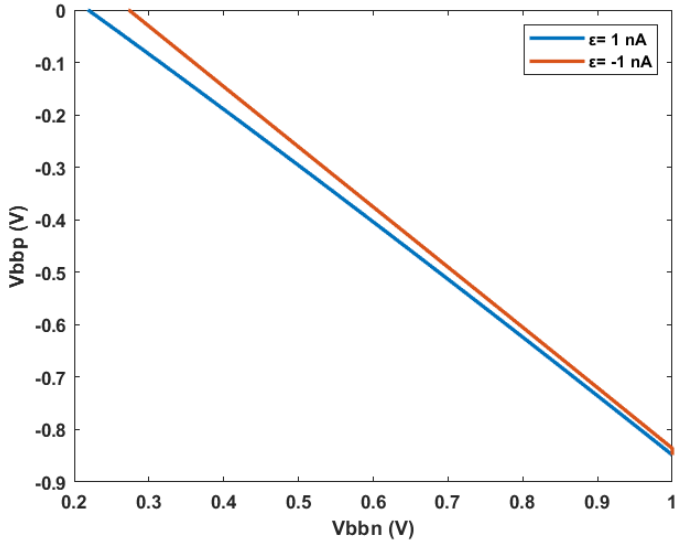

Figure 6.18: Contour map of $\epsilon(v_{bbn}, v_{bbp})$ at 75 C for the registers under study implemented with low threshold voltage, flipped well transistors. The lines represent the limits where $|\epsilon(v_{bbn}, v_{bbp})| \leq 1$ nA, encasing an area where every possible combination of V_{bbn} and V_{bbp} meet the imposed criterion.

difference $\epsilon = |I_1 - I_0|$. However, if it were possible to obtain lower values, the countermeasure would certainly be more effective.

This is the objective and content of the following chapter, where a circuit that attempts to obtain a body bias value that balances these currents is designed and tested with electrical simulations.

6.5 Conclusions

In this chapter, vulnerabilities to the random body bias countermeasure presented in Chapter 5 are identified, described and analyzed for the particular cases of the AES and the Trivium stream cipher. Some specifics are introduced regarding the AES and Trivium structure, and, to the best of the author's knowledge, the first description of the use of leakage power as a side-channel to attack the Trivium stream cipher is presented.

Through the derivation of the pertinent power models along the functionality of the cryptosystems described, vulnerabilities to the symmetric random body bias countermeasure are depicted, arriving at the conclusion that it might be unwise to rely on its use to protect cryptosystems.

As a response, another possible body bias scheme is explored, one in which the goal is not to introduce noise, but to reduce the magnitude of the signals

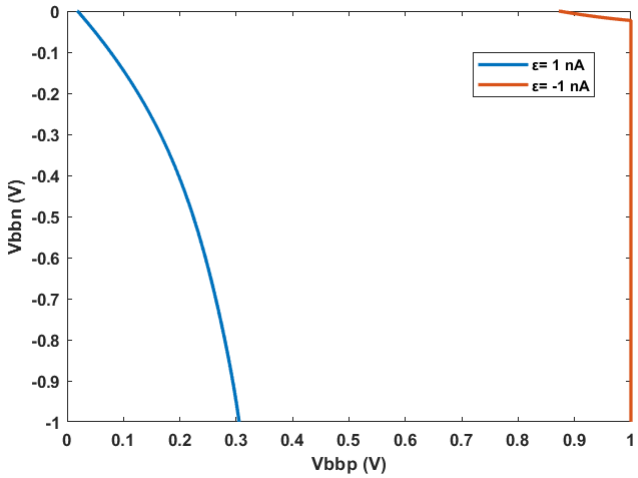

Figure 6.19: Contour map of $\epsilon(v_{bbn}, v_{bbp})$ at 27 C for the registers under study implemented with regular threshold voltage, reverse body bias transistors. The lines represent the limits where $|\epsilon(v_{bbn}, v_{bbp})| \leq 1$ nA, encasing an area where every possible combination of V_{bbn} and V_{bbp} meet the imposed criterion.

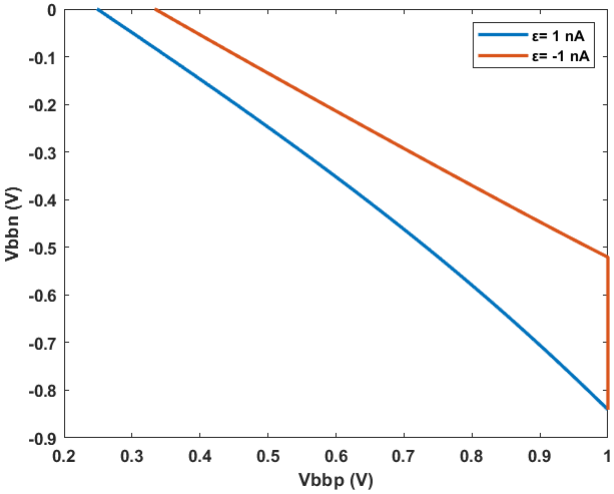

Figure 6.20: Contour map of $\epsilon(v_{bbn}, v_{bbp})$ at 80 C for the registers under study implemented with regular threshold voltage, reverse body bias transistors. The lines represent the limits where $|\epsilon(v_{bbn}, v_{bbp})| \leq 1$ nA, encasing an area where every possible combination of V_{bbn} and V_{bbp} meet the imposed criterion.

of interest, and to rely on the presence of non-algorithmic noise to hinder the capabilities of an attacker to extract the secret key.

The countermeasure is theoretically presented and analyzed, based on the bivariate model of the magnitude of ϵ as a function of both positive and negative body bias. By adopting a conservative stand, the countermeasure is analyzed in its capabilities to resist leakage power analysis attacks by formulating the appropriate statistical power model and calculating the PCC under a variety of conditions, including those that arise in bivariate power attacks.

Simulated CPA attacks are again produced to establish the rate of secret key identification, showcasing that this countermeasure is, at the very least, more protective than the symmetric random body bias counterpart under a bivariate attack.

The conditions for this theoretical countermeasure, that seeks a low magnitude of ϵ, are explored for different temperatures and also RBB, RVT transistors, where it can be seen that registers implemented with RBB, RVT transistors present wider ranges of body bias where ϵ becomes arbitrarily small. Thus, their use seems the most coherent with the purposes of this countermeasure.

Care is taken to choose a conservative stand and to not draw premature conclusions regarding overall effectiveness until a proof of concept circuit is designed and tested to determine the level to which ϵ can be effectively decreased. However, sufficient information is present to determine that the lower value that ϵ can achieve, the more effective the countermeasure would be.

Chapter 7

Current Balancing Body Bias Circuit Implementation

In the previous chapter we have seen that, despite the initial apparent effectiveness of a random body bias countermeasure, a bivariate attack can heavily undermine the noise introduced by the countermeasure proposed in Chapter 5.

As such, a new countermeasure is proposed in Chapter 6, based on the possibility of diminishing the disparities in leakage current presented in flip-flops storing distinct memory values.

This new countermeasure seeks a body bias point in which the circuit is still functional, but the leakage of information through the side channel that is static current is heavily diminished. Thus, rather than obscuring the secret key through the introduction of noise, the countermeasure hinders its identification by reducing the magnitude of the signal of interest.

Nonetheless, the analysis presented is purely theoretical, based on electrical and numerical simulations of the parameters under fixed conditions. Thus, while the analysis demonstrates that it is more difficult to identify the secret key when current balancing efforts are applied, it can make no assumptions on the feasibility of its implementation.

Thus, this chapter explores the design and implementation of a circuit that attempts to seek a body bias point in which the leakage of information is minimum.

The chapter introduces the conceptual requirements to implement such a circuit and goes on to showcase the design procedure followed. It then continues with the results obtained in terms of its functionality and ability to act as a countermeasure. The chapter then explores the effect of variability arising from process and mismatch variations of registers, seeking to determine how effective the countermeasure is when variability arising from the manufacturing process is taken into account.

7.1 Circuit Implementation

In order to implement such a system, various circuits would be necessary. The most obvious of these, is a body bias generator, able to produce the desired levels of body bias before each encryption process takes place.

At the same time, a sort of control system would be needed, which would inform the body bias generator to stop increasing the body bias voltage values once the desired value has been reached.

Finally, and perhaps the most critical given its complexities, a Digital to Analog Converter whose input is, ideally, a true random generator that can produce integer values between 0 and $2 \cdot s_{max}$. Given the obvious complexities of such a circuit, we are going to focus on the body bias generator and control system.

7.1.1 Body Bias Generator

The studied implementation of a body bias generator within this research group predates the development of this work. As such, not much of the theoretical background and their overall design is going to be presented here. Some chosen literature can be found in [74] [75].

The circuit of choosing is a CMOS cross-coupled charge pump with the structure seen in Fig. 7.1. The depicted figure shows a two-stage Cross coupled Charge pump.

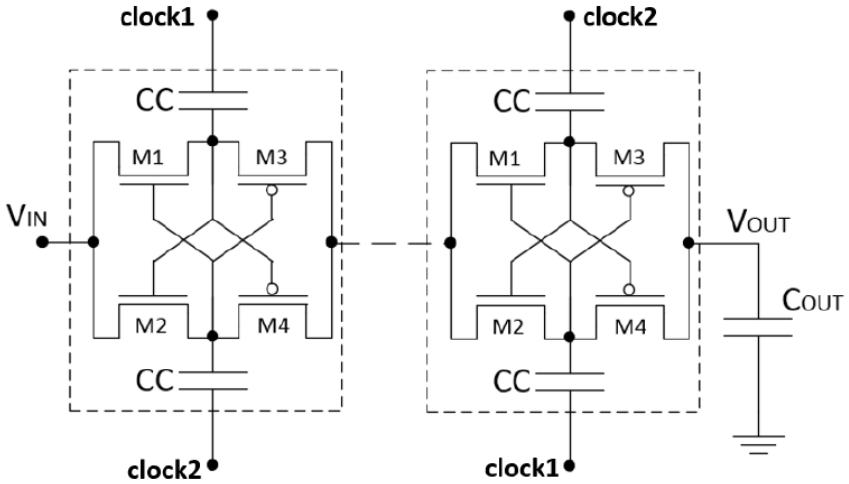

Figure 7.1: Schematic representation of a two-stage CMOS Cross-Coupled Charge Pump

These circuits act as voltage multipliers, with a transfer function, under ideal

conditions, of:

$$V_{out} = (n+1) \cdot V_{in} \tag{7.1}$$

Where n is the number of stages.

The circuit operates by transfering charge between the input V_{in} and fly-capacitors CC (or between fly-capacitors), whose bottom plates are connected to clock drivers. After a fly-capacitor has been charged, its clock signal, previously driven low during the charge cycle, becomes high. This increases the voltage seen by the charge stored in the capacitor, allowing it to being transferred at a higher potential onto the next capacitor.

In our design, however, for ease of modularity, we do not employ this exact structure. The circuit topology remains the same, but the input V_{in} is connected to ground. This transforms the transfer function into:

$$V_{out} = n \cdot V_{clk} \tag{7.2}$$

Where n is the number of stages, while V_{clk} is the voltage of the clock signal presented by the drivers.

This allows the use of the same design to generate the positive and negative charge pumps. The positive charge pump has been seen in Fig. 7.1. This same design can be reused for the negative charge pumps, where the only thing needed is the mirror image of the original charge pump. By inverting the flow of charge, the charge-pump "extracts" charge from the load capacitor, and injects it into ground. If the input of the positive charge pump, connected to ground, is close to the "output" of the negative charge pump, the charge dumped by the negative charge pump is picked up by the positive charge pump, reducing the possibility of generating bulk noise.

The charge pump is implemented with LVT, FDSOI transistors. The fly capacitors are, however, bulk transistors, and are thus voltage dependent. Their capacitance varies between 30 fF and 40 fF for an area of 4 μm^2 between 0.3 and 1 V.

Initially designed for ultra low power applications, the charge pumps are designed with a driver voltage V_{clk} of 0.3 V, intended to achieve a voltage of approximately 1 V. The ideal output voltage cannot be achieved due to parasitic capacitances [76].

Since the circuit requires to drive two sets of fly capacitors (those for the positive charge pump and those of the positive charge pump) and two clock phases, two sets of drivers capable of driving a total capacitance of approximately $2 \cdot (30 + 35 + 40)$ fF are needed.

We first design a single inverter, including its layout, and extract it to calculate the approximate total input capacitance (Fig. 7.2). We determine its input capacitance through simulations that equalize the path effort with a reference capacitance. The input capacitance of the inverter of Fig. 7.2 is approximately 0.6 fF.

With the input capacitance of the inverter, and the total capacitance incurred by the charge-pump fly-capacitors, we can optimally design the chain of

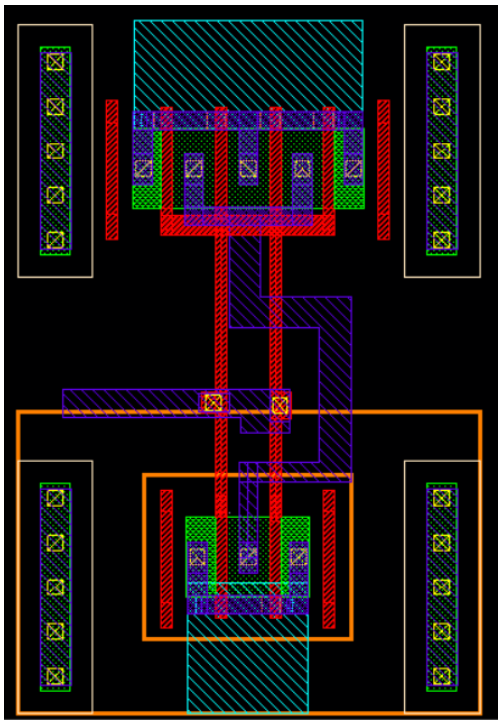

Figure 7.2: Layout of a designed single inverter

inverters for minimum delay. The chain of inverters ends up having 4 stages, each subsequent stage with 4 times the driving capability of the previous one.

In order to implement the charge pumps, we also require a phase generator able to produce two non overlapping clock phases. We base our design on the studies presented in [77].

The layout of the non-overlapping clock generator can be seen in Fig. 7.3.

The layout of the negative charge pump and drivers can be seen in Fig. 7.4 and 7.5.

The total design, seen in Fig. 7.6, has an approximate area of $58 \times 72\ \mu m^2$.

The double charge pump system was able to be implemented in die, and some measurements regarding their power consumption made. With an input voltage V_{clk} of 0.3 V, and an output voltage with capacitive load of $|V_{out}| = 1.01$ V (as they produce both positive and negative voltage) , the charge pumps, including the non-overlapping phase generator and driver chain, consume a total of, approximately, 2.4 μW.

They are able to produce any range of voltages between 0 and |1| V, although to be able to stop at a given value a control system is required.

Thus, in principle, the charge pump can act as a body bias generator.

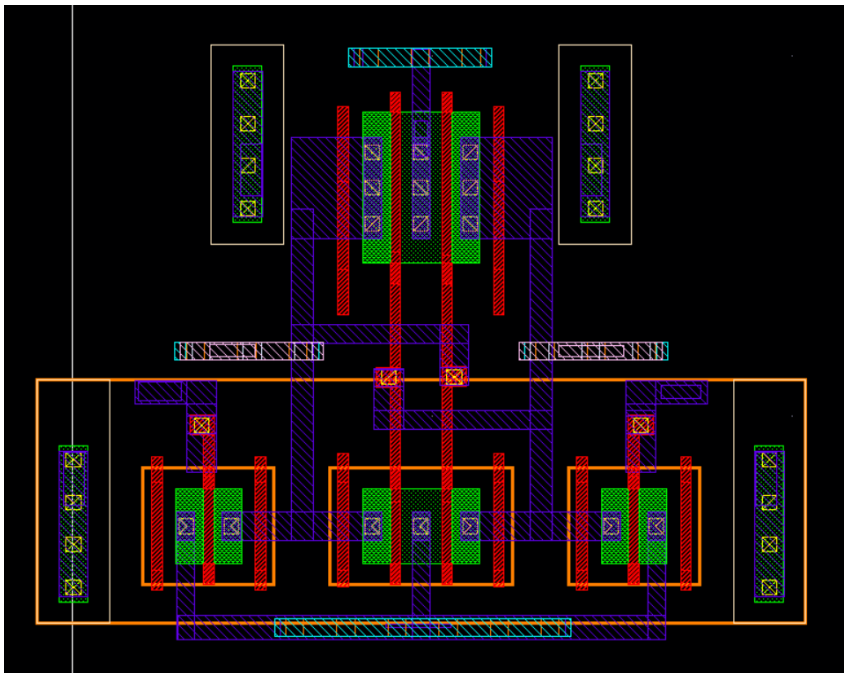

Figure 7.3: Non-overlapping two phase clock generator layout

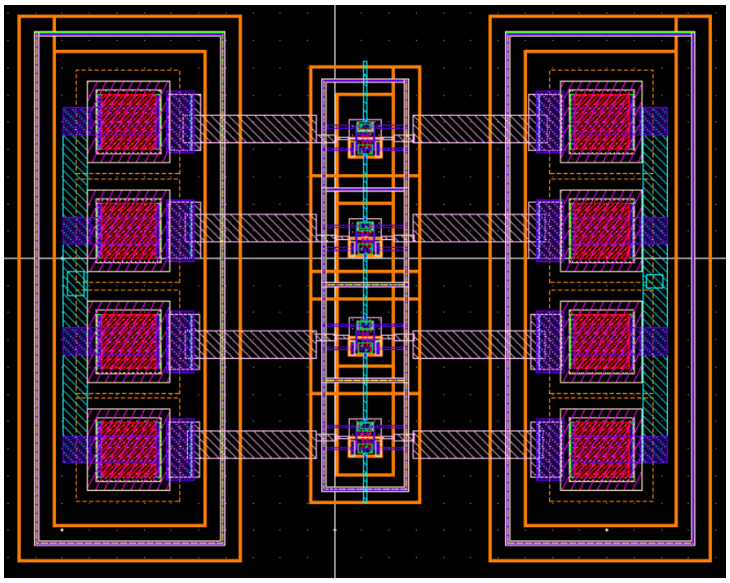

Figure 7.4: Layout of the negative charge pump

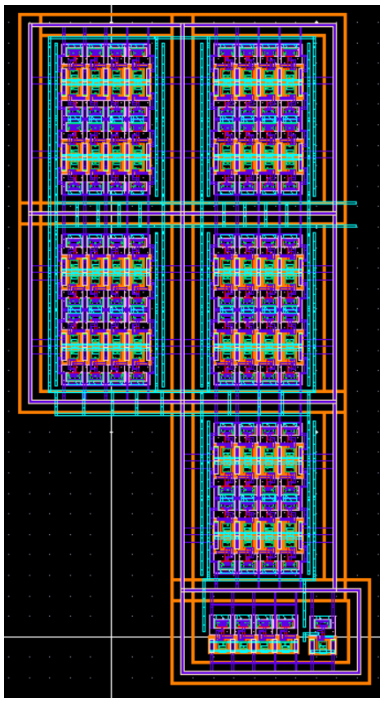

Figure 7.5: Layout of the driver chain

7.1.2 Hysteresis Control

In order to be able to stop the charge pumps at a given reference value, a control system is required. In order to do so, we opt to use a pulse skipping control of the charge pump clocks, so as to stop the switching of the drivers and their charge transfer capabilities. We settle for an hysteresis control to avoid continuous switching.

Given the notion presented in this chapter, we would be interested in an hysteresis window ΔV_H such that:

$$\Delta V_H = \Delta V_{bb} = \frac{DR}{2s_{max}} \tag{7.3}$$

Thus, the charge pump limits the body bias, for any particular value, within the range imposed by the conditions of S, pertaining to the countermeasure. That is, for a given value of $S = s$, the charge pump maintains the body bias between the values $[V_{bbQ} + s \cdot V_{bb}, V_{bbQ} + (s+1) \cdot V_{bb}]$

We derive the notions of a CMOS hysteresis comparator from [78] (see Fig. 7.7).

The problem with the utilization of these types of hysteresis comparators is that the tail transistor can easily leave the saturation condition once the common mode voltage of the differential input becomes too high. Thus, neither

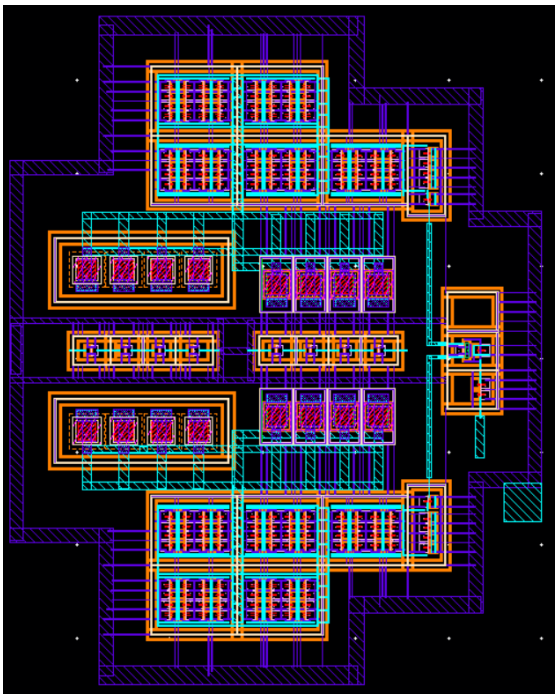

Figure 7.6: Full layout of the positive and negative Body Bias generator

the hysteresis range nor the functionality is guaranteed for the entirety of the dynamic range in which we are interested.

Thus, a pre-amplification stage is used to decouple the common voltage mode from the hysteresis comparator. This notion is derived from [79].

The outputs of the differential pair of Fig. 7.8 feed the inputs of the hysteresis circuit of Fig. 7.7.

Using a differential pair as a pre-amplification stage does not guarantee functionality across the entirety of the dynamic range, but it does guarantee a more or less constant hysteresis window. The hysteresis conditions can be shown to be [78]:

$$V_H = V_{gs2} - V_{gs1} \tag{7.4}$$

Given that:

$$\begin{aligned} V_{gs2} &= V_{dd} - \left(\frac{1}{R}\right) \cdot I_2 \\ V_{gs1} &= V_{dd} - \left(\frac{1}{R}\right) \cdot I_1 \end{aligned} \tag{7.5}$$

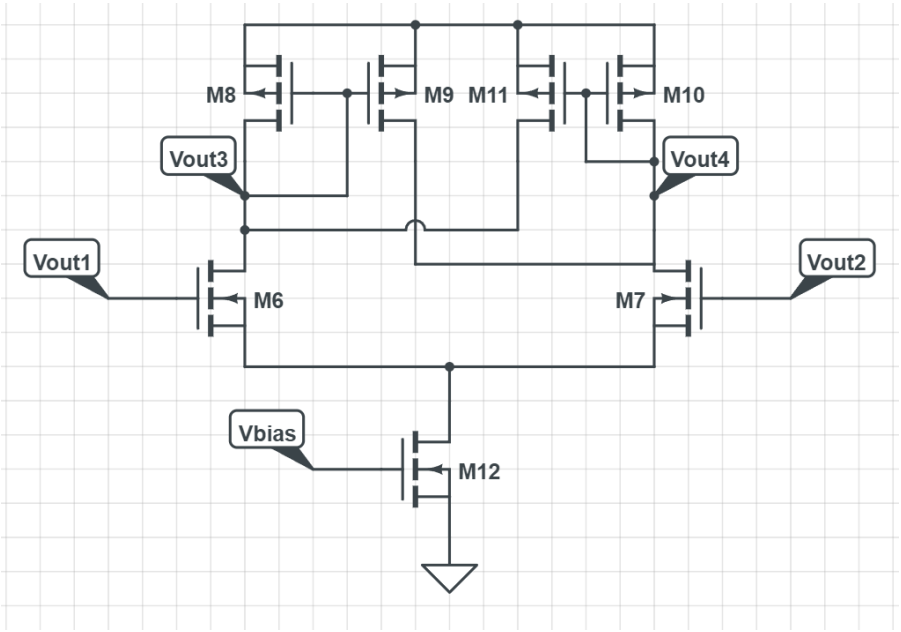

Figure 7.7: NMOS hysteresis comparator

Where R is the equivalent resistance of the PMOS active loads connected as diodes of the differential pair, approximately equal to $1/g_{m3}$=$1/g_{m4}$, where g_m are the transconductances. Also, for a differential pair assumed to be working in saturation:

$$\begin{aligned} I_2 &\approx \frac{I_{ss}}{2} - I_{ss} \cdot \frac{v_d}{2v_{ov}} \\ I_1 &\approx \frac{I_{ss}}{2} + I_{ss} \cdot \frac{v_d}{2v_{ov}} \end{aligned} \tag{7.6}$$

This condition is valid for small values of v_d, the input differential value of the pre-amplification pair. At the same time, v_{ov} and I_{ss} are, respectively, the overdrive voltage of the input transistors of this stage, and the tail current.

Sustituing 7.6 into 7.5 and into 7.4 we have:

$$V_H = \frac{v_d}{v_{ov}} \cdot R \cdot I_{ss} \tag{7.7}$$

Solving for v_d, we have:

$$v_d = \frac{V_H}{R \cdot I_{ss}} v_{ov} \tag{7.8}$$

What Equation 7.8 tells us is that, for a given V_H condition designed for the CMOS hysteresis comparator, the actual hysteresis as seen from the pre-

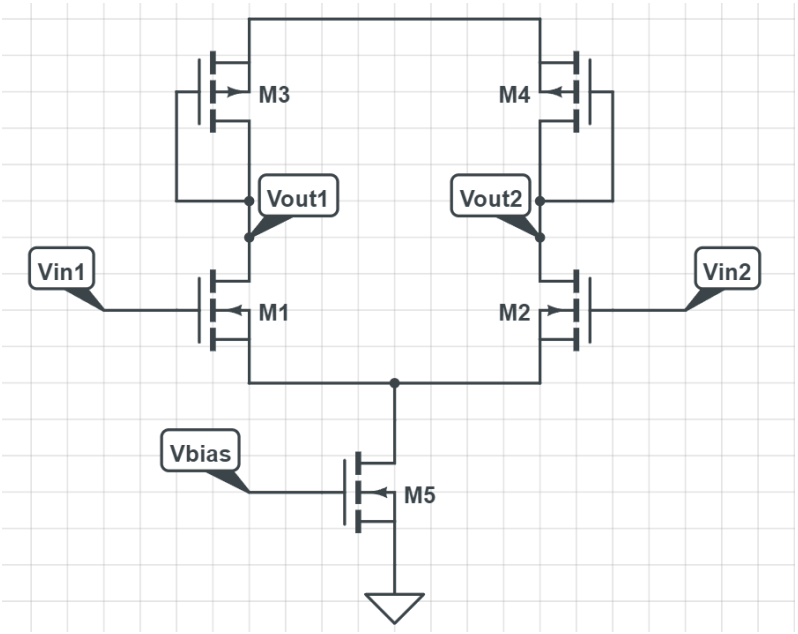

Figure 7.8: Pre-hysteresis stage schematic

amplification stage, depends on the overdrive value of the input transistors. For small values of v_{ov}, v_d remains relatively constant, albeit with a linear progression.

To solve the problem of saturation, we design a complementary version of the hysteresis comparator and the differential pair with PMOS as input transistors and NMOS as active loads. A push-pull output is used for both pairs of hysteresis + pre-amplification to produce a single ended output.

With this, we are able to implement a rail to rail hysteresis comparator with relatively stable hysteresis window. To test the hysteresis window, we set one of the inputs V_{in2} at a given value, and sweep the value of V_{in1} from $(V_{in2} - 50mV)$ to $(V_{in2} + 50mV)$.

The maximum hysteresis window is obtained for $V_{in2} = 50$ mV, with a hysteresis window of $\Delta V_H = 40$ mV. The minimum hysteresis window is obtained for $V_{in2} = 950mV$, with a hysteresis window of $\Delta V_H = 26mV$.

Although the hysteresis circuit is not implemented at the layout level, simulations of its schematic show that the circuit consumes a maximum power of approximately 118μW.

Some results of the combination of the charge-pump plus hysteresis comparator can be seen in Fig. 7.9. The purple and blue waveforms represent, respectively, the output of the positive and negative charge pumps. The green waveform is that produced by the hysteresis comparator for a reference value of $V_{ref} = 275$ mV.

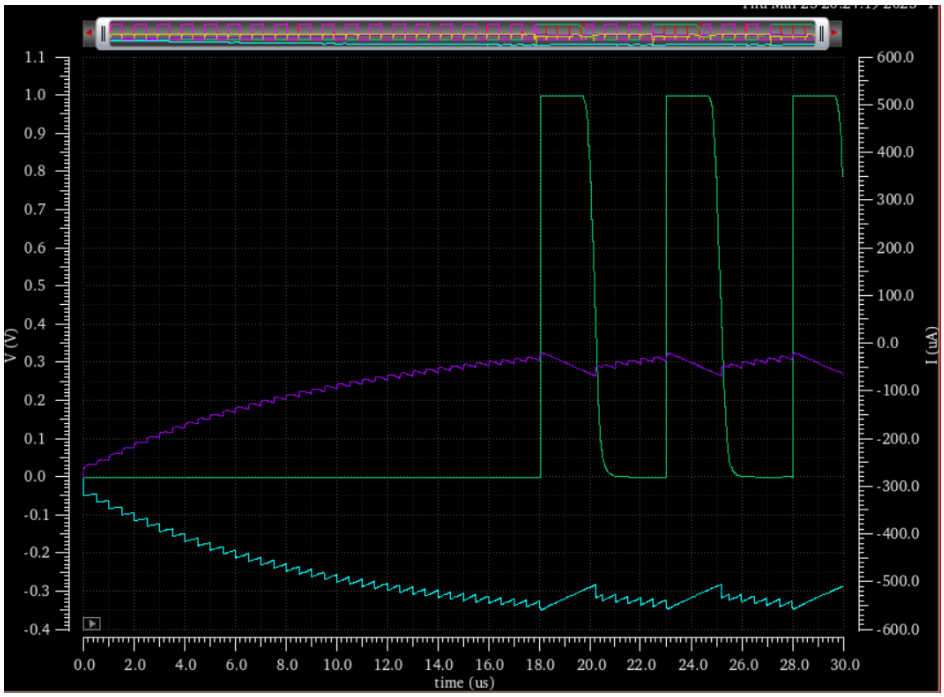

Figure 7.9: Charge-pump pulse skipping control through the hysteresis comparator

It can be seen that the circuit is able to maintain the output of the charge pumps around a given reference value, as was intended.

7.2 Current Balancing - Initial Exploration

If one wishes to balance currents, it comes as no surprise that one must, somehow, measure current. Further, since we want to reduce current disparities, it must be possible to also compare these currents. At the same time, the result of this comparison must be able to inform an action on the body bias of the registers present on the cryptosystem under study. This action must, at the same time, be well informed, responding to a requirement that arises from the aforementioned comparison between currents.

From these succinct conceptualizations one can gather various facts and translate them into modules of a circuit system.

With these we can posit the following requirements:

- We require a current sensor, or a voltage sensor succeeding a transducer with a preferably linear relation between current and voltage.
- We require a driver that can charge and/or discharge the body bias wells of the circuits of interest.
- We require a comparator of sorts.
- We require of a control unit which informs the response of the driver module when presented with information arising from the sensor unit and/or comparator.

The control unit and comparator can be thought of once the rest of the implementations are known, and the drivers have already been showcased in the previous section.

The question remains what the best or, at the very least, functional implementation could constitute the current or transducer+voltage sensor.

Initially two possibilities were explored and while we are not going to fully detail the failed implementation of the two, we are, at the very least, going to showcase why it fails.

7.2.1 Threshold-based Voltage reference

In [10], it was shown that the relation between the threshold voltage and the body bias of individual transistors is linearly related. As such, one possibility to attempt to establish a controlled system that attempts to fix both positive and negative body bias at a certain point is to use circuits whose voltage output is linearly dependent on both the body bias and threshold voltage.

One of these circuits are threshold-referenced current sources, that can be utilized as voltage references (Fig. 7.10)

It can be seen ([80]) that, analyzing the lower loop through KVL, the output current can be found as:

$$I_{out} = \frac{V_{GS_1}}{R_2} \tag{7.9}$$

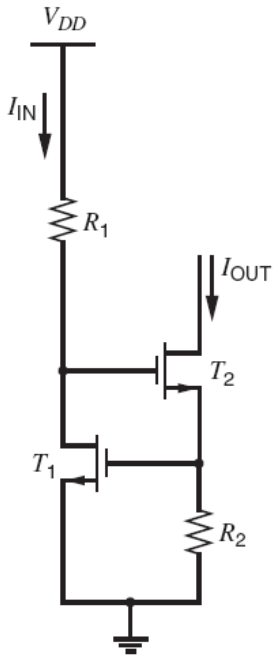

Figure 7.10: Schematic of a threshold-referenced current source

This expression can be rewritten as:

$$I_{out} = \frac{v_{ov1} + V_{th}}{R_2} \tag{7.10}$$

Where v_{ov}, assuming that transistor T_1 operates in the limit of saturation, can be expressed as:

$$v_{ov1} = \sqrt{\frac{2I_{IN}}{k(W/L)_1}} \tag{7.11}$$

If transistors T_1 is wide enough for the current flowing through it, the value of v_{ov1} can be made low enough so as to have little influence on I_{out}. Thus, assuming that $v_{ov1} << V_{th}$:

$$I_{out} \approx \frac{V_{th}}{R_2} \tag{7.12}$$

Consider now, the possibility, to modify the Threshold voltage of T_1, and only T_1. Consider, at the same time, that the relation between the threshold voltage of T_1 is linearly dependent on its body bias, according to an expression of the form (from [10]):

$$V_{th} = V_{th0} - m \cdot V_{bbn} \tag{7.13}$$

Where m is a constant slope. In this case, Equation 7.12 could be expressed as:

$$I_{out} = \frac{V_{th0}}{R_2} - \frac{m}{R_2} \cdot V_{bbn} \tag{7.14}$$

And, measuring the output voltage of the circuit in 7.10 at the R_2 terminal:

$$V_{out} = V_{th0} - m \cdot V_{bbn} \tag{7.15}$$

Obtaining, thus, a circuit whose output voltage is linearly dependent on the level of body bias applied.

The PMOS counterpart of this circuit can be seen in Fig. 7.11, and the $V_{out} - V_{bbp}$ relation can be seen in Fig. 7.12. In Fig. 7.12 it can be seen that the output voltage is also a linear function of V_{bbp}, such that:

$$V_{outp} = V_{th0p} - m \cdot V_{bbp} \tag{7.16}$$

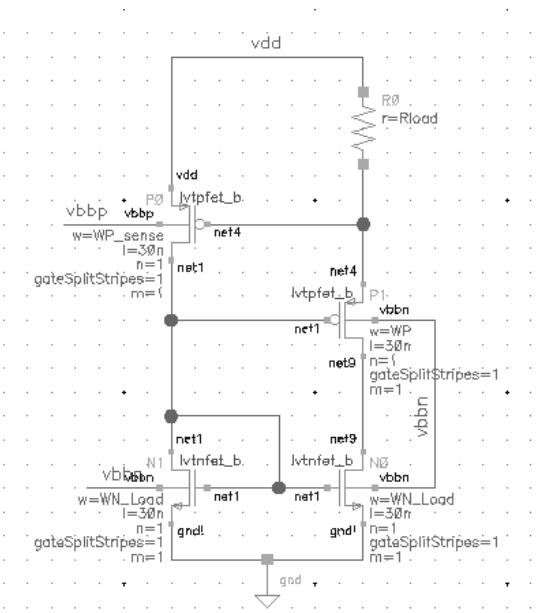

Figure 7.11: Schematic representation of a PMOS threshold-referenced current source / voltage regulator

Going back to Equation 6.2, we have also seen that under certain conditions, for an N-well body bias (V_{bbn}) voltage above a certain value (Fig. 6.9), the P-Well body bias (V_{bbp}) required to meet the criteria of $\epsilon \approx |1|$ nA is a linear function of this very same V_{bbn}, such that:

$$V_{bbp} = a_1 - b_1 \cdot V_{bbn} \tag{7.17}$$

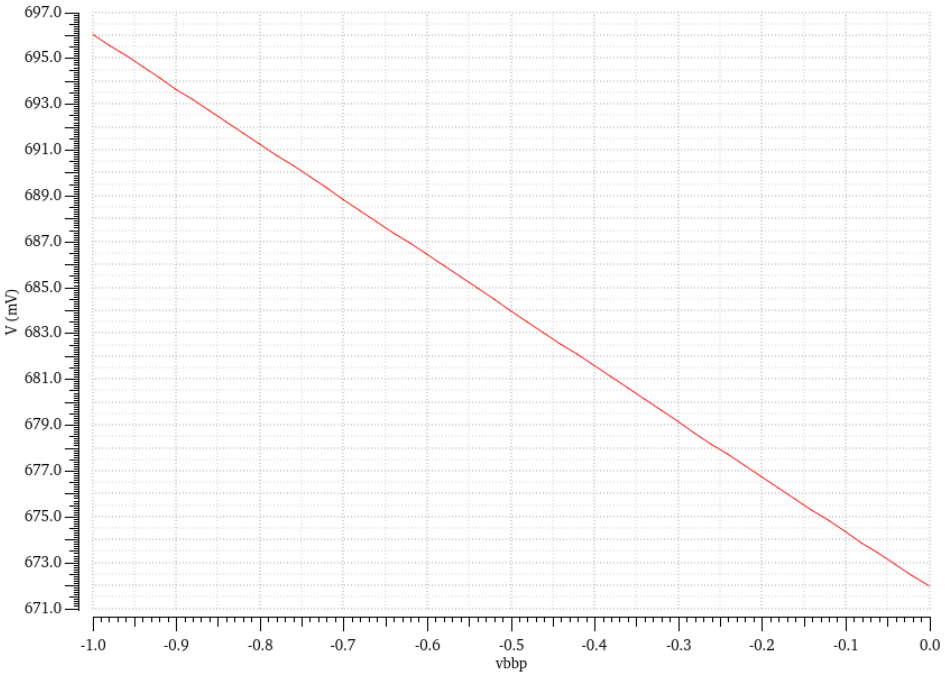

Figure 7.12: Output voltage vs P-Well body bias value (V_{bbp}) of the circuit depicted in Fig. 7.11. The voltage is measured at the node label as $net4$.

Thus, consider a reference value of V_{bbn}, which we will call $Vbbn_{ref}$, which could be, for example, 0.5 V. Under these conditions, the required V_{bbp} to meet the criteria of current balancing would be:

$$V_{bbp} = a_1 - b_1 \cdot Vbbn_{ref} \tag{7.18}$$

Substituting Equation 7.18 into Equation 7.16 we obtain the following:

$$\begin{aligned} V_{outp} &= V_{th0p} - m \cdot (a_1 - b_1 \cdot Vbbn_{ref}) \\ V_{outp} &= (V_{th0p} - m \cdot a_1) + m \cdot b_1 \cdot Vbbn_{ref} \end{aligned} \tag{7.19}$$

Where the first term of the right $(V_{th0p} - m \cdot a_1)$ is the y-intercept and $m \cdot b_1$ is the new slope.

What Equation 7.19 tells us is that when the the output voltage of the circuit of Fig. 7.11 equals that of $V_{out}(V_{bbn} = Vbbn_{ref})$, through the modification of the body bias V_{bbp} of Transistor P0, the body bias condition of current balancing would be met. Thus, this value of V_{outp}, which we can refer to as V_{ref}, can be used in a control system represented by the block diagram of Fig. 7.13.

Figure 7.14 depicts a simple possible implementation of the control system, in which the transfer function is represented by the gating of the clock that

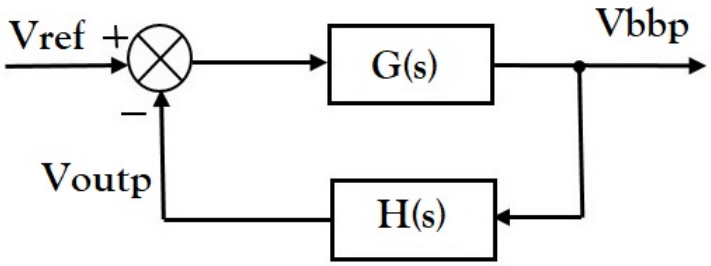

Figure 7.13: Block Diagram representing the possibility of implementing a control system to reach the desired value of V_{bbp} through the circuit depicted in Fig. 7.11

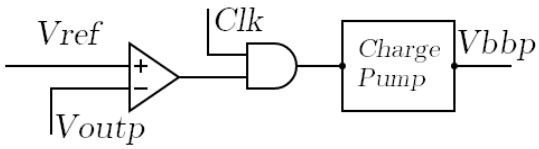

Figure 7.14: Simplified schematic representation of a possible implementation of part of the transfer loop that would enact the control system

feeds into a charge-pump that produces the negative body bias V_{bbp}. This body bias would feed both the threshold-regulated voltage reference and the registers of the cryptosystem, thus closing the feedback loop.

This implementation, however, has several initial drawbacks that can rapidly make it unfeasible. First and foremost, we have seen in 6.4 that the contour lines of $\epsilon(V_{bbn}, V_{bbp})$ change with temperature. Thus, the reference voltage of Figs. 7.13 and 7.14 should really be a function of temperature and not a static value. Implementing this could be extremely difficult, since this system is agnostic to the workings of the registers under study. That is, it derives no information directly from the registers upon which the system ultimately acts upon.

At the same time, temperature also has an effect on the output voltage and transfer characteristic of the threshold-referenced voltage regulator. This follows directly from the fact that the threshold voltage is in and of itself also a function of operating temperature. Thus, even if temperature had no effect on the leakage current of registers, it would have an effect on the value of V_{ref}.

In reality, these two temperature effects would compound, most likely incurring a significant error on the value of V_{ref}, which, without real and dynamic information from the system, would have to be externally set. It is then clear that another option is required.

7.2.2 IDDq

Based on the previous observations made, it would be desirable to directly measure the leakage currents of registers in order to have direct and dynamic information on the registers' side-channel leakage. This way, the system could respond to temperature variations that heavily impact the magnitudes of leakage current. However, we have to contend with the fact that the currents that we wish to measure are very small, in the order of hundreds of picoampers to a few nanoampers.

One simple possibility would be to place a shunt resistor between the voltage source node and the register that it feeds (Fig. 7.15). However, this presents its own problems. As previously stated, the currents measured are of a very small magnitude. Thus, the resistor should present a very high magnitude, the amplification should be intense or, most likely, both. This would significantly pollute the measurements of the leakage currents with noise, potentially diminishing their reliability to represent leakage currents of active registers.

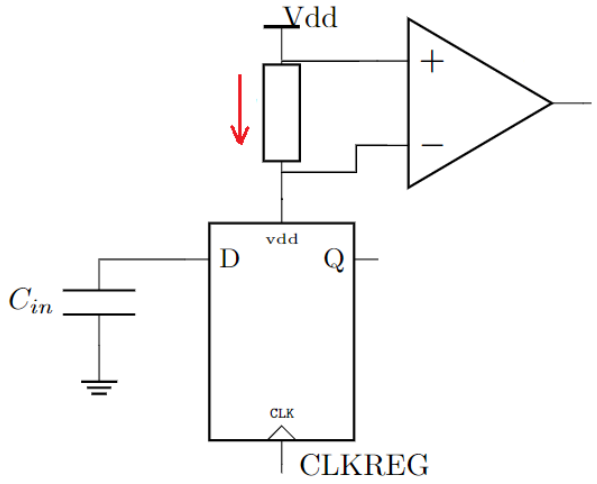

Figure 7.15: Potential leakage current front-end sensor based on the voltage drop experienced by a shunt resistor

At the same time, we have to think of the register of Fig. 7.15 as a unit under test, not part of the cryptosystem. Thus, registers that belong to the cryptosystem do not have a shunt resistor between the current source and their V_{dd} terminal. This way, the register under test sees a voltage drop at its V_{dd} terminal that the cryptosystem's registers do not experience. Because of this, the static currents consumed by the register under test might not be fully representative of the static currents consumed by the registers that belong to the cryptosystem.

Considering that currents are of a very small magnitude, we could sacrifice time and implement an integrating sensor. That is, use a front-end sensor that integrates the current consumption for a given period of time, rather than directly measuring the currents.

Currents are easily integrated through the use of capacitors, which naturally act as transducers between charge and voltage, facilitating the implementation of a sensing unit.

$$\Delta V_c = \frac{1}{C} \int_{t=0}^{t=\tau} I_c(t) \tag{7.20}$$

Thus, by using a switching capacitor and a scheme that would first charge the capacitor to a known voltage value and then sense the drop in voltage due to the discharging of the aforementioned capacitor, one could measure the integrated currents.

If the integrating time (τ) is fixed, given that our ultimate goal is to equalize leakage currents, we would wish to be able to compare these voltage drops within the established period of time:

$$(V_{ref} - V_{c1}) - (V_{ref} - V_{c2}) = V_{c2} - V_{c1}$$

$$V_{c2} - V_{c1} = \frac{1}{C} \int_{t=0}^{t=\tau} I_{c2}(t) - I_{c1}(t) dt \tag{7.21}$$

By modifying the body bias value, we could reach the desired goal of current equalization:

$$\int_{t=0}^{t=\tau} I_{c2}(t, v_{bb}) - I_{c1}(t, v_{bb}) dt \approx 0 \tag{7.22}$$

The problem now would be how to actually implement the comparison. If an analog design is favored, both a sample and hold circuit (to isolate the integrated signal) and an analog comparator would be necessary. The sample and hold would have to be very well designed to avoid charge injection and clock feedthrough. The analog comparator would also require complex design given the use of a 28 nm node and linearity requirements, among others. If an alternative exists, it would be preferable.

Based on these conceptualizations, we settle for a design based on the descriptions made on [81], and depicted in Fig. 7.16, implementing a Time-to-Digital converter using an integrative front-end for the sensing of quiescent currents.

7.3 Time-to-Digital Converter

The name Time-to-Digital converter is self-descriptive. These circuits, that range in complexity [82], convert the time it takes for an event to take place into a digital signal, namely, an integer bit representation.

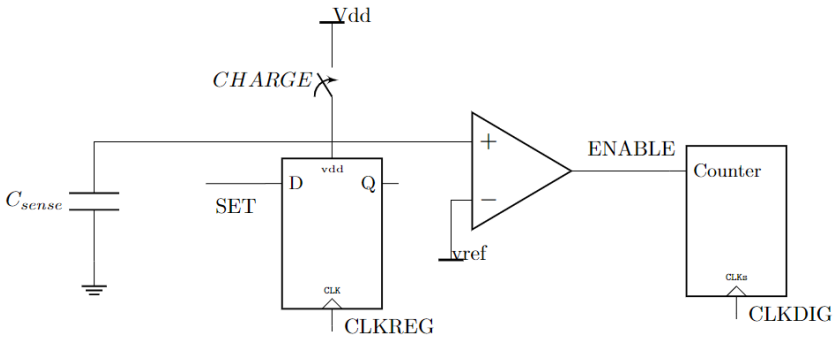

Figure 7.16: Capacitive leakage current sensor schematic representation

The circuit depicted in Fig. 7.16 acts as the analog front-end of a Time-to-Digital converter (TtDC). The sensor presents two phases, a charging phase, and a sensing phase. During the charging phase, the power switch, controlled by the signal $CHARGE$, charges the capacitor C_{sense} to V_{dd}. During the sensing phase, the switch opens, and C_{sense} discharges itself through the leakage current consumed by the flip-flop. The capacitor continues to discharge until the voltage across its terminals reaches V_{ref}, at which point, the $ENABLE$ signal produced by an analog comparator turns low, and the cycle begins anew.

At the same time, the $ENABLE$ signal determines when the counting register should count up (or, as it will be shown, down). Thus, at the end of the sensing cycle, a bit count representing the time it took for the sense capacitor to discharge is stored in the counting register.

With this, we have two scenarios: the register under test is storing a 1, or the register is storing a 0. In each scenario, we assume that the leakage current is, respectively, I_1 and I_0. We expect these currents to vary with V_{dd} and, therefore, we expect these currents not to be constant and not to be entirely linear (they most likely exhibit an exponential behavior as the capacitor is slowly discharging).

Thus, even if the comparison between two measurements (when register is storing a 1, and when it stores a 0) is perfect, we cannot guarantee that the integrative behaviour of the leakage current as it varies due to the discharge of C_{sense} is fully representative of the static conditions under which leakage current analysis attacks take place. Nonetheless, for the sake of simplicity, we are going to assume that both I_1 and I_0 are constant, regardless of the voltage across the terminals of C_{sense}.

With these, given a value of V_{ref}, the time it takes to discharge C_{sense} when the register is storing a 1 is:

$$\int_{v_{ref}}^{v_{dd}} dV_c = \int_{t=0}^{t=T} \frac{I_1}{C_{sense}} dt$$

$$\Delta V_c = I_1 \cdot \frac{T}{C_{sense}} \tag{7.23}$$

Where $\Delta V_c = V_{dd} - V_{ref}$. Assuming that the counter is clocked at a frequency f_s, represented in Fig. 7.16 by the signal $CLKDIG$, and assuming that $T >> \frac{1}{f_s}$ we expect the counter to count up to the value n, where n is defined as:

$$n = T/T_s \tag{7.24}$$

Where $T_s = 1/f_s$ and the division in 7.24 should be seen as an integer division. Thus, isolating T from 7.24, substituting it in 7.23 and isolating n, we have:

$$n_1 = \Delta V_c \cdot C_{sense} \cdot f_s \frac{1}{I_1} \tag{7.25}$$

Defining k as $k = \Delta V_c \cdot C_{sense} \cdot f_s$, we have:

$$n_1 = \frac{k}{I_1} \tag{7.26}$$

And, equivalently, for I_0:

$$n_0 = \frac{k}{I_0} \tag{7.27}$$

If we subtract 7.26 from 7.27 we get:

$$n_0 - n_1 = k(\frac{I_1 - I_0}{I_1 \cdot I_0}) \tag{7.28}$$

And, noting that $\epsilon = I_1 - I_0$, we finally arrive at an expression that is directly proportional to ϵ:

$$n_0 - n_1 = k\frac{\epsilon}{I_1 \cdot I_0} \tag{7.29}$$

With this, we have an integer value that is representative of the magnitude of ϵ and can be used as a direct and dynamic observation of the static current profile of the register under test.

There are two possibilities regarding the implementation of equation 7.29 into a system. The first one sacrifices area and power by simply doubling the circuit depicted in Fig. 7.16. In this case, there are two copies of the same circuit, with two registers under test, one which always stores a 1, and one that always stores a 0. At the same time, there are two counter registers, and the means through which to compare these two counters after the sensing phase.

However, another possibility is the sacrifice of time, rather than area and power. In this case, there is only one copy of the circuit depicted in Fig. 7.16. But now, rather than two phases of operation, there are four: $Charge_0$ - $Sense_0$ - $Charge_1$ - $Sense_1$. During the phases $Charge_0$ and $Sense_0$, the register under test stores a 0, and the counter counts *up*. And during phases $Charge_1$ and $Sense_1$ the register stores a 1, and the counter counts *down*. This way, at the end of phase $Sense_1$, the counter has the quantity $n_0 - n_1$ directly stored in it. This, however, doubles the measuring time. Which option is better depends on the constraints of the overall system. In this work, however, we have opted for the latter option.

7.3.1 Low vs Regular Threshold Transistors

In our present discussion and analysis we have, throughout the bulk of this work, used flip well, Low threshold voltage (LVT), forward body bias transistors, and registers implemented with this technology, as the building blocks for the developments of various countermeasures.

However, when seeking to implement a current balancing countermeasure utilizing a Time-to-Digital converter as a current sensor, we have to take into account some considerations that might make the use of these transistors unfeasible. And even if their implementation was feasible, it would be unsound.

As it was shown in section 6.4, Regular threshold voltage (RVT), non-flip well FDSOI transistors present a wider range of pairs of body bias values V_{bbn} and V_{bbp} where the leakage current disparity of registers is relatively small, as compared with registers implemented with LVT transistors.

At the same time, the behaviour of transistors with increased absolute body bias, be it, $|V_{bbn}|$ or $|V_{bbp}|$, is opposite in LVT and RVT transistors. That is, the threshold voltage of LVT transistors *decreases* with an increased absolute value of body bias (Forward Body Bias -FBB), while the threshold voltage of RVT transistors *increases* (Reverse Body Bias - RBB). Thus, RVT transistors see their static current consumption *decreased* as body bias is incremented (in absolute value). On the other hand, the static current consumption of LVT transistors increases with higher values of absolute body bias.

This latter option presents a problem when dealing with the sensitivity of the sensor, represented by equations 7.26, 7.27 and 7.29.

In essence, if the currents are too big, it is possible that the sensor "overloads" and becomes incapable of producing a count. This can be seen in Equation 7.26, for example, where in the case when:

$$k/I_1 < 1, \tag{7.30}$$

the sensor cannot distinguish between any possible value of I_1 that lies within $0 < I_1 < k$. In this case, the sensitivity can be enhanced by increasing ΔV, C_{sense}, or f_s.

However, the sensitivity of the comparison (Equation 7.29) is dependent both on k, directly proportional, and on the inverse of the product of I_1 and I_0.

We can rewrite Equation 7.29 by noting that:

$$\begin{aligned} I_1 &= I_n + \frac{\epsilon}{2} \\ I_0 &= I_n - \frac{\epsilon}{2} \end{aligned} \tag{7.31}$$

Thus:

$$I_1 \cdot I_0 = I_n^2 - \left(\frac{\epsilon}{2}\right)^2 \tag{7.32}$$

And if we consider that $\epsilon^2 << I_n^2$, then:

$$n_0 - n_1 = \frac{k}{I_n^2} \cdot \epsilon \tag{7.33}$$

It can be seen that the overall sensitivity of the comparison is reduced by a factor $1/I_n^2$. Thus, by using registers implemented using LVT transistors, as the body bias is increased to seek for a value of epsilon that is small, the sensor might "choke" by a reduction of its sensitivity before reaching a current balancing condition.

This is not the only problem. If the attacker is aware of this countermeasure, they might opt to increase the temperature at which the attack takes place. By increasing the temperature, the sensor might, again, "choke" even before it begins to operate on the body bias, being rendered useless.

Thus, the sounder option would be to utilize RVT transistors, with reverse body bias capabilities. These transistors not only present smaller leakage currents at the same body bias points and operating temperature, but they allow the implementation of a negative feedback control, in which, in case temperature increases, the overall static current might be forced to remain within a range of values.

As such, the whole system, including the cryptosystem, is intended to be implemented with RVT, RBB transistors. This, again, comes at the expense of time, as the transitions between digital states of cells built with these transistors requires longer times so as to not violate time constraints.

7.4 Control Unit & Drivers

So far we have decided on the structure and components of the front-end current sensor, and made a case for the use of a system implemented with RVT, RBB transistors.

With this, we have sufficient information to make a case for the overall design of the system, presented in diagram form in Fig. 7.17.

The sensor feeds into the Digital Control Unit (DCU), where two comparisons take place. The results of the comparison are fed into the core of the control unit, implemented as a Moore Finite State Machine.

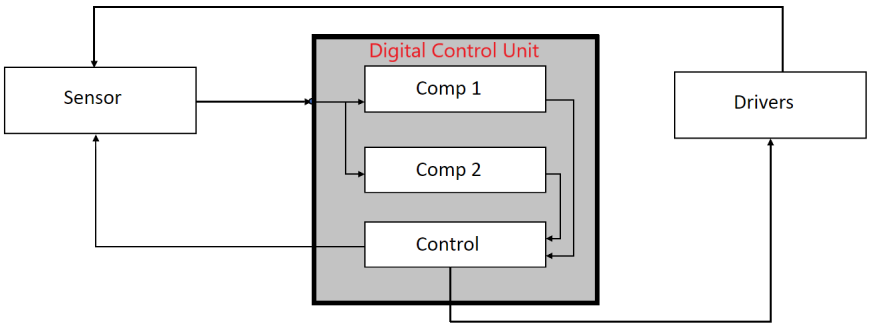

Figure 7.17: Block diagram representation of the current balancing seeking circuit

At the same time, the control unit determines the flow of signals that control the order of operations.

Regarding the driving units, we use charge pumps to charge the body bias wells to their appropriate voltage levels, whichever they may be. At this stage of the design, we limit ourselves to idealized Charge pumps, schematically represented in Figs. 7.18 and 7.19.

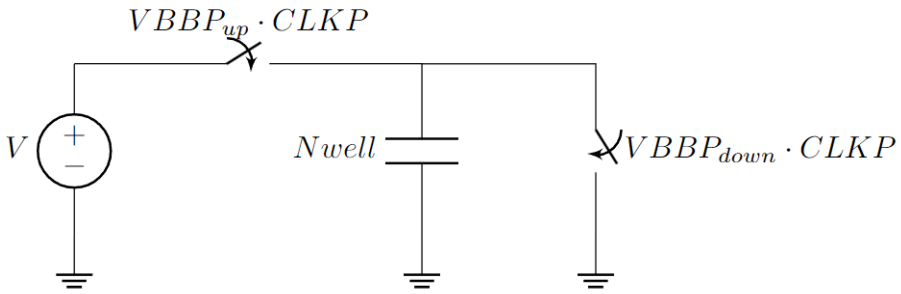

Figure 7.18: Schematic of the ideal positive voltage charge pump representing the drivers

The drivers are implemented in a way such that they are able to both charge and discharge the body bias wells of both the Register Under Test, part of the sensor, and the registers part of the cryptosystem, not represented here.

Thus, the Control unit affects the different drivers through signals $CLKP$ and $VBBN_{UP}$, $VBBN_{DOWN}$, $VBBP_{UP}$ and $VBBP_{DOWN}$. When either of the $VBBx$ signals is driven high and $CLKP$ is activated, the corresponding driver, be it a charge-pump or a ground switch, is activated (Figs. 7.18 and 7.19).

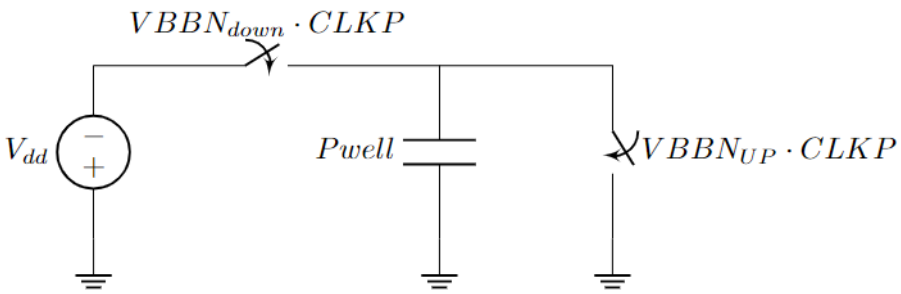

Figure 7.19: Schematic of the ideal negative voltage charge pump representing the drivers

At the same time, the control unit controls the 4 phases of the front-end sensor through signals $CHARGE$, SET, and $CLKREG$ (Fig. 7.16). When C_{sense} is fully charged, $CHARGE$ is driven low, disconnecting the capacitor and the Register Under Tests from the power supply. The capacitor is then discharged by either I_1 or I_0 depending on the data stored in the register. The output of the analog comparator acts as the $ENABLE$ to the counter, driven high until the voltage of the capacitor reaches V_{ref} and the count stops. At this point, $CHARGE$ is driven high so as to restore the node to V_{dd}, SET is toggled, and a pulse is sent through $CLKREG$ to store the new value in the register. Thus, the control unit guarantees that the sensor cycles through the four phases $Charge_0$ - $Sense_0$ - $Charge_1$ - $Sense_1$.

The DCU implements two algorithms that govern the behavior imparted on the drivers. The information required to inform the behaviours of the drivers is derived from the two comparators shown in Fig. 7.17.

Hysteresis Algorithm

Comparator 1 implements an hysteresis comparator upon the value of I_0. This comparator is present following the discussion of section 7.3.1. At the end of phase $Sense_0$, the counter of the front-end sensor stores a value equal to:

$$n_0 = k \cdot \frac{1}{I_0} \tag{7.34}$$

This value is representative of the magnitude of I_0. The digital control unit is equipped with two comparators, each with a reference constant value (n_{0MAX} and n_{0min}) against which n_0 is compared (Fig. 7.20). The block noted as C represents a combinational block implementing the function $C = \bar{C_1} \cdot \bar{C_2} + H \cdot (\bar{C_1} + \bar{C_2})$, which ensures the hysteresis behaviors. If n_0 is smaller than n_{0min}, the sig-

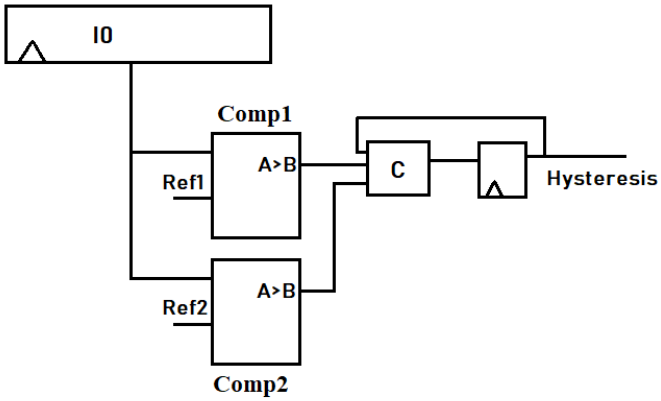

Figure 7.20: Diagram representation of the hysteresis comparator.

nals $VBBN_{UP}$ and $VBBP_{UP}$ are driven high. Thus, by activating the charge pumps, the back gates of reverse body bias transistors are charged, increasing their threshold voltage in the process and in this manner decreasing the leakage current consumption, until $n_0 > n_{0MAX}$.

This algorithm takes precedence over the one implemented through comparator 2. Thus, whenever the condition is met that prompts the activation of the hysteresis portion of the circuit, both signals $VBBN_{UP}$ and $VBBP_{UP}$ are driven high, regardless of the value of ϵ represented by $n_0 - n_1$. This guarantees that the sensitivity of the sensor is somewhat constant, even if temperature is increased.

Current Balancing Seeking Algorithm

The current balancing seeking algorithm is based on the notions of steepest descent algorithms. The function $\epsilon(V_{bbn}, V_{bbp})$ is a scalar field, whose gradient $\nabla\epsilon(V_{bbn}, V_{bbp})$ is a vector field representing the direction of steepest descent. This can be seen in Figs. 7.21, 7.22, and 7.23.

Figures 7.21, and 7.22 depict the contour lines of the function $|\epsilon(V_{bbn}, V_{bbp})|$ for registers implemented with RVT, RBB transistors at 80 ºC. Figure 7.22 includes the vectors representing the direction of steepest descent. Figure 7.23 showcases function $|\epsilon(V_{bbn}, V_{bbp})|$ as a 3D representation for clarity. It can be seen that there exist minimums, beyond which epsilon is always bigger.

We use here the notion of steepest descent not because we are choosing the path of faster descent of $|\epsilon|$, but because we are interested in the overshoot condition. In steepest descent algorithms, it is a known fact that, having a wide step can incur in convergence problems giving an overshoot over the minimum point.

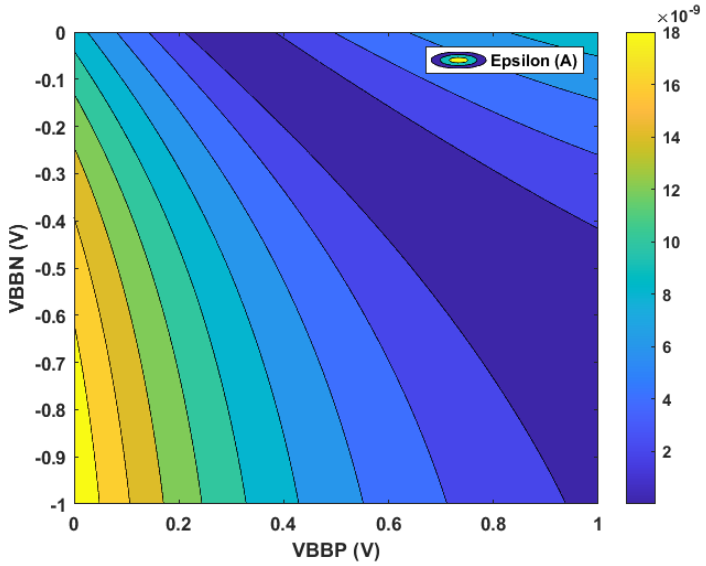

Figure 7.21: Contourn map of $|\epsilon(V_{bbn}, V_{bbp})|$ for D flip-flops under study implemented with RVT, RBB transistors at 80 ^{o}C

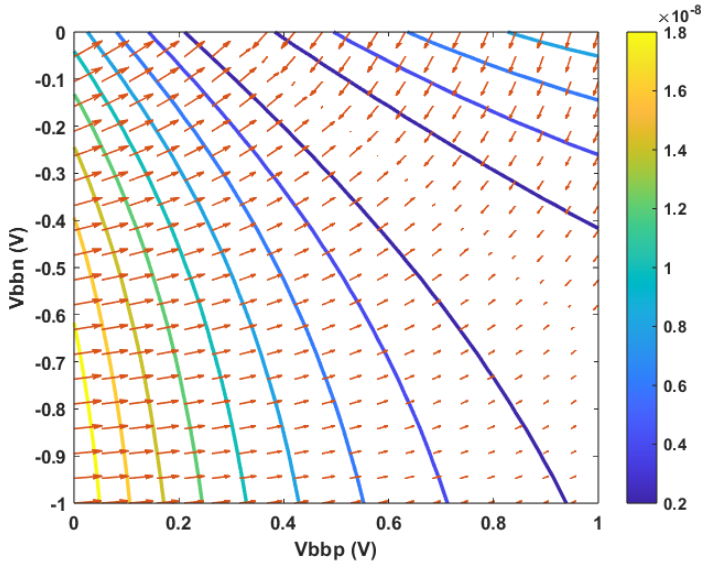

Figure 7.22: Contourn map of $|\epsilon(V_{bbn}, V_{bbp})|$ for D flip-flops under study implemented with RVT, RBB transistors at 80 ^{o}C superimposed with its gradient function

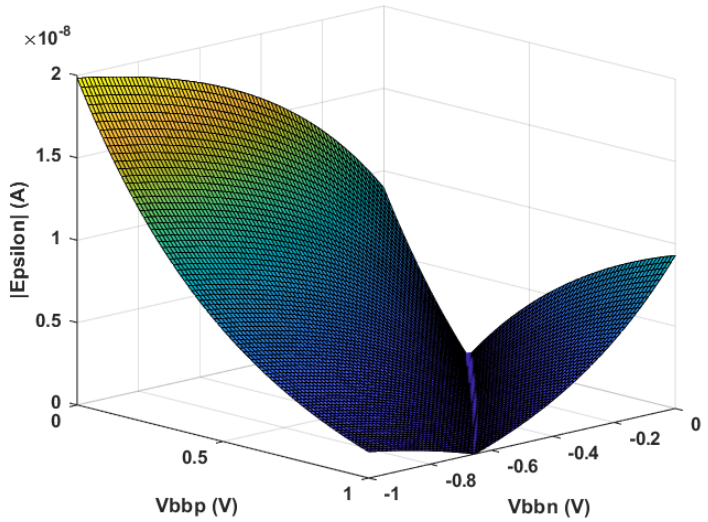

Figure 7.23: 3D representation of the function $|\epsilon(V_{bbn}, V_{bbp})|$.

$$d\epsilon(V_{bbn}, V_{bbp}) \approx \frac{\partial \epsilon}{\partial V_{bbn}} \Delta V_{bbn} + \frac{\partial \epsilon}{\partial V_{bbp}} \Delta V_{bbp} \tag{7.35}$$

In our implementation, it can be problematic to compare the value of ϵ against a constant, static reference value, given that the conditions can vary with temperature. Thus, instead of comparing the value of epsilon against a given reference, we seek the overshoot condition.

We define this condition by noting that, as ϵ varies with the body bias, the value of the sensor count can change signs:

$$n_0 - n_1 = k \cdot \frac{\epsilon}{I_0 \cdot I_1} \tag{7.36}$$

In order to sense this change, a secondary register other than the count register is needed.

The primary register is depicted in Figs. 7.24 and 7.25 as $\epsilon[n]$. This register is the count register of the front-end sensor (Fig. 7.16). At the end of the 4 phase sensing cycle, the count register stores the value $n_0 - n_1$, represented here as $\epsilon[n]$. Here, n refers to the n period of the sensing activity. Before a new cycle of sensing begins, the content of the count register $\epsilon[n]$ is loaded onto the secondary register marked as $\epsilon[n-1]$, after which register $\epsilon[n]$ is reset to 0. Thus, at period n, after the sensing cycle is complete, the control unit compares the contents of registers $\epsilon[n]$ and $\epsilon[n-1]$.

These registers are implemented as signed integers. Thus, the Most Significant Bit acts as the sign bit of the registers. To determine whether a sign

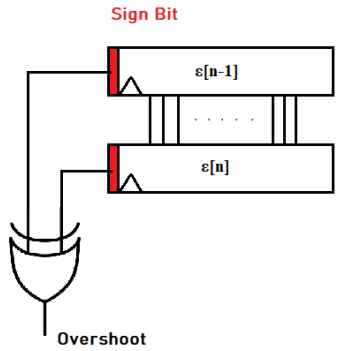

Figure 7.24: Schematic representation of the Overshoot detector implementation through an XOR gate.

XOR sign change		
inputs		Out
a	b	Z
0	0	0
0	1	1
1	0	1
1	1	0

change has taken place between period $n-1$ and period n, an XOR gate between the MSBs of both registers is used. It can be seen that, when the comparison between the MSBs indicates a change in sign between two contiguous periods, the XOR gate produces a signal (transitioning from $0 \rightarrow 1$ or viceversa). This signal is fed into the control unit. When the overshoot condition is reached, the drivers enter a stand-by phase.

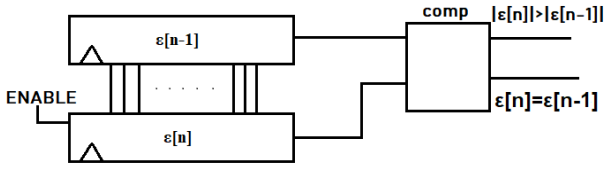

Figure 7.25: Schematic representation of the comparators informing the progression of the drivers used.

The system operates in a sensing-driving cycle. That is, after each 4-phase sensing cycle, the system enters a driving phase, informed by the result of the comparisons that take place after the sensing. The driving is controlled by signals signals $VBBN_{UP}$, $VBBN_{DOWN}$, $VBBP_{UP}$ and $VBBP_{DOWN}$ and the

Driver Decoder					
inputs		outputs			
CD_1	CD_0	$VBBN_{UP}$	$VBBN_{DOWN}$	$VBBP_{UP}$	$VBBP_{DOWN}$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table 7.1: Drivers' decoder truth table. The signals CD_1 and CD_0 represent the drivers' counter.

signal $CLKP$.

The system begins by activating $VBBN_{UP}$ and keeping it activated until the comparison $|\epsilon[n]| > |\epsilon[n-1]|$ is met (Fig. 7.25), or the overshoot condition is reached. When the condition is met $|\epsilon[n]| > |\epsilon[n-1]|$, it means that further keeping the current driver signal active would only increase the value of ϵ. In this case, the system cycles to the next driver signal. The system cycles through these signals in the order above stated: $VBBN_{UP} \rightarrow VBBN_{DOWN} \rightarrow VBBP_{UP} \rightarrow VBBP_{DOWN}$.

Another comparison $\epsilon[n] = \epsilon[n-1]$ informs a time out. When the comparison $\epsilon[n] = \epsilon[n-1]$ is met for more than four sensing-driving cycles, a time out is signaled, automatically transition from the current driver signal to the next.

Thus, the signals are $VBBN_{UP}$, $VBBN_{DOWN}$, $VBBP_{UP}$ and $VBBP_{DOWN}$ are controlled by a $2^1 \rightarrow 4$ decoder, whose input is a 2-bit counter, and whose outputs are the signals $VBBN_{UP}$, $VBBN_{UP}$, $VBBP_{UP}$ and $VBBP_{DOWN}$.

This way, the system is capable of traversing the entirety of the $V_{bbn} \times v_{bbp}$ plane, while seeking the appropriate body bias point. Recall that if the Hysteresis condition is met, both $VBBN_{UP}$ and $VBBP_{UP}$ become active, regardless of the value of the counter. This is done with two OR gates, whose inputs are the $VBBx_{UP}$ and $Hysteresis$ signal.

FSM Control implementation

Figures 7.26 and 7.27 show the finite state machines controlling both the front-end sensor and the driver.

During state *setup* and *set settle* the capacitor C_{sense} is charged to V_{dd} and the line SET (see Fig. 7.16) is driven the appropriate value to be loaded into the register, some time is waited to ensure hold and set-up times, and the clock pulse is sent to the register through signal $CLKREG$. While in state *setsettle*, the sensor is forced to wait until it receives signal $WAIT_D$, issued from the FSM controlling the drivers during its own *wait* state. The sensor then enters the state *count*, where the signal $CHARGE$ is driven low, disconnecting the register under test and sensing capacitor from the power supply. In this state, the counter counts up when the signal $Toggle$ (T in Fig. 7.26) is 0, and down when $T = 1$. When the enable signal from the front-end sensor is driven low

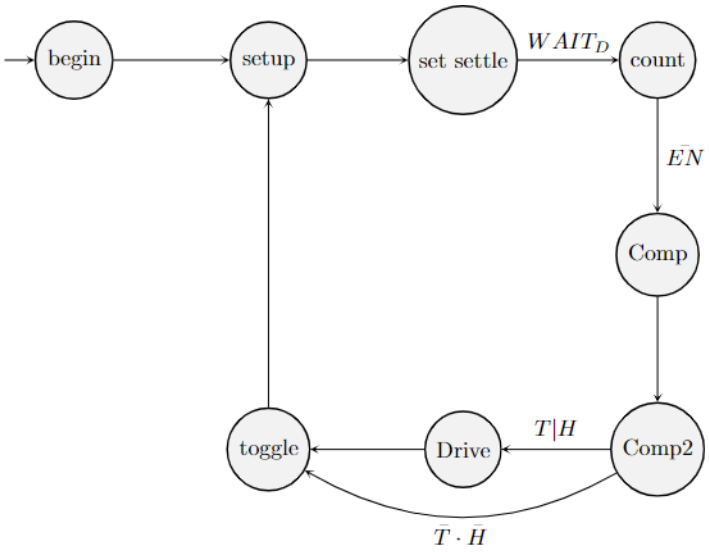

Figure 7.26: State transition diagram of the sensor portion of the control unit

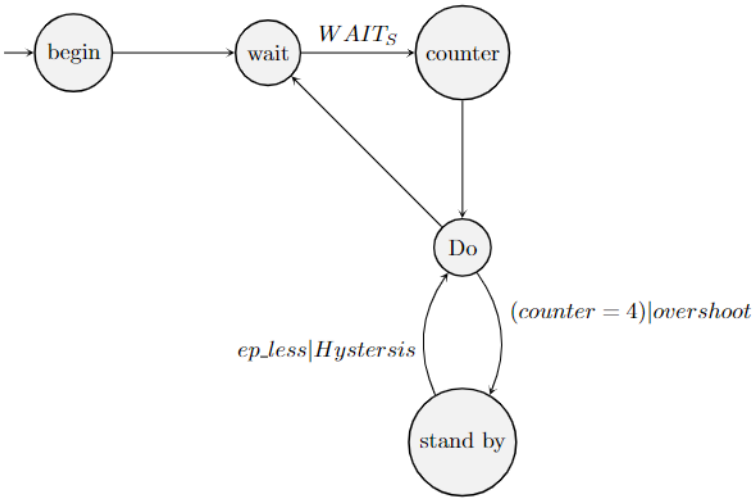

Figure 7.27: State transition diagram of the drivers portion of the control unit

(C_{sense} has reached the value V_{ref}), the sensor enters the comparison phase, where the comparators shown in these sections store their resulting values in flip-flops, to be used as signals for the driver's FSM. At this point, if T or H are equal to 1, the FSM enters the *Drive* state, where it issues a the signal $WAIT_S$ for the drivers' FSM, allowing the drivers to become active.

The drivers' FSM, once active, enters the state *counter*, when, depending on the information received from the comparators, increases the driver counter (Table 7.1). It then progresses into state *Do*, where a single pulse $CLKP$ (Figs. 7.19 and 7.18) is issued, charging or discharging the body bias wells according to the state of the signals $VBBN_{UP}$, $VBBN_{DOWN}$, $VBBP_{UP}$ and $VBBP_{DOWN}$. Then, the drivers' FSM can either enter the *standby* state, or the *wait* state, the cycle beginning anew.

7.5 Results

7.5.1 Circuit Simulation

We implement the system in the Cadence Virtuoso ADE environment with ideal components. The front-end sensor, excluding the counter, which is included in the Digital Control Unit, is implemented with ideal capacitor, switches and an ideal analog comparator (Fig. 7.16). The Register under test is an instance of a standard cell D flip-flop, implemented with RVT, RBB transistors.

The Digital Control Unit is implemented in VHDL at the RTL level and imported into the schematic as a behavioural block.

The drivers (Figs. 7.19 and 7.18) are also implemented with ideal components.

We then perform analog-mixed simulation of the circuit at differing temperatures under the conditions seen in Table 7.2. The complete derivation and reasoning behind these values can be found in the Annex (10.3). The results of the simulation at 80 ºC can be seen in Fig. 7.28.

Table 7.2: Simulation conditions

T	80 ºC
V_{dd}	1 V
V_{ref}	0.8 V
f_{clk}	250 MHz
C_{sense}	750 fF
R_s	25 kΩ
I_{0MAX}	15 nA
I_{0min}	10 nA
$bits$	17
n_{Hmin}	2500
n_{Hmax}	3500

It can be seen in Fig. 7.28 how, initially, both V_{bbn} (negative curve, orange) and V_{bbp} (positive curve, red) increment symmetrically due to the hysteresis conditions. Once the hysteresis condition is met, the value of V_{bbn} is decreased,

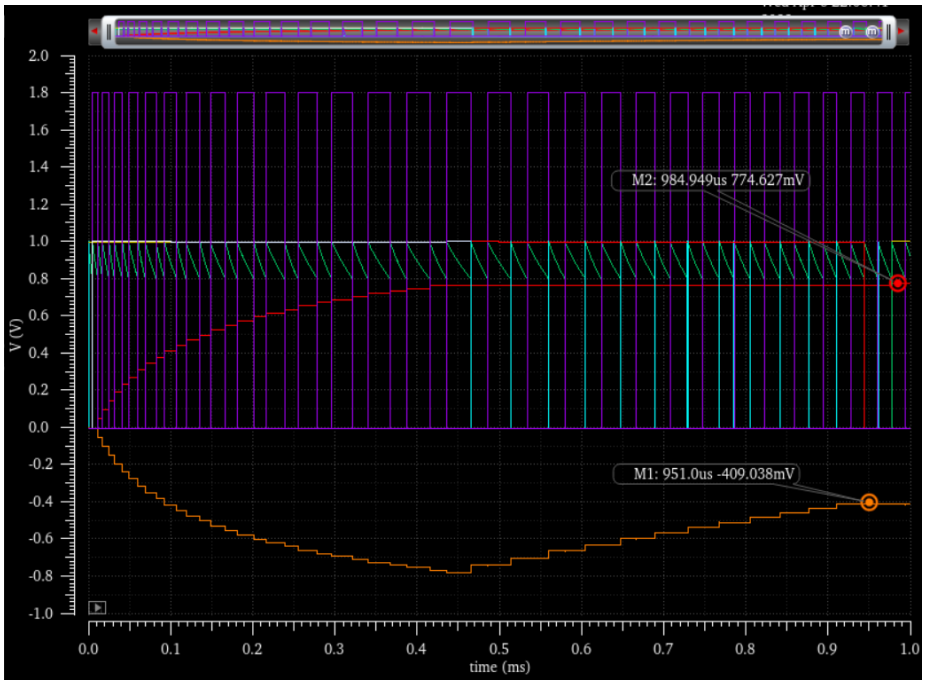

Figure 7.28: Analog-Mixed simulation results of the body bias seeking circuit at 80 ºC

until the condition of change of sign of ϵ is reached. At this point, the circuit enters the driver stand-by state.

Table 7.3 showcases the values of ϵ and I_0 displayed by the registers under study under the body bias conditions reached by the body bias seeking circuit at 80 ºC. It can be seen how I_0 is kept between 10 and 15 nA (the hysteresis conditions). It can also be seen how, with the aid of the circuit, the value of epsilon is reduced by a magnitude of approximately 32 as compared to the absence of a countermeasure.

Table 7.3: Results from an initial simulation at 80 ºC

	CBBB		No CBBB	
	V_{bbn}	V_{bbp}	V_{bbn}	V_{bbp}
	-409 mV	775 mV	0 V	0 V
ϵ	-274 pA		9 nA	
I_0	12 nA		41 nA	

Table 7.4 shows the results from the simulation at different temperatures. It also includes the factor $|K|$, which represents the ratio of the reduction of ϵ

Table 7.4: Results at varying temperatures

	80 ºC		50 ºC		35 ºC	
	V_{bbn}	V_{bbp}	V_{bbn}	V_{bbp}	V_{bbn}	V_{bbp}
	-409 mV	775 mV	0 V	311 mV	0 V	311 mV
ϵ	-274 pA		-62 pA		38 pA	
I_0	12 nA		8 nA		4 nA	
$\|K\|$	32		51		47	

when the countermeasure is applied and when its is not applied. That is:

$$K = \left| \frac{\epsilon_{noCBB}}{\epsilon_{CBB}} \right| \tag{7.37}$$

As it can be seen, the countermeasure is exceedingly successful in finding a body bias point in which the leakage currents I_0 and I_1 are equalized, with a wide range of temperatures.

A synthesis of this circuit reveals an approximate area of 700 μm^2.

7.5.2 PCC

In Chapter 6 we have calculated the PCC for bivariate leakage power attacks of the AES and the Trivium under the condition that $\epsilon \approx 1$ nA, at 27 ºC. With the results offered by the leakage current equalization circuit we can recompute and plot the PCC under a variety of conditions in order to gauge the effect of the countermeasure.

In order to do so, we solve Equation 6.36, repeated here for clarity, with some modifications.

$$\rho_{I_{leak},Z_n} = \frac{\mu_\epsilon \sigma_{Z_n}}{\sqrt{(\sigma^2_{Z_n} + \sigma^2_{Z_m})(\mu^2_\epsilon + \sigma^2_\epsilon) + 2\sigma^2_b}} \tag{7.38}$$

Equation 6.36 represents the PCC of a bivariate attack on the AES cryptosystem, under a correct key assumption, including both algorithmic and non-algorithmic noise.

Under our countermeasure, we are going to assume that ϵ is no longer a random variable, but a constant. Thus, $\sigma^2_\epsilon = 0$, and the equation can be rewritten as:

$$\rho_{I_{leak},Z_n} = \frac{\mu_\epsilon \sigma_{Z_n}}{\sqrt{\mu^2_\epsilon(\sigma^2_{Z_n} + \sigma^2_{Z_m}) + 2\sigma^2_b}} \tag{7.39}$$

We first plot Equation 7.39 for a fixed noise power of $\sigma_b^2 = -134dBW$ as a function of averaged sampled at 80 ºC with (blue) and without (orange) countermeasure.

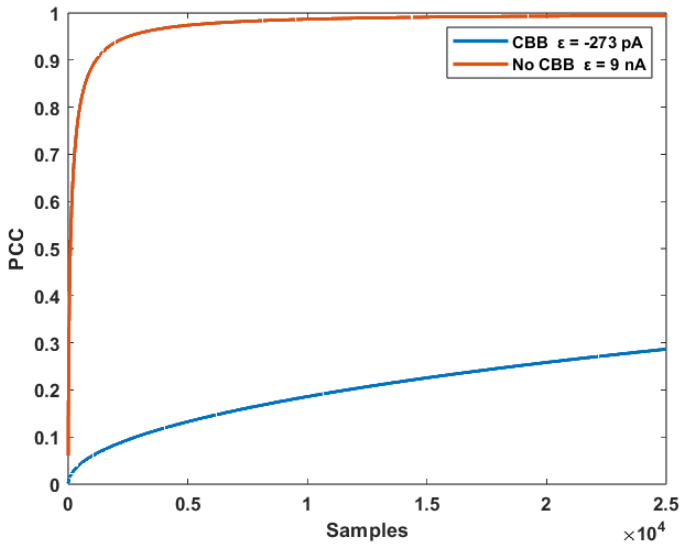

Figure 7.29: PCC between the Hamming Weight and the bivariate leakage power model (Equation 7.39) as a function of averaged number of traces in the presence of algorithmic and non-algorithmic GW noise with Current balanced body bias (Blue) and without a current balanced body bias (Orange) at 80 ºC

It can be seen that the current balanced circuit is much more resilient to attacks performed under the same temperature conditions.

Figure 7.30 showcases the Equation 7.39 a function of noise power for different number of averaged samples. It can be seen how the reduction of ϵ implies that, for the same noise power, the circuit is much more resilient to attacks (see Fig. 6.13).

We also perform a simulated attack on the AES in the manner described in Chapter 6 under these conditions (80 ºC) and note the success rate of key identification for 100 runs under the denoted number of averaged traces. The results can be seen in Table 7.5

It can be seen that a large number of total experiments are required to disclose the secret key, as compared to Table 6.5, even with a success rate as low as 20%.

With this, it can be seen that the countermeasure is highly effective in hindering the secret key acquisition, even at high temperatures.

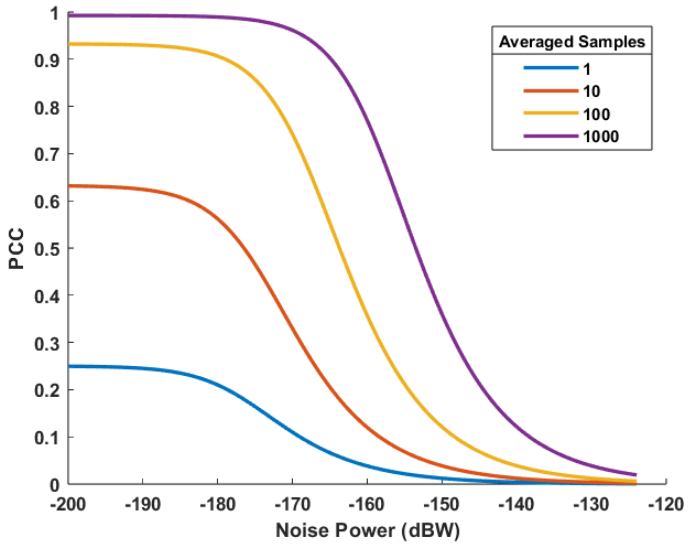

Figure 7.30: PCC between Z_n and the bivariate leakage power model with algorithmic and non-algorithmic noise (Equation 7.39) as a function of noise power under a current balancing body bias at 80 ^{o}C ($\epsilon = -273$ pA), for different number of N averaged traces

Table 7.5: PCC - Theoretical and Numerical Simulations of a 128 bit system in the presence of non-algorithmic noise

CB - GWN -134 dBW	
N	Success Rate
1	0
10	0
100	0
1000	0.04
10000	0.20

7.6 Conclusions

In this chapter, an exploration of different possible circuits with which to implement a control circuit to achieve a body bias point that reduces the magnitude of ϵ as much as possible is carried out.

Several circuits are contemplated and discarded, finally setting in the use of a switched-capacitor, current sensing system in the form of a Time to Digital Converter.

The functionality and parameters of interest are studied, and arguments are made in favor of the use of registers implemented with RVT, RBB transistors.

The circuit implements two control algorithms, one that fixes the value of I_0 in an hysteresis window regardless of the operating temperature, so as to avoid loss of sensor sensitivity; and another that seeks the body bias point that minimizes the value of ϵ.

While some of the chosen parameter values might seem a tad arbitrary, a conservative approach regarding timing constrains is favored. However, were the circuit capable of operating at higher frequencies, some of the area constraints imposed by the size of the capacitive elements might be relaxed, at the expense of increased dynamic power consumption.

The overall design of the circuit is proven to be effective in choosing values of body bias which greatly reduce the magnitude of ϵ, even at higher temperatures. In conjunction with the analyses presented in Chapter 6, this would indicate a highly effective countermeasure, as determined by the results obtained.

The main limitation of the analysis performed in this chapter is the fact that the circuit is implemented and simulated with ideal circuits under ideal conditions. Thus, some physical effects not here contemplated could unknowingly impact the effectiveness of the countermeasure.

The second limitation to take into consideration is the fact that the body bias value of the cryptosystem is driven according to the leakage current profile of a single register under test. Given the effects of process and mismatch variability that arise during the manufacturing process, having the register under test under some given leakage current conditions does not guarantee that the rest of registers behave similarly. These limitations are explored in the next chapter.

Chapter 8

Variability Assessment

In the previous chapter we have presented the design, implementation, and some results of a circuit that seeks a body bias point in which an equalization of the leakage currents of registers storing a 1 and a 0 occurs. The circuit has demonstrated efficiency in hindering the acquisition of the secret key of cryptographic systems under a bivariate leakage power attack, a type of attack that negates use of a random body bias as a countermeasure. This efficiency holds for differing temperatures, to different degrees.

It would then appear that the work is done, and no more considerations are to be taken. However, an important item that is to be taken into account is the fact that the circuit, as it stands, measures exclusively the leakage currents of a single register, which we have called the Register Under Test throughout the text. This poses a problem.

The current balancing circuit is not only modifying the body bias of the register under test, but it is also actuating on the body bias of registers that we are not directly measuring: the registers that compose the cryptosystem.

In a world devoid of stochastic processes, this would not pose a problem. However, during the manufacturing process of a die, the different components that comprise a chip are subjected to fluctuations in their masking and in the diffusion of dopants and other chemicals. These fluctuations give rise to functional variability within the same chip, and between distinct chips present in the same waffer.

It then occurs that, within the same chip, the leakage current profile of different registers might not be equal for the same given body bias point. It might then happen that this variability could be high enough to render the current balancing circuit obsolete.

In this chapter we explore the variability of the registers under study and attempt to determine the extent to which these random effects that arise during the manufacturing process have on the efficacy of the current balancing circuit.

8.1 Initial Approach - Montecarlo Simulations

The initial problem lays in the fact that, in order to find a current balancing body bias point, we modify the body bias values of the register under test, and fix that same condition for all the other registers. These registers, however, might have different values of ϵ at the body bias values found by the current balancing circuit, given process and mismatch variations.

In order to quantify the effects of process and mismatch variability, we employ the same circuit used in chapter 5, designed to simulate the magnitude of ϵ for a given value of body bias at a given temperature. In this particular case, we do not restrict ourselves to a single simulation, but perform a Montecarlo simulation with process and mismatch variability. We set a total of 200 simulations, each set of 200 simulations at different temperatures, using the body bias values found by the current balancing circuit (see Table 7.4). The Montecarlo simulation randomizes exclusively instances of the registers. The rest of the components in the testbench are kept constant to eliminate confounding variables.

With this, we obtain the approximate distribution of $\epsilon(V_{bbn}, V_{bbp})$ under current balancing conditions. One such distribution can be seen in Fig. 8.1 , for the body bias conditions present in Table 7.4 at 80 C.

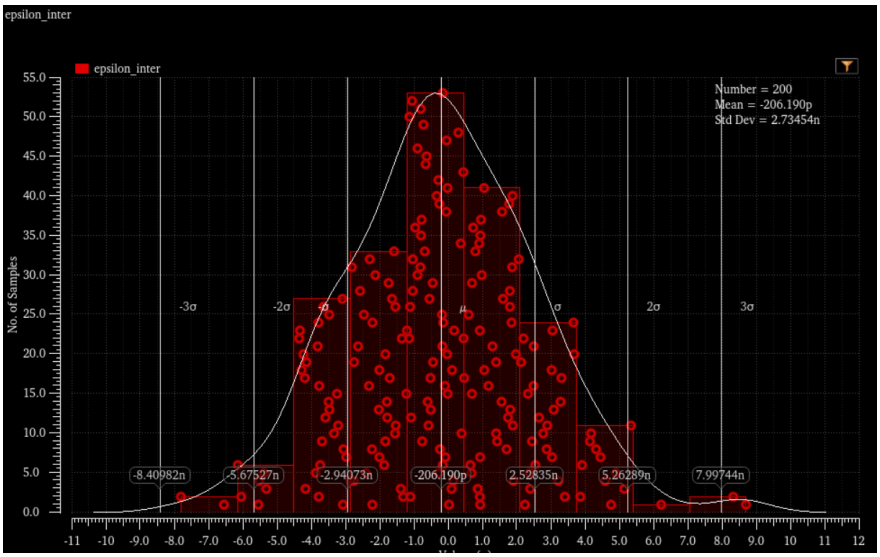

Figure 8.1: Histogram depicting the distribution of ϵ for body bias values $V_{bbn} = -409$ mV and $V_{bbp} = 775$ mV for 200 samples

With this, we can obtain the mean and standard deviation of ϵ for a given body bias point at a given temperature. For ease of analysis we are going to assume that the resulting distributions are Normal, which might not necessarily be the case, but the approximation might suffice for the subsequent derivations.

We perform Montecarlo simulations for three different temperatures, 80 C, 50 C, and 35 C. At the same time, we test four possible body bias schemes:

- Dual Drivers Current Balancing (DDCB): This is the body bias scheme that results from the current balancing circuit described and implemented in the previous chapter. With DDCB, both N- and P-Wells begin at the initial condition $V_{wells}(t = 0^+) = 0V$, and the drivers charge the wells until the current balancing condition is met.

- Single Driver Current Balancing (SDCB): This body bias scheme also utilizes the current balancing circuit. However, in this case the P-well (V_{bbp}) is tied to a constant voltage of 1 V, and the current balancing circuit can only modify the voltage of transistor' N-Wells (V_{bbn}). This body bias scheme is explored for two reasons. The first one is that, as seen in Figs. 6.19 and 6.20, in registers implemented with RBB transistors, the value of V_{bbp} which gives widest interval of V_{bbn} in which the magnitude of ϵ is relatively small is $V_{bbp} = 1$ V, especially at larger temperatures. The second reason is that, at higher values of $|V_{bbn}|$, the standard deviation of ϵ is lower.

- |1| V: Both N- and P-Wells are tied to constant values of, respectively, $V_{bbn} = -1$ V and $V_{bbp} = 1$ V. We test this to determine whether simply having the system operate at the maximum body bias possible is better as a countermeasure than implementing an entire circuit that consumes area, power, and time.

- 0 V: In this case, the body bias for both types of wells is set to 0 V, to have a control group.

For the case of the DDCB scheme, the body bias values are those depicted in Table 7.4.

For the SDCB scheme, the body bias values produced by the current balancing circuit can be seen in Table 8.1.

Table 8.1: Single Driver Current Balancing Results at varying temperatures

80 ºC		50 ºC		35 ºC	
V_{bbn}	V_{bbp}	V_{bbn}	V_{bbp}	V_{bbn}	V_{bbp}
-600 mV	1 V	-550mV	1 V	-425 mV	1 V

The resulting moments (expected value and variance, expressed as standard deviation) of the different distributions obtained under the distinc conditions tested can be seen in Table 8.2.

Some trends can be observed. For a given scheme, the standard deviation of ϵ, σ_ϵ increases with temperature for all cases while, depending on the scheme,

Table 8.2: Mean and standard deviation of registers' leakage current

	T= 80 ^{o}C - (nA)		T= 50 ^{o}C - (nA)		T= 35 ^{o}C - (nA)	
BB Scheme	μ_ϵ	σ_ϵ	μ_ϵ	σ_ϵ	μ_ϵ	σ_ϵ
DDCB	-0.21	2.73	0.07	2.04	0.23	1.09
SDCB	-0.18	1.84	0.08	0.56	0.02	0.34
|1| V	2.13	1.57	.87	0.46	0.60	0.24
0 V	9.54	9.45	-3.00	1.35	-1.38	0.71

the expected value μ_ϵ shifts, either positively for the 0 V and |1| V cases, or negatively for the DDCB and SDCB schemes.

At the same time, it can be seen how the minimum variance of ϵ is achieved with the maximum possible body bias value |1|. Thus, we implement the SDCB scheme, in which one well is always set at its maximum body bias to see if it diminishes the variance of the DDCB scheme. Not only that, but the SDCB scheme presents the smallest mean value of ϵ of all other schemes at a given temperature.

With this, we now need to determine how this variability affects the protection of the cryptosystem supposedly introduced by the current balancing circuit by decreasing the magnitude of ϵ. However, ϵ is now a random variable not because of the body bias, which is fixed for a given temperature by the current balancing circuit, but because every instance of a register produces a random leakage current.

8.2 Variability Analysis - Ideal Distribution

We begin our analysis by going back to the very first equation of leakage current power model used to describe and attack on register slices of cryptosystems. That is:

$$I_{leak} = \epsilon \cdot HW + n \cdot I_0 \tag{8.1}$$

In this equation, the assumption is that every register presents the same magnitude of ϵ. However, with variability in mind, every instance of a register can present distinct values of ϵ. Thus, we need to derive a different equation to take this factor into account.

Suppose we have an instance of n registers, and that each register is statistically independent from each other. Assume, for the moment, that $n = 2$ for ease of derivation. We have seen that ϵ follows an approximate normal distribution. Thus, for an register array of two flip-flops, we have that the leakage current consumption is a continuous random variable of the form:

$$I_{leak}(x) = \epsilon_1(x) \cdot z_1 + \epsilon_2(x) \cdot z_2 + n \cdot I_0 \tag{8.2}$$

Where the different z_i are random variables representing the bits stored in flip-flop i. In principle, I_0, here considered a constant, would also be a random variable. However, for ease of analysis we are going to assume that I_0 is a constant term formed by the summation of the different I_{0_i} magnitudes of each register, and assume that their variability is absorbed by the different ϵ_i terms.

The properties of the random variables z_i have been described in Chapter 5, but are repeated here for clarity. Each z_i represents a bit value. Thus, each z_i follows a discrete random uniform distribution, with $P[z_i = 1] = P[z_i = 0] = 1/2$. Thus, $\mu_z = 1/2$ and $\sigma_z^2 = 1/4$.

Thus, in our particular case, the Hamming Weight of the register array is $HW = z_1 + z_2$. This way, We begin our analysis by calculating the Pearson Correlation Coefficient between the function $I_leak(x)$ and the Hamming Weight stored in the array.

As described in previous chapters, the PCC is computed according to the following equation:

$$\rho_{I_{leak(x)},HW} = \frac{Cov(I_{leak(x)}, HW)}{\sqrt{Var(I_{leak(x)})} \cdot \sqrt{Var(HW)}} \tag{8.3}$$

To calculate the PCC we again note that:

$$Cov(I_{leak(x)}, HW) = E[I_{leak}(x) \cdot (z_1 + z_2)] - E[I_{leak(x)}]E[z_1 + z_2] \tag{8.4}$$

Using the fact that the random variables $\epsilon_i(x)$ are independent from the random variables z_i, we can calculate this equation as:

$$Cov(I_{leak(x)}, HW) = E[\epsilon_1(x)] \cdot \sigma_{z_1}^2 + E[\epsilon_2(x)] \cdot \sigma_{z_2}^2 \tag{8.5}$$

Where $\sigma_{z_i}^2 = E[z_i^2] - E[z_i]^2$. We can also use the fact that the random variables $\epsilon_i(x)$ follow an approximately normal distribution with parameters μ_ϵ and σ_ϵ^2.

Thus, similarly to what we had initially, we have:

$$Cov(I_{leak(x)}, HW) = \mu_\epsilon \cdot \sigma_{z_1}^2 + \mu_\epsilon \cdot \sigma_{z_2}^2 = \mu_\epsilon \cdot \sigma_{HW}^2 \tag{8.6}$$

The variance of $I_{leak(x)}$ can be calculated using the fact that, for independent random variables $Var(X \cdot Y) = \sigma_x^2 \cdot \sigma_y^2 + \mu_x^2 \cdot \sigma_y^2 + \mu_y^2 \cdot \sigma_x^2$. Thus, the PCC can be finally calculated as:

$$\rho_{I_{leak(x)},HW} = \frac{\mu_\epsilon \cdot \sigma_{HW}}{\sqrt{\sigma_{HW}^2\sigma_\epsilon^2 + \mu_\epsilon^2\sigma_{HW}^2 + \mu_{HW}^2\sigma_\epsilon^2}} \tag{8.7}$$

Ignoring, for the moment, algorithmic noise, we can assume that the samples are populated by some form of non-algorithmic noise, and introduce a term σ_b^2, which we are assuming is the noise produced by a 1 Ω shunt resistor operating at 1 V for measurements up to 10 MHz, thus introducing a noise power of -134 dBW. Also, if we consider that there is inter-trace averaging, with N being the number of samples, we have:

$$\rho_{I_{leak(x)},HW} = \frac{\mu_\epsilon \cdot \sigma_{HW}}{\sqrt{\sigma_{HW}^2\sigma_\epsilon^2 + \mu_\epsilon^2\sigma_{HW}^2 + \mu_{HW}^2\sigma_\epsilon^2 + \frac{\sigma_b^2}{N}}} \tag{8.8}$$

8.2.1 Results

With this, we can plot some results and observe some trends. We plot Equation 8.8 for the distributions that arise at 80 C for the different body bias schemes as a function of averaged samples.

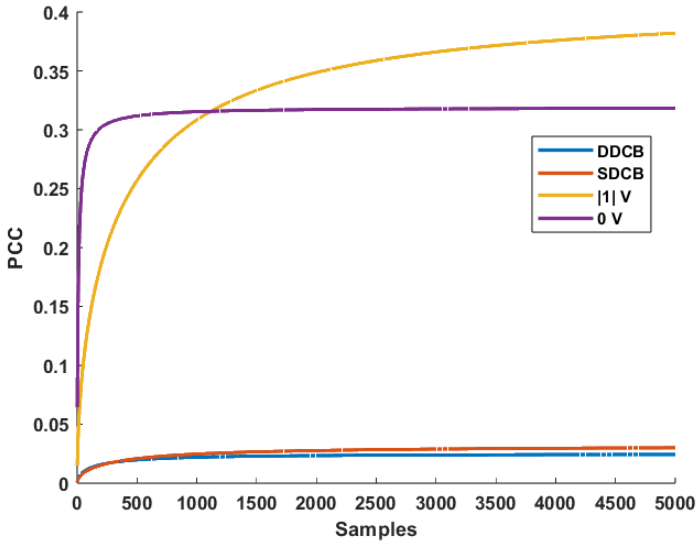

Figure 8.2: PCC as a function of samples of Equation 8.10 at 80 C for the different distributions obtained through a Montecarlo analysis of registers under body bias conditions seen in Table 8.2

It can be seen that both the DDCB and the SDCB schemes are extremely resilient to noise averaging given the distributions of the average registers, and present a very small value of the PCC which might highly hinder the secret key acquisition in CPA attacks.

While these results seem promising, we must introduce a new concept here before reaching some conclusions. While these notions that are going to be introduced might seem unjustified and somewhat arbitrary, they will be clarified in subsequent sections. For the moment, we must slightly modify the assumption of the distribution of $\epsilon(x)$.

Folded Normal Distribution

So far, we have seen that ϵ follows an approximately normal distribution. However, under certain circumstances, we cannot solve or plot equation 8.8 with this assumption in mind, otherwise the results would not accurately represent the PCC obtained through a correlation power analysis attack of a register array with variability.

Instead, we must treat the probability density function of $\epsilon(x)$ as a folded normal distribution; that is, a distribution that arises by considering the absolute value $|\epsilon(x)|$. The reasoning will be explained shortly hereafter.

Under these new conditions, both the expected value and variance of the distribution of $|\epsilon(x)|$ are:

$$\mu_{|\epsilon|} = \sqrt{\frac{2\sigma_\epsilon^2}{\pi}} \cdot e^{-\frac{\mu_\epsilon^2}{2\sigma_\epsilon^2}} + \mu_\epsilon \cdot erf\left(\frac{\mu_\epsilon}{\sqrt{2\sigma_\epsilon^2}}\right)$$

$$\sigma_{|\epsilon|}^2 = \mu_\epsilon^2 + \sigma_\epsilon^2 - \mu_{|\epsilon|}^2 \tag{8.9}$$

Where $erf(\cdot)$ represents the error function. With these, we can substitute μ_ϵ and σ_ϵ^2 by their new forms in Equation 8.8.

$$\rho_{I_{leak(x)},HW} = \frac{\mu_{|\epsilon|} \cdot \sigma_{HW}}{\sqrt{\sigma_{HW}^2\sigma_{|\epsilon|}^2 + \mu_{|\epsilon|}^2\sigma_{HW}^2 + \mu_{HW}^2\sigma_{|\epsilon|}^2 + \frac{\sigma_b^2}{N}}} \tag{8.10}$$

With this, we can proceed to solve equation 8.10 for the different distributions obtained at distinct temperatures for a given body bias scheme. The results can be seen in Fig. 8.3 , where equation 8.10 is plotted as a function of N samples for the different body bias schemes depicted in Table 8.2 at 80 C.

It can be seen that both the DDCB and the SDCB present the most resilience against noise averaging, with the SDCB scheme being slightly better. However, the results are not as good as the ones shown in Fig. 8.2.

Another important aspect to note of equation 8.10, as opposed to the similar equation presented in Chapter 5 is the fact that the variance of ϵ, σ_ϵ^2, does not scale with sampling. This 'noise' is not introduced by a varying body bias, but by the fact that, due to process and mismatch variability, each register consumes different amounts of leakage current for a given body bias. Thus, the maximum PCC has a ceiling and cannot effectively reach a value of 1 regardless of the amount of samples that are averaged.

8.3 Variability analysis - Register Instances

In essence, the above analysis presumes that the registers under attack follow ideally the distribution that arises from the Montecarlo simulations. However, if attacks are performed on 8-bit register slices, their number is low enough that

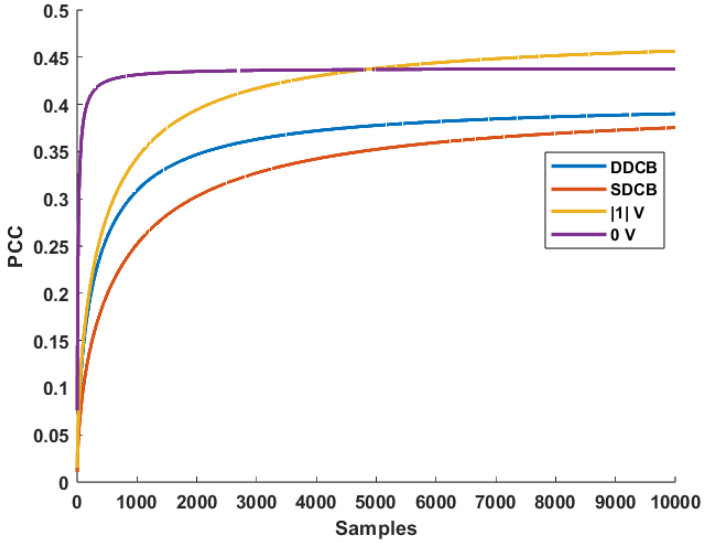

Figure 8.3: PCC as a function of samples of Equation 8.10 at 80 C for the different distributions obtained through a Montecarlo analysis of registers under body bias conditions seen in Table 8.2 considering a folded distribution

no guarantee can be made regarding their true distribution. Thus, registers must be treated as random instances of the distribution.

Consider, then, for ease on analysis, an attack on two registers. These registers, being instances of some random distribution, have fixed leakage current consumption at a given temperature and body bias point. The total leakage consumption of this 2-bit register array can then be expressed as:

$$I_{leak} = I_{1,1} \cdot x_1 + I_{0,1} \cdot \overline{x}_1 + I_{1,2} \cdot x_2 + I_{0,2} \cdot \overline{x}_2 \tag{8.11}$$

Where $I_{1,i}$ represents the leakage current consumed by register i when it stores a 1 (x_i), and $I_{0,i}$ is the leakage current consumed by register i when it stores a 0 ($\overline{x}_i$). x_i is a random variable representing bit i, where $\overline{x}_i$ is the bit complement of x_i.

Thus, for example, if $x_1 = 0$ and $x_2 = 1$, the total leakage current consumption of this two bit array is:

$$I_{leak} = I_{0,1} + I_{1,2} \tag{8.12}$$

Note that $I_{1,i}$ and $I_{0,i}$ are now constant parameters, being instances of the distributions that would be obtained from a Montecarlo simulation.

If we want to obtain the PCC between the leakage current consumed by this 2-bit register array, and the HW of the bits stored within it ($x_1 + x_2$), we again must develop the following expressions:

$$E[(x_1 + x_2) \cdot I_{leak}]$$

$$E[I_{leak}]$$

$$E[x_1 + x_2]$$

$$Var(I_{leak}) \tag{8.13}$$

The complete derivation can be found in the Annexes 10.4 . Thus, for an n-bit register array with process and mismatch variation, we have that the PCC between the leakage current and the Hamming Weight can be expressed as:

$$\rho_{I_{leak},HW} = \frac{1}{\sqrt{n}} \cdot \frac{\epsilon_1 + \epsilon_2 + ... + \epsilon_n}{\sqrt{\epsilon_1^2 + \epsilon_2^2 + ... + \epsilon_n^2}} \tag{8.14}$$

Where every ϵ_i is a constant, obtained from a realization of distribution obtained from the Montecarlo simulation.

Note that Equation 8.14 better clarifies the fact that, unless all the values of ϵ_i are equal, no matter how many samples are taken and averaged, the PCC has a ceiling that depends on how much leakage current each registers consumes for a given body bias point at a given temperature (see Fig. 8.3).

8.3.1 Folded Normal Distribution

Another point of consideration for registers under the DDCB or the SDCB, where the mean of ϵ is relatively small as compared to their standard deviation (Table 8.2) is the following. For some registers, ϵ_i could be positive, and for other registers ϵ_i could be negative. In essence it is possible that some register combinations cancel the numerator of equation 8.14, such that:

$$\sum_{i=1}^{n} \epsilon_i \approx 0 \tag{8.15}$$

while no such cancellation happens in the denominator.

Consider now, for the moment, the dummy cryptosystem introduced in 5.2. In this dummy cryptosystem, the only "cryptographic" function present is the XORing between the plaintext and the secret key, the result of which is stored in the register array.

Consider, under these circumstances, a single register whose leakage current profile can be expressed as:

$$I_{leak} = I_1 \cdot x_1 + I_0 \cdot \bar{x}_1 \tag{8.16}$$

Assume that $x_1 = p_1 \oplus k_1$, where p_1 is the plaintext bit and k_1 the key bit. Under a correct key assumption, the covariance between the leakage current and the bit x_1 can be shown to be:

$$Cov(I_{leak}, x_1) = \frac{1}{4}\epsilon \tag{8.17}$$

Where $\epsilon = I_1 - I_0$.

However, for an incorrect key bit; that is, for $x_1 = p_1 \oplus \bar{k}_1$, the covariance between the leakage current and the bit x_1 is equal to $Cov(I_{leak}, x_1) = -\frac{1}{4}\epsilon$. Thus, for a particular register, the covariance between its leakage current and the bit x_1 can be expressed as:

$$Cov(I_{leak}, x_1) = (-1)^{D_1} \cdot \frac{1}{4}\epsilon \tag{8.18}$$

Where $D_1 = k_1 \oplus k_1'$ is the Hamming Distance between the correct key bit and another potential key bit. Under these conditions, the numerator of Equation (8.14) becomes:

$$(-1)^{D_1} \cdot \epsilon_1 + (-1)^{D_2} \cdot \epsilon_2 + ... + (-1)^{D_n} \cdot \epsilon_n \tag{8.19}$$

Where the vector $D = K_{correct} \oplus K_{incorrect}$. What this equation implies is that, while in some instances the PCC between the leakage consumption and Hamming Weight under a correct key assumption might be very small, under process and mismatch variability it might highly correlate with a false key. Under such conditions, finding the incorrect key implies simultaneously finding the correct secret key, since the leakage current would be adequately correlated to the Hamming Weight of $P \oplus K_{correct} \oplus K_{incorrect}$, with P being the plaintext vector.

An example of this can be seen in the Trivium stream cipher. Consider an attack on 8-bits during the fist 12 periods of the initialization phase (Table 6.1). After 8 periods, the Y register of the Trivium stores the following Hamming Weight (disregarding algorithmic noise):

$$HW = \sum_{j=0}^{m-1} k_{66-j} \oplus IV_{78-j} \tag{8.20}$$

For $m = 8$.

Consider, also, that the register array in which these bits are stores has, under a current balancing body bias scheme, the following leakage consumption:

$$\begin{aligned}
\epsilon_1 &= -\epsilon_2 = 1 \text{ nA} \\
\epsilon_3 &= -\epsilon_4 = 0.5 \text{ nA} \\
\epsilon_5 &= -\epsilon_6 = 0.25 \text{ nA} \\
\epsilon_7 &= -\epsilon_8 = 0.125 \text{ nA}
\end{aligned}$$

The summation of these parameters leads to a total cancellation:

$$\sum_{i=1}^{8} \epsilon_i = 0 \text{ nA} \tag{8.21}$$

Which means that, if one where to perform an attack on the secret key, the PCC of the correct key would appear to be 0. However, there would be an incorrect key with which the PCC would be highly correlated. In fact, the incorrect key with which the attack would highly correlate for the example given, would be one with the following property:

$$K_{correct} \oplus K_{false} = 01010101 \tag{8.22}$$

That is, there is a false key whose Hamming Distance with the correct key is determined by variability. In this case, were one to calculate the PCC using equation 8.14, the result would be $\rho = 0$ for the secret key, and $\rho = 0.813$ for this false key. However, by simply XORing the false key with all the prospective keys, the correct key would be identified.

While this introduces a layer of confusion, this implies that, under process and mismatch variability, the PCC can be a poor predictor of protection for register slice leakage power attacks under current balancing countermeasures unless corrections are introduced.

Thus, it is better to consider the modification of Equation 8.14 where the elements of leakage consumption are considered in their absolute value:

$$\rho_{I_{leak},HW} = \frac{1}{\sqrt{n}} \cdot \frac{|\epsilon_1| + |\epsilon_2| + ... + |\epsilon_n|}{\sqrt{\epsilon_1^2 + \epsilon_2^2 + ... + \epsilon_n^2}} \tag{8.23}$$

This is why we have calculated the PCC in the previous section using the folded normal distribution.

Considering the presence of algorithmic and non-algorithmic noise, Equation 8.23 becomes:

$$\rho_{I_{leak},HW} = \frac{1}{\sqrt{n}} \cdot \frac{|\epsilon_1| + |\epsilon_2| + ... + |\epsilon_n|}{\sqrt{(\epsilon_1^2 + \epsilon_2^2 + ... + \epsilon_n^2) + \frac{1}{N}(\epsilon_{n+1}^2 + ... + \epsilon_m^2) + \frac{4}{N} \cdot \sigma_b^2}} \tag{8.24}$$

Where ϵ_{n+1} through ϵ_m represent the leakage consumption of the bits not pertinent to the attack (algorithmic noise), and σ_b^2 is the noise power of non-algorithmic noise (thermal, measurement noise...). N is, again, the number of samples taken and averaged.

Note that this explanation and derivation is valid for the case where the cryptographic function is simply the XORing between the plaintext and the secret key. If the intermediate variables had a more complex relationship to the plaintext and secret key, as is the case of the AES, these considerations might not be necessary. The AES presents a non-linear function in the form of its substitution box, such that the value stored in the state matrix ends up being $S(p \oplus k)$, with $S(\cdot)$ being the substitution function. The substitution box, being a non-linear function, might erase the potential correlation with a false key.

Thus, for the case of the AES, the absolute values of ϵ_i might not need to be contemplated. At the very least, the author of this text has not been able to find a way in which to replicate for the AES the effect considered for the

Trivium stream cipher under a current balancing scheme. That does not mean that these effects do not exist, and careful considerations must be made in the future. Nonetheless, results for both cases will be presented.

8.3.2 Results

PCC

In order to present the results, we make use of the distributions depicted in Table 8.2. With the help of Matlab, we define a distribution object $\epsilon \sim \mathcal{N}(\mu, \sigma)$, and we simulate 1000 instances of 8-bits, each taken at random from the defined distribution. For each of these 1000 instances, we also produce 120-bits following the same distribution, to account for algorithmic noise. Thus, with a noise power $\sigma_b^2 = -134$ dBW, we calculate equation 8.24 for a variety of N samples.

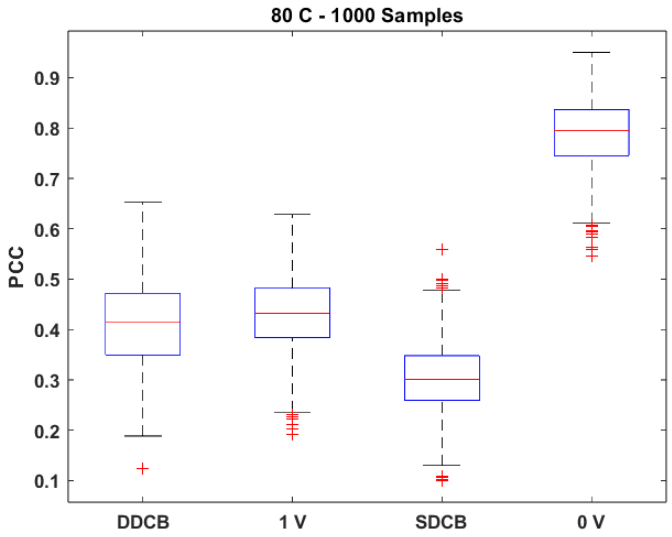

Figure 8.4: Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 1000 samples per instance

The results can be seen in Figs. 8.4 through 8.6. While the mean PCC is not quite as that of Fig. 8.3, the trends observed are repeated here. In all cases, the SDCB body biasing scheme presents the smallest PCC and resilience to noise averaging.

We present the same results for the case of $N = 10000$ at 50 and 30 ^{o}C (Figs. 8.7 and 8.8)

It can be seen that, at lower temperatures, the DDCB body bias scheme is not quite as protective as simply maintaining the body bias at |1| V. The SDCB scheme, however, is still the most protective one.

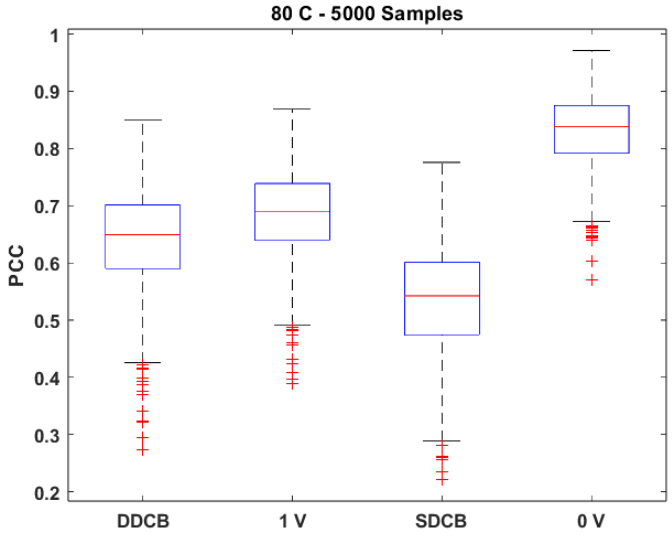

Figure 8.5: Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 5000 samples per instance

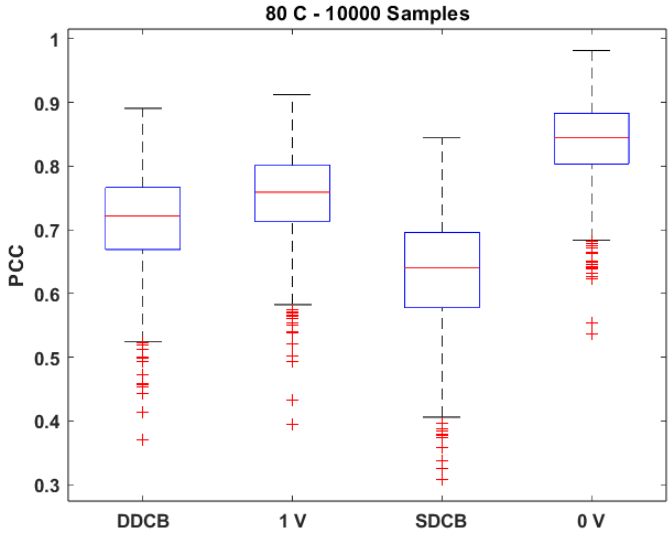

Figure 8.6: Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance

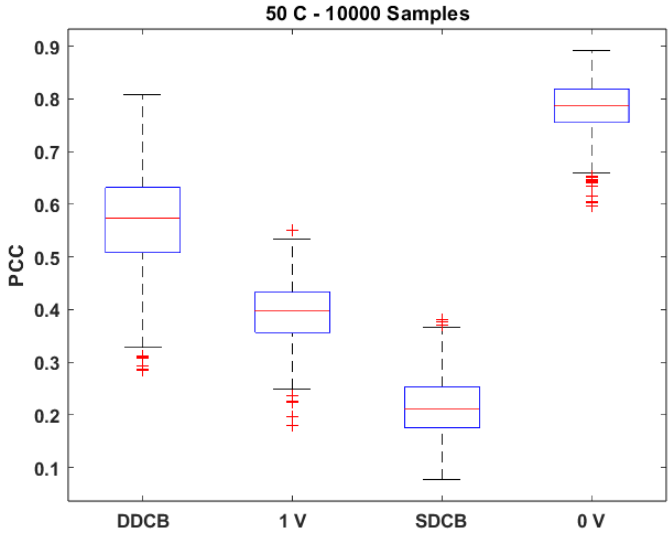

Figure 8.7: Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance

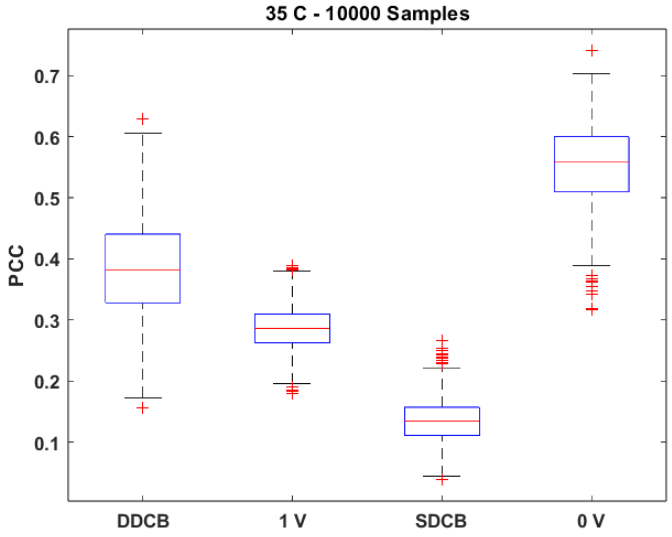

Figure 8.8: Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance

These figures are obtained considering a folded normal distribution. That is, each instance is considered in its absolute value.

We also perform the same simulation for the case of the AES, where the folded normal distribution is not used. The results can be seen in Fig. 8.9.

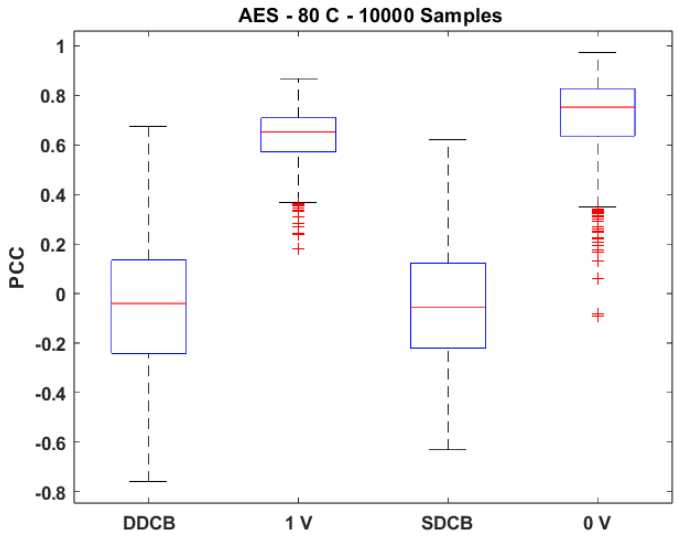

Figure 8.9: Boxplot for a 1000 instances of equation 8.24 for different body bias schemes at 80 C and 10000 samples per instance without considering a folded normal distribution

It can be seen in Fig. 8.9 that, even with 10000 averaged samples, both the DDCB and SDCB schemes present mean PCC around a value of 0. In theory, this would imply a great degree of protection. However, the author cannot guarantee that there does not exist a way to undo the effect of the non-linearities of the S-box onto the variabilities of the registers. In those cases, the results of Fig. 8.9 would be those of Fig. 8.6

Simulated CPA

We next attempt to determine the difficulty with which the secret key can be identified for the different body bias schemes. In order to do so, we again invoke a 1000 instances of 8 registers, and perform a simulated CPA on each of the instances.

In order to do so, we create a vector of 8 ϵ values for each of the instances:

$$\vec{\epsilon} = [\epsilon_1, \epsilon_2, ..., \epsilon_8] \tag{8.25}$$

At the same time, we calculate the matrix of plaintext and secret key XOR bits. That is, for each plaintext value $P = i$, with $0 \leq i \leq 255$, we XOR the

value P with the secret key and convert the result into an 1x8 vector of bits:

$$P_i \oplus k = [p_{i,1} \oplus k_1, p_{i,2} \oplus k_2, ..., p_{i,8} \oplus k_8] \tag{8.26}$$

With this, we obtain a 256x8 matrix, one row for each plaintext value:

$$\begin{bmatrix} p_{1,1} \oplus k_1 & p_{1,2} \oplus k_2 & \dots & p_{1,8} \oplus k_8 \\ p_{2,1} \oplus k_1 & p_{2,2} \oplus k_2 & \dots & p_{2,8} \oplus k_8 \\ \vdots & \vdots & \vdots & \vdots \\ p_{256,1} \oplus k_1 & p_{256,2} \oplus k_2 & \dots & p_{256,8} \oplus k_8 \end{bmatrix} \tag{8.27}$$

Thus, we calculate the vector of leakage currents by computing the vector product between each of the rows of plaintext-key matrix and the $\vec{\epsilon}$ vector:

$$I_{leak_i} = \sum_{j=1}^{8} (p_{i,j} \oplus k_j) \cdot \epsilon_j \tag{8.28}$$

Obtaining a vector of leakage current consumption for each plaintext value for a given instance of 8-bits. We then add the noise components to this vector, averaged according to the number of samples taken. Finally, we calculate the PCC between this leakage current vector and the Hamming Weight of each possible plaintext-key combination.

With this, we note the amount of times that the secret key is identified.

Table 8.3: Success rate of key identification for 1000 instances of a simulated CPA at 80 C

Folded Distribution - 80 C				
	DDCB	SDCB	|1| V	0 V
N= 1000	0.335	0.228	0.459	0.833
N= 5000	0.626	0.496	0.686	0.934
N= 10000	0.717	0.631	0.794	0.941

Table 8.4: Success rate of key identification for 1000 instances of a simulated CPA at 50 C

Folded Distribution - 50 C				
	DDCB	SDCB	|1| V	0 V
N= 1000	0.252	0.025	0.130	0.736
N= 5000	0.521	0.121	0.482	0.900
N= 10000	0.607	0.199	0.582	0.948

Table 8.5: Success rate of key identification for 1000 instances of a simulated CPA at 35 C

Folded Distribution - 35 C				
	DDCB	SDCB	|1| V	0 V
N= 1000	0.103	0.014	0.062	0.291
N= 5000	0.307	0.040	0.330	0.690
N= 10000	0.440	0.107	0.556	0.793

The results can be seen in Tables 8.3 through 8.5, for the cases in which the folded normal distribution has been contemplated. The rate of successful key identification more or less follows the trends observed in the boxplots of Figs. 8.4 through 8.8. At higher temperatures, both the DDCB and SDCB schemes, specially the SDCB one are the most resilient to noise averaging. At 50 C, the DDCB appears to be less functional than the |1| V scheme, but even then the rate of key identification is not too high.

At lower temperatures the SDCB scheme presents a very poor rate of secret key identification, a testament for, in principle, the great effectiveness of this countermeasure to protect the cryptosystem against leakage power analysis attacks.

Note that averaging 10000 samples implies a total of 256*10000 measurements. After these huge quantity of measurements, with the SDCB body biasing scheme the secret key can only be disclosed with, at most, a frequency of 0.63 at high temperatures.

Table 8.6 presents the rate of key identification for a 1000 instances extracted with a normal distribution, disregarding the absolute value, for the potential AES case.

Table 8.6: Success rate of key identification for 1000 instances of a simulated CPA at 80 C

Normal distribution (AES) - 80 C				
	DDCB	SDCB	|1| V	0 V
N= 1000	0.377	0.248	0.962	0.994
N= 5000	0.590	0.512	≈1	≈1
N= 10000	0.628	0.575	≈1	≈1

8.4 Conclusions

In this chapter, the effect of process and mismatch variations regarding the effectiveness of the current balancing circuit as a countermeasure is explored.

In order to do so, Montecarlo simulations are employed to extract the distribution of ϵ obtained under the conditions presented by the current balancing circuit.

With this, a statistical power model of a register array under attack with the considerations of variability is analyzed for different possible body bias schemes and temperatures.

The case for the use of a folded normal distribution in the analysis is presented and taken into account in the analysis. Relying again on calculations on the PCC and simulated CPA attacks, the scheme that presents the highest degree of protecting for all possible conditions is the one described as Single Driver Current Balancing. While not as resilient as the results obtained in the ideal case of Chapter 6, the results still indicate a high degree of protection, even at high temperatures, where the secret key cannot be disclosed more than 0.63 of the times the experiment is run, even with enormous amounts of averaged traces.

Again, the results here presented are limited by the fact that they rely exclusively on numerical simulations. The errors in the idealities presented here might compound with the idealities of the current balancing circuit, but the results seen are sufficiently effective so as to warrant further investigation in a potential physical implementation.

Chapter 9

Conclusions

In this thesis, an exploration on the nature of leakage current consumption of registers implemented with FDSOI as a function of their body bias has been carried out. This assessment has been produced under the framework of understanding how leakage currents of bitsliced implementations of cryptosystems can be utilized as a side-channel to perform leakage power analysis attacks. Given the possibility to dynamically modify the threshold voltage of FDSOI transistors through their fourth terminal and, indirectly, modify the leakage current profile of registers, this work has sought to determine whether provable secure countermeasures can be developed using this transistor technology.

In order to do so, an initial understanding of the leakage current profiles of FDSOI registers as a function of their body bias is obtained, as detailed in Chapter 5. The exploration goes on to determine which magnitudes have an influence on the pearson correlation coefficient of registers implementing dummy cryptosystems, particularly as related to their body bias.

With this, an initial countermeasure based on the random application of a symmetric body bias level at the beginning of each encryption process is proposed. This countermeasure is modelled to include the effect of sources of algorithmic and non-algorithmic noise, as well as explored under different temperature and technological conditions (FBB vs RBB). This initial countermeasure has proven to force a potential attacker to significantly increase the number of traces required to be able to disclose the secret key, hindering attack efforts. It is also seen that this countermeasure benefits from higher temperatures when register are implemented with FBB transistors, while RBB transistors are penalized in this sense, given their reduced capability to introduce noise and thus decorrelate the power traces from the data being processed; that is, the noise they are able to introduce does not sufficiently scale with the increase of the floor of their leakage current consumption, facilitating the attack at higher temperatures.

Unfortunately, further exploration on the validity of this initial proposal reveals certain vulnerabilities than can be exploited to render it practically useless. A case model is developed for two particular cryptographic algorithms,

the AES and the Trivium stream cipher, detailing how a bivariate power attack that takes advantage of a known state can practically bypass the entire protective effect that a random symmetric body bias provides. This prompted further exploration, particularly regarding the asymmetric application of a body bias, seeking whether pair of body bias values (positive and negative) exist such that FDSOI registers see the disparities in leakage current consumption depending on data processed diminished. This current balancing body bias is, initially, explored from a purely mathematical perspective, relying on electrical simulation models to extract pertinent parameters. The author is then able to determine the validity of this approach, which not only significantly increases the number of traces required to disclose the secret key, but also offers protection against bivariate attacks.

This approach is taken further by designing the necessary circuits so as to be able to implement this theoretical countermeasure in an autonomous manner. That is, a circuit that automatically seeks a body bias point which balances the leakage current consumption of registers, regardless of the data they store. This is achieved through the dynamic modification of the body bias wells while the cryptographic system is in operation. Once the circuit is implemented, simulations showcase its capability to seek and maintain a body bias value that highly decreases the leakage current disparities of circuits, achieving functionality for a wide range of temperatures. With this proof of concept circuit and the results obtained, further statistical analysis in the form of PCC figures of merit under the conditions obtained are performed, showcasing the success of this countermeasure implementation in greatly hindering the ability of an attacker to extract the secret key.

To further validate the proposed countermeasure, an analysis is performed considering the effect of mismatch and process variability that could somehow impact the effectiveness of the body bias seeking circuit. Montecarlo simulations are performed to obtain the probability density functions of pertinent magnitudes, and a statistical model developed under the conditions obtained through the body bias seeking circuit to consider the effect of variability. At the end, it is determined that variability between registers within the same die does affect the level of protection provided by the body bias seeking circuit. Nonetheless, the countermeasure does continue to offer a great degree of protection, particularly in the case of the Single Drive Current Balancing scheme. This level of protection varies with temperature but remains relatively high, forcing an attacker to perform great many averages per trace just to be able to disclose the secret key with poor probability.

In essence, the main objective of this thesis has been accomplished, in as much as the work has culminated in a proof of concept circuit that acts as an effective countermeasure to leakage current power analysis attacks for a wide range of temperature. The circuit is simple enough so as to present low area overhead, and while no attempts have been carried out in this work, optimization efforts could reduce its power consumption.

Again, given its simplicity, the circuit could exist as a given module, imported during the design of a cryptographic implementation, to easily garner its

benefits.

At the same time, the circuit does not interfere with other potential countermeasures, which could also be applied to further the level of protection against leakage or dynamic power attacks.

A case must be made for the use of FDSOI technology. Firstly, because the countermeasure necessitates that the implementation be made with such technology. But also given the properties of FDSOI transistors in and of themselves. This relatively new technology has been able to, at the very least, curtail some short-channel effects, particularly those related to electrostatic stability of the channel and leakage current. Thus, this technology inherently offers a certain degree of protection against LPA as compared to other bulk technologies, given the smaller magnitudes of static power consumption that they incur, even at higher temperatures.

Thus, the implementation of cryptographic algorithm in low-power circuits could greatly benefit from such a combination of FDSOI technology and current balancing countermeasure. Among other cases, with the prospective increase of IoT nodes populating diverse settings, such a countermeasure could offer benefits in security sensitive applications. This extra layer of protection could curtail some of the security issues that arise in an interconnected world, where the ability to have a network of decentralized information 'objects' has taken precedent over their security implications.

9.1 Publications

During the development of this thesis the following publications have been produced:

- K. Palma and F. Moll, "Analysis of Random Body Bias Application in FD-SOI Cryptosystems as a Countermeasure to Leakage-Based Power Analysis Attacks," in IEEE Access, vol. 9, pp. 114977-114988, 2021, doi: 10.1109/ACCESS.2021.3105635.

- K. Palma and F. Moll, "Current Balancing Random Body Bias in FD-SOI Cryptosystems as a Countermeasure to Leakage Power Analysis Attacks," in IEEE Access, vol. 10, pp. 13451-13459, 2022, doi: 10.1109/ACCESS.2022.3144639.

- K. Palma and F. Moll, "Simulated Leakage Power Analysis Attack of the Trivium Stream Cipher," 2022 37th Conference on Design of Circuits and Integrated Circuits (DCIS), Pamplona, Spain, 2022, pp. 01-06, doi: 10.1109/DCIS55711.2022.9970061.

Chapter 10

Annexes

10.1 Annex I - Perfect Secrecy

Consider a cryptosytem formed by a plaintext space (X), a key space (K), and a ciphertext space (Y), and assume that the plaintext and key space follow a particular probability distribution such that:

$$\sum_i P[X = i] = 1$$
$$\sum_j P[K = j] = 1.$$

And for each element of the either of the above distributions, the probability of any given element is greater or equal than zero.

The probability of the ciphertext can be defined by the above probability distributions, in as much as any element $Y = y$ of the ciphertext is the result of an encryption function $e_k(x)$. Assuming that each key and plaintext element are independent from each other, we have:

$$P[Y = y] = \sum_{\{K:y \in Y(K)\}} P[K = k] \cdot P[X = d_k(y)] \tag{10.1}$$

Where equation 10.1 states that the probability of obtaining a given ciphertext element is the sum of the probability between all pair of keys and plaintext that yield that particular element. The product is a result of their being independent from each other.

From this it also follows that the conditional probability of obtaining a ciphertext element given a plaintext element can be expressed as:

$$P[Y = y|X = x] = \sum_{\{K:x=d_k(y)\}} P[K = k] \tag{10.2}$$

With these notions it is sufficient to express the problem at the heart of perfect secrecy as:

$$P[X = x|Y = y] = \frac{P[X = x] \cdot \sum\limits_{\{K:x=d_k(y)\}} P[K = k]}{\sum\limits_{\{K:y\in Y(K)\}} P[K = k] \cdot P[X = d_k(y)]} \tag{10.3}$$

Equation 10.3 represents the conditional probability of x being a particular plaintext element given a known ciphertext element.

A perfectly secret cryptosytem is one in which the following equality holds:

$$P[X = x|Y = y] = P[X = x] \tag{10.4}$$

That is, the probability of the plaintext distribution is not altered by having ciphertext elements.

In the case where the order of each set $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{K}|$, where the cryptosystem is a bijection such that there is no collition between plaintext elements (that is, $e_{k1}(x) = e_{k2}(x)$ only if $k_1 = k_2$) some of the above notions can be simplified.

Considered a fixed ciphertext element y. In a bijective cryptosystem, for each possible plaintext element x_i there is only one possible key k_i that yield that ciphertext element.

Equation 10.3 then becomes:

$$P[X = x_i|Y = y] = \frac{P[X = x_i] \cdot P[K = k_i]}{P[Y = y]} \tag{10.5}$$

Under such circumstances, the conditions of perfect secrecy is met when $P[K = k_i] = P[Y = y]$.

As an example, consider a cryptosystem formed by $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}| = |\{0, 1\}^1|$, where the encrypting function is the XORing between a plaintext bit and a key bit:

$$y = x \oplus k \tag{10.6}$$

Assume that the bit y follows a uniform probability distribution such that:

$$P[Y = 1] = P[Y = 0] = \frac{1}{2} \tag{10.7}$$

For a given ciphertext element (namely, y=1), there are two possible key and plaintext combinations that yield this result. As such, as long as the keyspace follows a discrete uniform distribution ($P[K = i] = \frac{1}{|K|}, 0 \leq i \leq 1$), the condition holds. Under such conditions, having access to a ciphertext element provides an eavesdropper no further information regarding the plaintext. Note that the condition holds only as long as each plaintext element is encrypted with a distinct key, chosen at random, every time.

10.2 Annex II - PCC Derivation Under Random Body Bias Scheme

In order to determine the PCC between the Hamming Weight and leakage current consumption I_{leak} of a register array, we need to calculate the Covariance $E[I_{leak} \cdot HW]$, and the expected values $E[I_{leak}]$ and $E[HW]$.

Going back to equation 5.18 we can see that:

$$E[I_{leak}(S) \cdot HW] = E[n \cdot I_0(S) \cdot HW] + E[\epsilon(S) \cdot HW^2] \tag{10.8}$$

Where $I_0(S)$ and $\epsilon(S)$ are no longer constants, but random variables.

Assuming, for the moment, that $E[I_0(S)]$ and $E[\epsilon(S)]$ are well defined and equal to, respectively, μ_{I0} and μ_ϵ, and taking into account that S and HW are random variables independent from each other:

$$E[I_{leak}(S) \cdot HW] = n \cdot \mu_{I0} \cdot \mu_{HW} + \mu_\epsilon \cdot E[HW^2]$$

$$E[I_{leak}(S)] = n \cdot \mu_{I0} + \mu_\epsilon \cdot \mu_{HW}$$

$$E[HW] = \mu_{HW} \tag{10.9}$$

And, finally, the covariance between the Hamming Weight and the leakage current consumption of the register array in the presence of the countermeasure is:

$$Cov(I_{leak}(S), HW) = \mu_\epsilon \cdot \sigma^2_{HW} \tag{10.10}$$

Next, we need to determine the variance of $I_{leak}(S)$. The expression of the variance of the leakage current is cumbersome. For ease of readability, it can be broken down into three components:

$$Var(I_{leak}) = Var(n \cdot I_0(S) + \epsilon(S) \cdot HW) = p1 + p2 + p3 \tag{10.11}$$

Where, taking into account that HW and $\epsilon(S)$ are independent random variables:

$$p_1 = Var(HW \cdot \epsilon(S)) =$$

$$= \sigma^2_{HW}\sigma^2_\epsilon + \mu^2_\epsilon\sigma^2_{HW} + \mu^2_{HW}\sigma^2_\epsilon \tag{10.12}$$

Where we have use the fact that HW and $\epsilon(S)$ are independent from one another. Also:

$$p_2 = Var(n \cdot I_0(S)) = n^2 \cdot \sigma^2_{I0} \tag{10.13}$$

And finally:

$$p_3 = 2 \cdot Cov(HW \cdot \epsilon(S), n \cdot I_0(S)) =$$

$$= 2 \cdot n \cdot \mu_{HW} \cdot Cov(\epsilon(S), I_0(S)) \tag{10.14}$$

where σ_ϵ^2 is the variance of $\epsilon(S)$ and μ_{I0} and σ_{I0}^2 the expected value and variance of $I_0(S)$.

Next we need to determine the expected value of $\epsilon(S)$, $I_0(S)$, and their variance.

In order to do so, we use the definition of the expected value for discrete random variables.

$$E[X] = \sum_{i=0}^{n} x_i \cdot P[X = x_i] \tag{10.15}$$

Having $\epsilon(S) = a_\epsilon \cdot e^{b_\epsilon \cdot (V_{BBq} + S \cdot \Delta V_{bb})}$, and with S being able to adopt any integer value between $-s_{max}$ and s_{max}, with probability:

$$P[S = i] = \frac{1}{2s_{max} + 1}$$

$$\forall i, -s_{max} \leq i \leq s_{max} \tag{10.16}$$

With this, we have:

$$E[\epsilon(S)] = \sum_{i=-s_{max}}^{s_{max}} \frac{1}{2s_{max} + 1} \cdot a_\epsilon \cdot e^{b_\epsilon \cdot (V_{BBq} + i \cdot \Delta V_{bb})} \tag{10.17}$$

This expression can be solved as a geometric series once we have performed an index shift. Consider $m = i + s_{max}$. Then:

$$E[\epsilon(S)] = \frac{1}{2s_{max} + 1} \cdot \sum_{m=0}^{2 \cdot s_{max}} a_\epsilon \cdot e^{b_\epsilon \cdot (V_{BBq} + (m - s_{max}) \cdot \Delta V_{bb})} =$$

$$\frac{1}{2s_{max} + 1} \cdot a_\epsilon \cdot e^{b_\epsilon \cdot V_{BBq}} \cdot e^{-b_\epsilon \cdot s_{max} \cdot \Delta V_{bb}} \cdot \sum_{m=0}^{2 \cdot s_{max}} e^{m \cdot b_\epsilon \cdot \Delta V_{bb}}$$

And using the fact that, for a geometric series:

$$\sum_{i=0}^{n-1} a \cdot r^i = a \cdot \left(\frac{1 - r^n}{1 - r}\right) \tag{10.18}$$

We can finally express the expected value of $\epsilon(S)$ as:

$$E[\epsilon(S)] = \frac{1}{2s_{max} + 1} \cdot a_\epsilon \cdot e^{b_\epsilon \cdot V_{BBq}} \cdot e^{-b_\epsilon \cdot s_{max} \cdot \Delta V_{bb}} \cdot \frac{(1 - e^{b_\epsilon \cdot \Delta V_{bb} \cdot (2s_{max} + 1)})}{(1 - e^{b_\epsilon \cdot \Delta V_{bb}})} \tag{10.19}$$

The same procedure can be use to calculate the expected value of $I_0(S)$.

To calculate the variance of $I_0(S)$ and $\epsilon(S)$ we use the definition $Var(X) = E[X^2] - E[X]^2$.

We note that $E[X^2]$ can be calculated using the definition:

$$E[X] = \sum_{i=0}^{n} x_i^2 \cdot P[X = x_i] \tag{10.20}$$

Which leads to:

$$E[\epsilon^2(S)] = \frac{1}{2s_{max}+1} \cdot a_\epsilon^2 \cdot e^{2 \cdot b_\epsilon \cdot V_{BBq}} \cdot e^{-2 \cdot b_\epsilon \cdot s_{max} \cdot \Delta V_{bb}} \cdot \frac{(1 - e^{2 \cdot b_\epsilon \cdot \Delta V_{bb} \cdot (2s_{max}+1)})}{(1 - e^{2 \cdot b_\epsilon \cdot \Delta V_{bb}})} \tag{10.21}$$

And the variance can be calculated by simply subtracting equation 10.19 from equation 10.21.

Note that the Covariance between $\epsilon(S)$ and $I_0(S)$ appears in Equation (10.14). To compute it, we used the definition:

$$Cov(\epsilon(S), I_0(S)) = E[\epsilon(S) \cdot I_0(S)] - E[\epsilon(S)]E[I_0(S)] \tag{10.22}$$

Both $E[\epsilon(S)]$ and $[I_0(S)]$ are already well defined. Since each realization of S is independent from each other, the covariance between experiments is zero. Thus, only the intra-experiment covariance is meaningful. In this way, the covariance can be calculated by noting that $E[\epsilon(S) \cdot I_0(S)]$ is simply:

$$E[\epsilon(S) \cdot I_0(S)] = \frac{1}{2s_{max}+1} \sum_i \epsilon(s_i) I_0(s_i) =$$

$$\frac{1}{2s_{max}+1} \cdot (a_\epsilon \cdot a_{I0}) \cdot e^{(b_\epsilon + b_{I0}) \cdot V_{BBq}} \cdot e^{(b_\epsilon + b_{I0}) \cdot s_{max} \cdot \Delta V_{bb}} \cdot$$

$$\cdot \frac{(1 - e^{(b_\epsilon + b_{I0}) \cdot \Delta V_{bb} \cdot (2s_{max}+1)})}{(1 - e^{(b_\epsilon + b_{I0}) \cdot \Delta V_{bb}})} \tag{10.23}$$

10.3 Annex III - Current Balancing Proof of Concept Design

In this section, the process and analysis with which the different parameter values is chosen is thoroughly explained.

10.3.1 V_{ref}

We begin our implementation of the system by determining the different parameters that govern the behaviour of the Time-to-Digital sensor.

In order to do so, we begin by implementing a testbench at the schematic level. The schematic can be seen in Fig. 10.1, where the registers and the sensing capacitors are connected to V_{dd} through a power switch. We perform a transient simulation, whereby both the registers under test and the sensing capacitors are connected to V_{dd} while the registers are loaded with a 1 and a 0. After sufficient time has passed for the circuit to settle, the power switch is disconnected and the registers begin to slowly discharge the sensing capacitors.

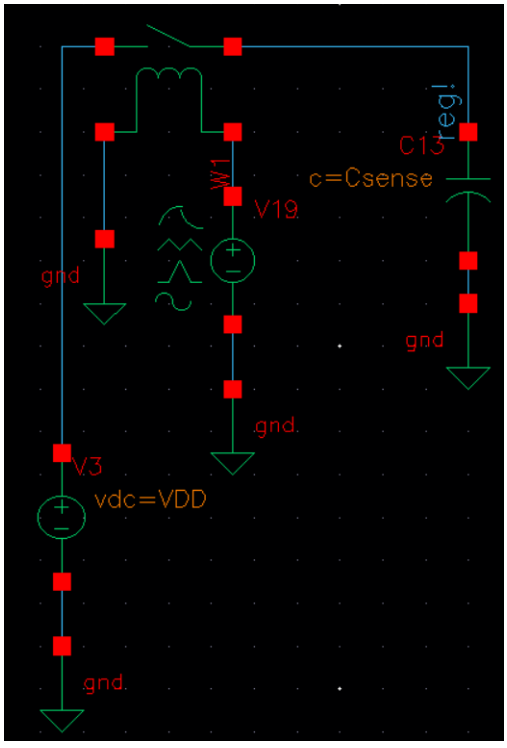

Figure 10.1: Schematic representation of the circuit implementing the capacitive sensing capabilities of the TtD converter. The *reg*! label refers to the power pin of the registers under test

We are interested in seeing which value of ΔV_c (Equation 7.23) produces the best quality of comparison. As stated in previous sections, as the capacitor discharges, the value of V_{dd} that powers the register under test diminishes. At some point, the register will stop behaving at nominal conditions, and its behaviour and current profile might not be defined and/or representative of its nominal operation.

With ΔV_c being equal to $V_{dd} - V_{ref}$, the question at hand is which value of V_{ref} implements the best possible comparison. This value also affects the sensitivity k of the TtD converter (with $k = \Delta V_c \cdot C_{sense} \cdot f_s$).

To determine the value of V_{ref} we take a reverse approach. We begin by manually seeking a body bias point $V_{bbn} \times V_{bbp}$ at a given temperature where the magnitude of ϵ is vanishingly small. We then set the body bias values of the registers under test at these magnitudes, and perform a transient simulation of the testbench schematic.

Thus, both C_{sense} capacitors first charges to V_{dd}, and then are allowed to fully discharge through the registers storing a 1 and storing a 0. The voltage across the capacitors terminals is then measured and compared. The resulting waveform can be seen in Fig. 10.2.

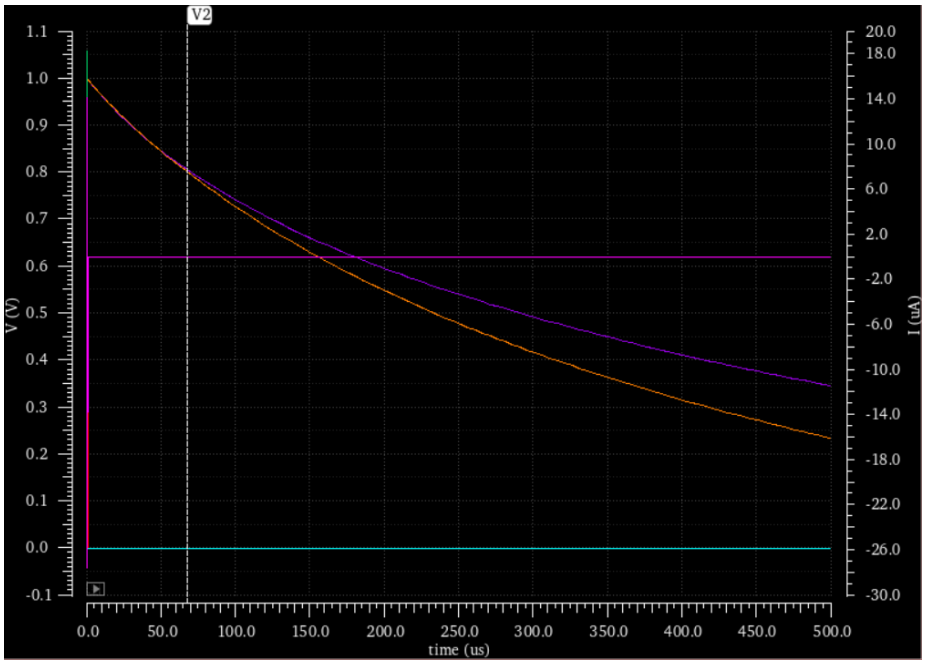

Figure 10.2: V_{c1} (purple) and V_{c0} (orange) as a function of time

It can be seen that, the closer V_{ref} is to V_{dd}, the more similar the disparities between V_{c0} and V_{c1} are at current balancing conditions. This behavior is consistent at all temperatures examined (between 27 ^{o}C and 80 ^{o}C). Beyond

$V_{ref} = 0.8V$, the rate of discharge diverges, indicating that V_{ref} should be kept above 0.8 V at all times.

We next perform the difference of integration of curves V_{c0} and V_{c1}. That is, while the sensor measures and compares the time it take discharge the capacitors, the static current that discharges the capacitors might evolve as the capacitor is discharging. On the other hand, ϵ is measured at static conditions. Thus, for a body bias point (manually sought) in which ϵ becomes vanishingly small, we are interested in determining which value of V_{ref} (above 0.8 V) produces the smallest error.

In order to do so, we compute the equation:

$$\int_{t=0}^{t=T} V_{c0}(t) - V_{c1}(t)dt = error \tag{10.24}$$

Where the value $t = 0$ is the time at which the capacitors begin discharging, and the value $t = T$ is the time at which capacitor V_{c1} has reached the desired value of V_{ref}.

We then compare for which value of V_{ref} such that $V_{c1}(t = T) = V_{ref}$ the error is smaller. We try this for $V_{ref} = 0.85$ V and $V_{ref} = 0.80$ V. We don't go any higher because then the sensor would be loosing sensitivity.

Unfortunately the results are not temperature consistent. That is, for some temperatures, having $V_{ref} = 0.85$ V produces smaller errors, but for other temperatures having $V_{ref} = 0.80$ V produces better results. As such, we finally settle for $V_{ref} = 0.80$ V, which would produce a higher value of ΔV_C and, thus, k.

10.3.2 f_s and C_{sense}

The determination of C_{sense} and f_s does not respond to any empirical understanding, but rather the sensitivity requirements of the TtD converter and limitations of the technology. The 28 nm, FDSOI registers have been seen in other projects within the research group to be relatively fast (up to 1 GHz of operating frequency), particularly for the case of LVT, FBB registers operating at nominal conditions ($V_{dd} = 1$ V). However, registers implemented with RVT, RBB transistors become slower as their body bias increases, so a conservative approach is initially taken with $f_s = 250MHz$.

With this, we are left with the value of C_{sense}, which will be determined depending on the required sensitivity k.

We have seen that, without the hysteresis condition, the sensor can "choke" at given temperatures. Assuming, for the moment, a maximum operating temperature of 80 ºC, and that the system begins operating at this temperature, the sensor must be able to discriminate the leakage currents that arise at these operating conditions. With that, we first observe which is the maximum leakage current achieved at 80ºC, and a body bias condition of $V_{bbn} = V_{bbp} = 0$ V.

Under this conditions, the simulation of the registers under test bench shows a maximum leakage current of $I_{MAX} = 42$ nA. Thus, the sensor must be able

to, at least, detect leakage currents of up to 42 nA. The minimum sensitivity k_{min} can then be calculated with the following equation, solving for $n = 1$.

$$n = \frac{k_{min}}{I_{max}} \tag{10.25}$$

Under these conditions, in order for n to equal 1, $k_{min} = I_{MAX}$. With this, $k_{min} = 42 \cdot 10^{-9}$ A^{-1}. With the conditions that $\Delta V_c = 0.2$ V and $f_s = 250$ MHz, $C_{sense_{min}}$ must equal 0.84 fF.

At the same time, the value of C_{sense} is also constrained by the Hysteresis conditions and the desired ϵ sensitivity.

The hysteresis condition is set such that the leakage currents of registers storing a 0 (I_0) ranges between 10 and 15 nA, such that if the sensor detects that $I_0 > 15$ nA, the hysteresis condition is activated and kept active until $I_0 < 10$ nA.

Assuming the worst case regarding the value of I_0 under hysteresis conditions (that is, $I_0 = 15$ nA), we settle for the moment for a value of $C_{sense} = 750fF$. This sets $k = 3.75 \cdot 10^{-5}$. Assuming that $I_1 \approx I_0 >> \epsilon$, the sensitivity of the sensor can be determined as:

$$n_0 - n_1 = k \cdot \frac{\epsilon}{I_0^2} \tag{10.26}$$

$$1 = 3.75 \cdot 10^{-5} \cdot \frac{\epsilon}{(15 \cdot 10^{-9})^2} \tag{10.27}$$

With this, we have $\Delta\epsilon = 6$ pA.

Note that it might be possible to relax these conditions somewhat. Depending on the granularity of the charge pump, the step increase of the body bias might produce changes in $\epsilon(V_{bbn}, V_{bbp})$ greater than the minimum sensitivity of the sensor.

In fact, if we approximate the gradient of ϵ with its total differential 7.35 and consider a granularity of the Charge Pumps such that it can only modify the body bias through discrete increments of 50 mV, only in very narrow areas is $|d\epsilon| \approx 6$ pA.

At the same time, in order to keep the same level of sensitivity, there is a trade-off between the frequency at which the sensor operates (f_s) and the value of the sensing capacity (C_{sense}). Increasing the frequency by a factor p allows the reduction of the required capacitance by the same factor $k = \Delta V_c \cdot p \cdot f_s \cdot \frac{C_{sense}}{p}$, establishing a trade-off between power and area.

Nonetheless, these values serve just to set an initial proof of concept, and are intended to be functional rather than optimized.

10.3.3 Charge-Pumps

The charge-pumps in this initial design are approximated as the ideal switched circuits depicted in Figs. 7.19 and 7.18.

Both circuits present a capacitive load P_{well} and N_{well} of 1 pF, connected in parallel to the body bias wells of the register under test.

We note that, when the discharging switch is opened and the charging switch is closed, the dynamics governing the charging of the load capacitor can be expressed in the frequency domain as:

$$I_c(S) = \frac{(V_{dd} - V_c)}{R_{switch}} \cdot \frac{1}{S + \frac{1}{R_{switch}C}} \tag{10.28}$$

Converting to the time domain and integrating for the time the signal $CLKP$ is active (T_P), the increase in voltage of the capacitive load can be written as:

$$\Delta V_c = C \cdot (V_{dd} - V_c)(1 - e^{-\frac{T_p}{R_s C}}) \tag{10.29}$$

With this, it can be seen that the amount of charged transferred from the power supply to the capacitive load varies with the voltage currently present between the capacitor terminals. Thus, for a given pulse of duration T_p, the higher the current voltage V_c, the smaller ΔV_c is. Because of this, the granularity of this idealized charge pump be can be solved at the point of maximum charge transfer. That is, when $V_c = 0$ V.

Setting T_p at 1 ns through a counter that increments once the drivers' FSM enters the state Do, and having set the capacitive load to 1 pF, we must only solve for a switch resistance that allows a maximum voltage increase of $\Delta V_c = 50$ mV. This leaves us with a switch resistance of approximately 25 kΩ.

10.3.4 Number of Bits, n_{min} and n_{MAX}

With these, we now determine the number of bits the TtD converter counter comprises, as well as the reference values of the hysteresis comparator.

With a $k = 3.75 \cdot 10^{-5}$ A^{-1}, and presuming a minimum detectable current of $I_0 \approx I_1 = 1$ nA, the number of counts the sensor can reach is:

$$n = \frac{k}{I_0} = \frac{3.75 \cdot 10^{-5}}{1 \cdot 10^{-9}} = 37500 \tag{10.30}$$

And the number of bits required to count up to this number is:

$$bits = \frac{ln(37500)}{ln(2)} \tag{10.31}$$

This amounts to 15.20 bits, which must be rounded up to 16 and, as the bits are signed, further to 17.

Finally, the hysteresis conditions are determined by:

$$n_{Hmax} = \frac{k}{I_{lower}}$$
$$n_{Hmin} = \frac{k}{I_{higher}}$$

With $k = 3.75 \cdot 10^{-5}$ A^{-1} and the lower hysteresis condition set at 10 nA, $n_{Hmax} = 3750$. Liwekise, with the higher hysteresis condition set at 15 nA, we have $n_{Hmin} = 2500$.

10.4 Annex IV - PCC for register instances under process and mismatch variability

In order to derive an expression of the PCC under an instance of two registers under process and mismatch variation, an expression for the covariance must be solved. In the following expressions, x_1 and x_2 represent the bits stored in, respectively, registers 1 and 2; and $I_{1,i}$, $I_{0,i}$ are the leakage current consumption of register i when they store either a 1 or a 0. With this, we have:

$$E[(x_1 + x_2) \cdot I_{leak}] = \\ E[x_1^2 \cdot I_{1,1} + x_2^2 \cdot I_{1,2} + x_2 \cdot x_1 \cdot I_{1,1} + x_2 \cdot \overline{x}_1 \cdot I_{0,1} + x_1 \cdot x_2 \cdot I_{1,2} + x_1 \cdot \overline{x}_2 \cdot I_{0,2}] \tag{10.32}$$

Where we have used the fact that $x_1 \cdot \overline{x}_1 = x_2 \cdot \overline{x}_2 = 0$. Again, assuming that each bit follows a discrete, uniform, unbiased probability distribution, with $E[x_i] = 1/2$ and $Var(x_i) = 1/4$, for every i, and that bits are independent from one another, we have:

$$E[(x_1 + x_2) \cdot I_{leak}] = \frac{3}{4} \cdot (I_{1,1} + I_{1,2}) + \frac{1}{4} \cdot (I_{0,1} + I_{0,2}) \tag{10.33}$$

Continuing with $E[x_1 + x_2]$, we have:

$$E[x_1 + x_2] = 1 \tag{10.34}$$

Then:

$$E[I_{leak}] = E[I_{1,1} \cdot x_1 + I_{0,1} \cdot \overline{x}_1 + I_{1,2} \cdot x_2 + I_{0,2} \cdot \overline{x}_2] = \\ \frac{1}{2} \cdot (I_{1,1} + I_{1,2}) + \frac{1}{2} \cdot (I_{0,1} + I_{0,2}) \tag{10.35}$$

And, finally:

$$Cov(I_{leak}, x_1 + x_2) = E[I_{leak} \cdot (x_1 + x_2)] - E[I_{leak}] \cdot E[x_1 + x_2] = \\ \frac{1}{4} \cdot [I_{1,1} + I_{1,2}] - \frac{1}{4} \cdot [I_{0,1} + I_{0,2}] \tag{10.36}$$

This expression can be rearranged to:

$$Cov(I_{leak}, x_1 + x_2) = \\ \frac{1}{4} \cdot [I_{1,1} - I_{0,1}] + \frac{1}{4} \cdot [I_{1,2} - I_{0,2}] \tag{10.37}$$

The variance of I_{leak} can be solved by noting that x_1 and x_2 (and any combination of their complements) are independent from each other. Thus:

$$Var(I_{leak}) =$$

$$Var(x_1 \cdot I_{1,1} + \overline{x}_1 \cdot I_{0,1}) + Var(x_2 \cdot I_{1,2} + \overline{x}_2 \cdot I_{0,2}) \tag{10.38}$$

With this, we have:

$$Var(x_1 \cdot I_{1,1} + \overline{x}_1 \cdot I_{0,1}) = \frac{1}{4} \cdot (I_{1,1}^2 + I_{0,1}^2 - 2 \cdot I_{1,1} \cdot I_{0,1}) \tag{10.39}$$

Which can be simplified to:

$$Var(x_1 \cdot I_{1,1} + \overline{x}_1 \cdot I_{0,1}) = \frac{1}{4}(I_{1,1} - I_{0,1})^2 \tag{10.40}$$

Thus:

$$Var(I_{leak}) = \frac{1}{4}(I_{1,1} - I_{0,1})^2 + \frac{1}{4}(I_{1,2} - I_{0,2})^2 \tag{10.41}$$

And, finally, the PCC can be expressed as:

$$\rho = \frac{\frac{1}{4} \cdot (I_{1,1} - I_{0,1}) + \frac{1}{4} \cdot (I_{1,2} - I_{0,2})}{\sqrt{\frac{1}{4}(I_{1,1} - I_{0,1})^2 + \frac{1}{4}(I_{1,2} - I_{0,2})^2} \cdot \sqrt{\frac{1}{2}}} \tag{10.42}$$

We can now make use of the definition of ϵ. That is:

$$(I_{1,1} - I_{0,1}) = \epsilon_1$$

$$(I_{1,2} - I_{0,2}) = \epsilon_2 \tag{10.43}$$

And substitute equations 10.43 into 10.42:

$$\rho = \frac{\frac{1}{4} \cdot \epsilon_1 + \frac{1}{4} \cdot \epsilon_2}{\sqrt{\frac{1}{4}\epsilon_1^2 + \frac{1}{4}\epsilon_2^2} \cdot \sqrt{\frac{1}{2}}} \tag{10.44}$$

Finally, for an n-bit register array, we have:

$$\rho_{I_{leak},HW} = \frac{1}{\sqrt{n}} \cdot \frac{\epsilon_1 + \epsilon_2 + ... + \epsilon_n}{\sqrt{\epsilon_1^2 + \epsilon_2^2 + ... + \epsilon_n^2}} \tag{10.45}$$

Bibliography

[1] D. Justo, D. Cavalheiro, and F. Moll, "Body bias generators for ultra low voltage circuits in fdsoi technology," in *2017 32nd Conference on Design of Circuits and Integrated Systems (DCIS)*, 2017, pp. 1–6.

[2] R. Schaller, "Moore's law: past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[3] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology — CRYPTO '96*, N. Koblitz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 104–113.

[4] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1666, pp. 388–397, 1999.

[5] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, ser. WOST'99. USA: USENIX Association, 1999, p. 17.

[6] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[7] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[8] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *Information and Communications Security*, P. Ning, S. Qing, and N. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 529–545.

[9] M. Alioto, M. Poli, and S. Rocchi, "A general power model of differential power analysis attacks to static logic circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 711–724, 2010.

[10] S. Clerc, T. Di Gilio, and A. Cathelin, *"The Fourth Terminal"*, ser. Integrated Circuits and Systems, S. Clerc, T. Di Gilio, and A. Cathelin, Eds. Cham: Springer International Publishing, 2020. [Online]. Available: http://link.springer.com/10.1007/978-3-030-39496-7

[11] D. R. Stinson and M. B. Paterson, *Cryptography: Theory and practice*, 4th ed. Chapman & Hall/CRC, 2018.

[12] A. Kerckhoffs, "La cryptographie militaire," *Journal des Sciences Militaires*, pp. 161–191, 1883.

[13] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[14] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," in *Advances in Cryptology — EUROCRYPT '93*, T. Helleseth, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 386–397.

[15] C. D. Canniere and B. Preneel, "Trivium specifications," *eSTREAM, ECRYPT Stream Cipher Project*, vol. 2006, 2006.

[16] R. Bevan and E. Knudsen, "Ways to enhance differential power analysis," in *Information Security and Cryptology — ICISC 2002*, P. J. Lee and C. H. Lim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 327–342.

[17] Q. Luo and Y. Fei, "Algorithmic collision analysis for evaluating cryptographic systems and side-channel attacks," *2011 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2011*, pp. 75–80, 2011.

[18] Y. Fei, Q. Luo, and A. A. Ding, "A statistical model for dpa with novel algorithmic confusion analysis," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 233–250.

[19] Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A Statistics-based Fundamental Model for Side-channel Attack Analysis." *IACR Cryptology ePrint Archive*, vol. 2014, p. 152, 2014. [Online]. Available: https://eprint.iacr.org/2014/152

[20] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3156, pp. 16–29, 2004.

[21] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[22] C. Clavier, J. S. Coron, and N. Dabbous, "Differential power analysis in the presence of hardware countermeasures," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1965 LNCS, pp. 252–263, 2000.

[23] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1666, pp. 398–412, 1999.

[24] L. Goubin and J. Patarin, "DES and differential power analysis the "duplication" method," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1717, pp. 158–172, 1999.

[25] T. S. Messerges, "Using Second-Order Power Analysis to attack DPA resistant software," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1965 LNCS, pp. 238–251, 2000.

[26] M. Joye, P. Paillier, and B. Schoenmakers, "On second-order differential power analysis," *Lecture Notes in Computer Science*, vol. 3659, pp. 293–308, 2005.

[27] E. Prouff, M. Rivain, and R. Bévan, "Statistical analysis of second order differential power analysis," *IEEE Transactions on Computers*, vol. 58, no. 6, pp. 799–811, 2009.

[28] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proceedings of the 28th European Solid-State Circuits Conference*, 2002, pp. 403–406.

[29] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Improving the security of dual-rail circuits," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 282–297.

[30] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, 2004, pp. 246–251 Vol.1.

[31] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, J. R. Rao and B. Sunar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 172–186.

[32] K. Tiri and I. Verbauwhede, "Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology," in *Cryptographic Hardware and Embedded Systems - CHES 2003*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 125–136.

[33] A. Shamir, "Protecting smart cards from passive power analysis with detached power supplies," in *Cryptographic Hardware and Embedded Systems — CHES 2000*, Ç. K. Koç and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 71–77.

[34] W. Yu and S. Kose, "Time-Delayed Converter-Reshuffling: An Efficient and Secure Power Delivery Architecture," *IEEE Embedded Systems Letters*, vol. 7, no. 3, pp. 73–76, 2015.

[35] W. Yu, O. A. Uzun, and S. Köse, "Leveraging on-chIP voltage regulators as a countermeasure against side-channel attacks," *Proceedings - Design Automation Conference*, vol. 2015-July, 2015.

[36] W. Yu and S. Köse, "Charge-Withheld Converter-Reshuffling: A Countermeasure Against Power Analysis Attacks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 5, pp. 438–442, 2016.

[37] W. Yu and S. Köse, "Exploiting voltage regulators to enhance various power attack countermeasures," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 2, pp. 244–257, 2018.

[38] W. Yu and Y. Wen, "Leakage Power Analysis (LPA) Attack in Breakdown Mode and Countermeasure," *International System on Chip Conference*, vol. 2018-Septe, pp. 238–243, 2019.

[39] C. Tokunaga and D. Blaauw, "Secure aes engine with a local switched-capacitor current equalizer," in *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, 2009, pp. 64–65,65a.

[40] C. Tokunaga and D. Blaauw, "Securing encryption systems with a switched capacitor current equalizer," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 1, pp. 23–31, 2010.

[41] S. Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Y. Xie, "Power attack resistant cryptosystem design: A dynamic voltage and frequency switching approach," *Proceedings -Design, Automation and Test in Europe, DATE '05*, vol. 2005, pp. 64–69, 2005.

[42] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," in *1990 37th IEEE International Conference on Solid-State Circuits*, 1990, pp. 238–239.

[43] K. Baddam and M. Zwolinski, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure," *Proceedings of the IEEE International Conference on VLSI Design*, pp. 854–859, 2007.

[44] S. Ordas, M. Carbone, G. Ducharme, S. Tiran, and P. Maurine, "Efficiency of the rdvfs countermeasure," in *2014 IEEE Faible Tension Faible Consommation*, 2014, pp. 1–4.

[45] J. Giorgetti, G. Scotti, A. Simonetti, and A. Trifiletti, "Analysis of Data Dependence of Leakage Current in CMOS Cryptographic Hardware," in *Proceedings of the 17th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 78–83.

[46] Lang Lin and W. Burleson, "Leakage-based differential power analysis (LDPA) on sub-90nm CMOS cryptosystems," in *2008 IEEE International Symposium on Circuits and Systems*, 2008, pp. 252–255.

[47] M. Alioto, M. Poli, and S. Rocchi, "Power analysis attacks to cryptographic circuits: A comparative analysis of DPA and CPA," *Proceedings of the International Conference on Microelectronics, ICM*, pp. 333–336, 2008.

[48] M. Alioto, S. Bongiovanni, M. Djukanovic, G. Scotti, and A. Trifiletti, "Effectiveness of leakage power analysis attacks on dpa-resistant logic styles under process variations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 429–442, 2014.

[49] A. Moradi, "Side-channel leakage through static power," in *Cryptographic Hardware and Embedded Systems – CHES 2014*, L. Batina and M. Robshaw, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 562–579.

[50] S. M. Del Pozo, F. Standaert, D. Kamel, and A. Moradi, "Side-channel attacks from static power: When should we care?" in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 145–150.

[51] T. Moos, A. Moradi, and B. Richter, "Static power side-channel analysis—an investigation of measurement factors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 376–389, 2020.

[52] J. Waddle and D. Wagner, "Towards efficient second-order power analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–15.

[53] T. Moos, A. Moradi, and B. Richter, "Static power side-channel analysis of a threshold implementation prototype chip," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 1324–1329.

[54] W. Yu and S. Köse, "Security implications of simultaneous dynamic and leakage power analysis attacks on nanoscale cryptographic circuits," *Electronics Letters*, vol. 52, no. 6, pp. 466–468, 2016.

[55] N. Zhu, Y. Zhou, and H. Liu, "Employing Symmetric Dual-Rail Logic to Thwart LPA Attack," *IEEE Embedded Systems Letters*, vol. 5, no. 4, pp. 61–64, 2013.

[56] B. Fadaeinia, T. Moos, and A. Moradi, "BSPL: Balanced Static Power Logic," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 558, 2020.

[57] ——, "Balancing the leakage currents in nanometer cmos logic - a challenging goal," *Applied Sciences*, vol. 11, no. 15, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/15/7143

[58] Nianhao Zhu, Yujie Zhou, and Hongming Liu, "Counteracting leakage power analysis attack using random ring oscillators," in *PROCEEDINGS OF 2013 International Conference on Sensor Network Security Technology and Privacy Communication System*, 2013, pp. 74–77.

[59] W. Yu and S. Köse, "Security-Adaptive Voltage Conversion as a Lightweight Countermeasure Against LPA Attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, pp. 2183–2187, 2017.

[60] W. Yu and S. Köse, "False Key-Controlled Aggressive Voltage Scaling: A Countermeasure Against LPA Attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 12, pp. 2149–2153, 2017.

[61] Y. Wen and W. Yu, "Breaking LPA-resistant cryptographic circuits with principal component analysis," *Integration*, vol. 80, pp. 1–4, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926021000511

[62] J. Zhu, R. Martin, and J. Chen, "Punchthrough current for submicrometer mosfets in cmos vlsi," *IEEE Transactions on Electron Devices*, vol. 35, no. 2, pp. 145–151, 1988.

[63] I. De and C. M. Osburn, "Impact of super-steep-retrograde channel doping profiles on the performance of scaled devices," *IEEE Transactions on Electron Devices*, vol. 46, no. 8, pp. 1711–1717, 1999.

[64] A. Quelen, G. Pillonnet, P. Flatresse, and E. Beigné, "A 2.5μW 0.0067mm2 automatic back-biasing compensation unit achieving 50% leakage reduction in FDSOI 28nm over 0.35-to-1V VDD range," *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, vol. 61, pp. 304–306, 2018.

[65] M. Blagojevic, M. Cochet, B. Keller, P. Flatresse, A. Vladimirescu, and B. Nikolic, "A fast, flexible, positive and negative adaptive body-bias generator in 28nm FDSOI," *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, vol. 2016-Septe, pp. 9–10, 2016.

[66] T. Kuroda, T. Fujita, S. Mita, T. Nagamatu, S. Yoshioka, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, "A 0.9 V 150 MHz 10 mW 4 mm/sup 2/ 2-D discrete cosine transform core processor with variable-threshold-voltage scheme," in *1996 IEEE International Solid-State Circuits Conference. Digest of TEchnical Papers, ISSCC*, vol. 9200, no. 11. IEEE, 1996, pp. 166–167,. [Online]. Available: http://ieeexplore.ieee.org/document/488555/

[67] N. Kamae, A. K. Islam, A. Tsuchiya, and H. Onodera, "A body bias generator with wide supply-range down to threshold voltage for within-die variability compensation," *2014 IEEE Asian Solid-State Circuits Conference, A-SSCC - Proceedings of Technical Papers*, pp. 53–56, 2015.

[68] A. Siddiqi, N. Jain, and M. Rashed, "Back-bias generator for post-fabrication threshold voltage tuning applications in 22nm FD-SOI process," *Proceedings - International Symposium on Quality Electronic Design, ISQED*, vol. 2018-March, pp. 268–273, 2018.

[69] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, vol. 1, no. SUPPL., pp. 344–345+538, 2002.

[70] R. Giterman, O. Keren, and A. Fish, "Improving the Security of a 6T SRAM using Body-Biasing in 28 nm FD-SOI," in *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2018, pp. 1–2.

[71] National Institute of Standards and Technology, "Advanced encryption standard," *NIST FIPS PUB 197*, 2001.

[72] ——, "Advanced Encryption Standard," vol. 26, 2001.

[73] Y. Jia, Y. Hu, F. Wang, and H. Wang, "Correlation power analysis of Trivium," *Security and Communication Networks*, vol. 5, no. 5, pp. 479–484, may 2012. [Online]. Available: http://doi.wiley.com/10.1002/sec.329

[74] G. Villar-Piqué, H. J. Bergveld, and E. Alarcón, "Survey and benchmark of fully integrated switching power converters: Switched-capacitor versus inductive approach," *IEEE Transactions on Power Electronics*, vol. 28, no. 9, pp. 4156–4167, 2013.

[75] M. H. Eid and E. Rodriguez-Villegas, "Analysis and design of cross-coupled charge pump for low power on chip applications," *Microelectronics Journal*, vol. 66, pp. 9–17, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026269216307492

[76] K. Palma and F. Moll, "Linear, time-invariant model of the dynamics of a cmos cc-cp," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2019, pp. 726–729.

[77] B. Nowacki, N. Paulino, and J. Goes, "A simple 1 GHz non-overlapping two-phase clock generators for SC circuits," in *Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES 2013*, 2013, pp. 174–178.

[78] P. E. Allen, *CMOS analog circuit design*, int. 2nd ed. ed., ser. Oxford series in electrical and computer engineering. New York: Oxford University Press, 2010.

[79] T. Saito and S. Komatsu, "A low-voltage hysteresis comparator for low power applications," in *2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2017, pp. 427–430.

[80] P. R. Gray, *Analysis and Design of Analog Integrated Circuits*, 5th ed. Wiley Publishing, 2009.

[81] J. Vazquez and J. Pineda de Gyvez, "Built-in current sensor for ΔIDDQ testing," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 3, pp. 511–518, 2004.

[82] S. Henzler, *Time-to-Digital Converters*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[83] O. Uzun, "Speed, Power Efficiency, and Noise Improvements for Switched Capacitor Voltage Converters - PhD," no. June, 2017. [Online]. Available: https://digitalcommons.usf.edu/cgi/viewcontent.cgi?article=8167&context=etd

[84] B. B. Kormann, "High-Efficiency , Regulated Charge Pumps for High-Current Applications," pp. 1–15, 2003.

[85] S. N. Dhanuskodi, S. Keshavarz, and D. Holcomb, "Llpa: Logic state based leakage power analysis," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 218–223.

[86] D. Das, S. Maity, S. B. Nasir, S. Ghosh, A. Raychowdhury, and S. Sen, "High efficiency power side-channel attack immunity using noise injection in attenuated signature domain," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 62–67.

[87] Y. S. Hwang, R. L. Shih, P. H. Fu, Y. J. Hsiao, J. J. Chen, and C. C. Yu, "A compact fast-transient charge-pump boost converter using hysteretic compensated techniques," *2016 IEEE International Conference on Electron Devices and Solid-State Circuits, EDSSC 2016*, pp. 484–487, 2016.

[88] A. Moradi and A. Poschmann, "Lightweight cryptography and DPA countermeasures: A survey," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6054 LNCS, pp. 68–79, 2010.

[89] M. B. Capino, L. N. Rivera, J. A. Hora, and J. C. Pasco, "Closed-loop step-down fractional charge pump converter in 0.18um CMOS technology," *8th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2015*, no. December, pp. 1–10, 2016.

[90] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2007, pp. 286–296.

[91] L. Wang, C. Wu, L. Feng, A. Chang, and Y. Lian, "A Low-Power Forward and Reverse Body Bias Generator in CMOS 40 nm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1403–1407, 2018.

[92] E. Trichina, "Combinational logic design for aes subbyte transformation on masked data," 2003, not published elsewhere. e.v.trichina@samsung.com 12368 received 11 Nov 2003. [Online]. Available: http://eprint.iacr.org/2003/236

[93] P. Schaumont and K. Tiri, "Masking and dual-rail logic don't add up," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 95–106.

[94] K. S. Khouri and N. K. Jha, "Leakage power analysis and reduction during behavioral synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 6, pp. 876–885, 2002.

[95] Q. S. Phan, L. Bang, C. S. Pasareanu, P. Malacaria, and T. Bultan, "Synthesis of Adaptive Side-Channel Attacks," *Proceedings - IEEE Computer Security Foundations Symposium*, pp. 328–342, 2017.

[96] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti, "Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 2, pp. 355–367, 2010.

[97] C. H. Wann, K. Noda, T. Tanaka, M. Yoshida, and Chenming Hu, "A comparative study of advanced mosfet concepts," *IEEE Transactions on Electron Devices*, vol. 43, no. 10, pp. 1742–1753, 1996.

[98] S. R. Banna, P. C. H. Chan, P. K. Ko, C. T. Nguyen, and Mansun Chan, "Threshold voltage model for deep-submicrometer fully depleted soi mosfet's," *IEEE Transactions on Electron Devices*, vol. 42, no. 11, pp. 1949–1955, 1995.

[99] A. Agarwal, S. Mukhopadhyay, A. Raychowdhury, K. Roy, and C. H. Kim, "Leakage power analysis and reduction for nanoscale circuits," *IEEE Micro*, vol. 26, no. 2, pp. 68–80, 2006.

[100] F. X. Standaert, E. Peeters, G. Rouvroy, and J. J. Quisquater, "An overview of power analysis attacks against field programmable gate arrays," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 383–394, 2006.

[101] G. Goos, J. Hartmanis, and J. Leeuwen, *Lecture Notes in Computer Science*, 1999, vol. 1716.

[102] C. C. Hu, *Modern semiconductor devices for integrated circuits*. Prentice Hall, 2010.

[103] M. Bucci, L. Giancane, R. Luzzi, G. Scotti, and A. Trifiletti, "Delay-based dual-rail precharge logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 7, pp. 1147–1153, 2011.

[104] B. Halak, J. Murphy, and A. Yakovlev, "Power balanced circuits for leakage-power-attacks resilient design," *Proceedings of the 2015 Science and Information Conference, SAI 2015*, pp. 1178–1183, 2015.

[105] H. Lohrke, S. Tajik, T. Krachenfels, C. Boit, and J.-P. Seifert, "Key Extraction Using Thermal Laser Stimulation A Case Study on Xilinx Ultrascale FPGAs," *tCHES 2018*, vol. 2018, Issu, no. 3, pp. 573–595, 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7287

[106] H. Li, K. Wu, B. Peng, Y. Zhang, X. Zheng, and F. Yu, "Enhanced correlation power analysis attack on smart card," *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008*, pp. 2143–2148, 2008.

[107] Y. K. Ramadass and A. P. Chandrakasan, "Voltage scalable switched capacitor DC-DC converter for ultra-low-power on-chip applications," *PESC Record - IEEE Annual Power Electronics Specialists Conference*, pp. 2353–2359, 2007.

[108] M. Lipski and S. Gregori, "Analysis of Charge Reuse in Switched-Capacitor Power-Converter Drivers," *Midwest Symposium on Circuits and Systems*, vol. 2019-Augus, pp. 742–745, 2019.

[109] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti, "Leakage power analysis attacks: Well-defined procedure and first experimental results," *Proceedings of the International Conference on Microelectronics, ICM*, pp. 46–49, 2009.

[110] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[111] J. Y. Kim, Y. H. Jun, and B. S. Kong, "CMOS charge pump with transfer blocking technique for no reversion loss and relaxed clock timing restriction," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 1, pp. 11–15, 2009.

[112] S. Mangard, "Hardware countermeasures against DPA - A statistical analysis of their effectiveness," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2964, pp. 222–235, 2004.

[113] M. Safta, P. Svasta, M. Dima, A. Marghescu, and M. N. Costiuc, "Design and setup of Power Analysis attacks," *2016 IEEE 22nd International Symposium for Design and Technology in Electronic Packaging, SIITME 2016*, no. 978, pp. 110–113, 2016.

[114] M. Liu, *Demystifying Switched Capacitor Circuits*. Newnes, 2006.

[115] D. Kilani, B. Mohammad, H. Saleh, and M. Ismail, "Digital pulse frequency modulation for switched capacitor DC-DC converter on 65nm process," *2014 21st IEEE International Conference on Electronics, Circuits and Systems, ICECS 2014*, vol. 2, pp. 642–645, 2015.

[116] M. S. Makowski and A. Kushnerov, "Canonical switched capacitor converters. comments, complements, and refinements," in *2017 European Conference on Circuit Theory and Design (ECCTD)*, 2017, pp. 1–4.

[117] O. A. Uzun and S. Köse, "Converter-gating: A power efficient and secure on-chip power delivery system," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 2, pp. 169–179, 2014.

[118] N. Mentens, B. Gierlichs, and I. Verbauwhede, "Power and fault analysis resistance in hardware through dynamic reconfiguration," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5154 LNCS, pp. 346–362, 2008.

[119] G. Ghibaudo, "Electrical characterization of FDSOI CMOS devices," *European Solid-State Device Research Conference*, vol. 2016-Octob, pp. 135–141, 2016.

[120] H. W. Hwang, J. H. Chun, and K. W. Kwon, "A low power cross-coupled charge pump with charge recycling scheme," *3rd International Conference on Signals, Circuits and Systems, SCS 2009*, pp. 1–5, 2009.

[121] P. Corsonello, S. Perri, and M. Margala, "An integrated countermeasure against differential power analysis for secure smart-cards," in *2006 IEEE International Symposium on Circuits and Systems*, 2006, pp. 4 pp.–.

[122] R. Jevtic, M. Ylitolva, and L. Koskinen, "Reconfigurable Switched Capacitor DC-DC Converter for Improved Security in IoT Devices," *2018 IEEE 28th International Symposium on Power and Timing Modeling, Optimization and Simulation, PATMOS 2018*, pp. 243–247, 2018.

[123] M. M. Pelicia, R. P. Coimbra, and P. B. Zanetta, "A back biasing voltage generator for 28nm UTBB-FDSOI RVT CMOS digital circuits," *ICICDT 2018 - International Conference on IC Design and Technology, Proceedings*, pp. 1–4, 2018.

[124] M. S. Makowski and D. Maksimovic, "Performance limits of switched-capacitor dc-dc converters," in *Proceedings of PESC '95 - Power Electronics Specialist Conference*, vol. 2, 1995, pp. 1215–1221 vol.2.

[125] G. Villar-Piqué, H. J. Bergveld, and E. Alarcón, "Survey and benchmark of fully integrated switching power converters: Switched-capacitor versus inductive approach," *IEEE Transactions on Power Electronics*, vol. 28, no. 9, pp. 4156–4167, 2013.

[126] A. R. Kazmi, M. Afzal, M. F. Amjad, and A. Rashdi, "Attacks on stream ciphers," pp. 373–396, 2009.

[127] P. Socha, J. Brejník, and M. Bartík, "Attacking AES implementations using correlation power analysis on ZYBO Zynq-7000 SoC board," *2018 7th Mediterranean Conference on Embedded Computing, MECO 2018 - Including ECYPS 2018, Proceedings*, no. June, pp. 1–4, 2018.

[128] R. Li, X. Cui, W. Wei, D. Wu, K. Liao, N. Liao, K. Ma, D. Yu, and X. Cui, "A combined countermeasure against DPA and implementation on des," *2013 IEEE International Conference of Electron Devices and Solid-State Circuits, EDSSC 2013*, pp. 3–4, 2013.