



SAPIENZA
UNIVERSITÀ DI ROMA

Design techniques for secure cryptographic circuits in deep submicron technologies

Dipartimento di Ingegneria dell'Informazione, Elettronica e Telecomunicazioni

Dottorato di Ricerca in Ingegneria Elettronica – XXVII Ciclo

Candidate

Simone Bongiovanni

ID number 798246

Thesis Advisor

Prof. Alessandro Trifiletti

Co-Advisor

Prof. Giuseppe Scotti

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

2nd March 2015

Thesis defended on 2nd March 2015

in front of a Board of Examiners composed by:

Prof. Giovanni Busatto - Università di Cassino

Prof. Gian Carlo Cardarilli - Università di Roma Tor Vergata

Prof. Luigi Zeni - Seconda Università degli Studi di Napoli

Design techniques for secure cryptographic circuits in deep submicron technologies

Ph.D. thesis. Sapienza – University of Rome

ISBN: 000000000-0

© 2015 Simone Bongiovanni. All rights reserved

Author's email: bongiovanni@die.uniroma1.it

List of Figures

1.1	A schematic representation of the different possible methodologies to design digital VLSI circuits.	3
1.2	Block diagram of a standard digital design flow [119].	6
1.3	Description of a standard side-channel scenario [116].	10
1.4	Origin of the dynamic currents in a CMOS inverter when output switches from low to high (left) and from high to low (right).	14
1.5	Origin of the leakage currents in a CMOS inverter when input is low (left) and high (right) in steady state condition.	14
1.6	Block diagram describing the complete PAAs procedure [72].	20
1.7	A list of 24 block ciphers adoptable for lightweight applications.	25
1.8	Different elaboration steps of the encoding procedure of AES-128.	27
1.9	S-Box of the <i>SubstitutionBytes</i> layer of the AES encoder.	28
1.10	Different elaboration steps of the encoding procedure of Serpent.	29
1.11	S-Boxes of the Serpent encoder.	30
1.12	Different elaboration steps of the encoding procedure of PRESENT.	31
1.13	S-Box of the PRESENT encoder.	31
1.14	A model of the contribution of the load capacitance [124].	39
1.15	Performances and features of some selected DPLs styles.	41
1.16	A detailed description of a standard semi-custom design flow.	42
1.17	A detailed description of a secure semi-custom design flow for implementing a DPL circuit.	43
2.1	Two possible realistic scenarios for practical PAAs: an attacker can measure filtered (a) or non-filtered current traces (b).	48
2.2	Timing diagram of logic-0 (a), logic-1 (b) signal in RTZ logics	52
2.3	Timing diagram of logic-0 (a), logic-1 (b), invalid (c) signal in TELs	53
2.4	Cell template of a TEL gate composed of two independent circuit halves implementing the logic functions F_1 and F_2	55
2.5	Cell templates of basic combinational gates, compatible with the TEL encoding and designed with only AND logic gates.	56
2.6	Full custom cell template of a TEL-compatible inverter (iDDPL style) (left) and a RTZ inverter (SABL style) (right).	56
2.7	A pipelined circuit template in which the information is enclosed inside a time interval δ	60
2.8	Variation of the delay δ along combinational logics before and after a register.	62

2.9	Testbench for the simulations of the TEL inverter cell with an unbalanced load and a variable RC filter on the PSN.	63
2.10	Distribution of the current samples during the evaluation phase for the unbalanced TEL inverter, with a variable filtering capacitance.	63
2.11	PAAs scenario for a TEL circuit, with the insertion of an on-chip filter for removing the high frequencies components directly at layout level.	65
2.12	ΔFFT vector for the TEL inverter for different value of the mismatch factor ($\delta = 500ps$): MF = 1 (black), 2(red), 3 (green) and 4 (blue).	68
2.13	ΔFFT vector for the SABL inverter for different value of the mismatch factor: MF = 1 (black), 2(red), 3 (green) and 4 (blue).	68
2.14	Plot of the frequency f_0 as a function of δ	71
2.15	ΔFFT vector for the TEL inverter for $\delta = 100ps$	71
2.16	ΔFFT vector for the TEL inverter for $\delta = 5ns$	71
2.17	A generic bit slice hardware implementation of the Serpent encryption processor.	74
2.18	Data path of the SERPENT-block using a DPL implementation.	74
2.19	Timing diagram of in/out signals and latency of the processed words at each clock cycle of the pipeline.	75
2.20	Current traces for each of the 256 input combinations of the TEL circuit in the evaluation and postcharge phases of the third clock cycle.	77
2.21	FED vector for TEL circuit with low unbalance on the interconnect wires (black curve) and with a maximum unbalance ($MF = 3$) (red curve).	77
2.22	Equivalent circuit model for the testbench in Cadence simulations [59]	78
2.23	FED vector for the TEL circuit calculated after having filtered the current traces ($f_0 = 30MHz$).	78
2.24	Correlation coefficients plot of the 256 simulated traces of the TEL circuits as a function of time; correct key is indicated in bold black line.	80
2.25	Correlation coefficients plot of the 256 simulated traces of the SABL circuits as a function of time; correct key is indicated in bold black line.	80
2.26	PBC of the TEL circuit for the bits of the word at the output of the S-Box (correct key in bold).	82
2.27	PBC of the SABL circuit for the bits of the word at the output of the S-Box (correct key in bold).	82
2.28	Correlation coefficients plot as a function of the number of input plaintexts for the TEL circuit, in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.	84
2.29	Correlation coefficients plot as a function of time for the TEL circuit in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.	84
2.30	Correlation coefficients plot as a function of time for the SABL circuit in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.	86
2.31	Correlation coefficients plot as a function of time for the SABL circuit in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.	86

2.32	MTD as a function of the noise standard deviation.	86
2.33	Correlation coefficients plot as a function of the number of input plaintexts for the TEL circuit, in the case of $\sigma_{noise} \approx 10^{-3} < \sigma_{noise}^{CR}$; correct key is indicated in bold black line.	87
2.34	Correlation coefficients plot as a function of time for the TEL circuit in the case of $\sigma_{noise} \approx 10^{-3} < \sigma_{noise}^{CR}$; correct key is indicated in bold black line.	87
3.1	Cell template of an iDDPL BUFF/INV gate for the TEL circuits.	92
3.2	Timing diagram of the BUFF/INV cell in correspondence of signal (1,0) (logic-1 in CMOS domain)	92
3.3	iDDPL gate suffering on the memory effect (a) and iDDPL gate without memory effect with the presence of internal keeper transistors (b)	93
3.4	Cell template of an iDDPL AND/NAND logic gate.	95
3.5	Current traces of an iDDPL BUFF/INV (left) and an AND/NAND gate (right)	95
3.6	Circuit for the conversion of a CMOS signal into the TEL domain for the iDDPL style.	95
3.7	Implementation of a delay element line based on a current starved inverter for the DDPL flip-flop.	96
3.8	Dependence of the delay δ with the control voltage V_{bias}	97
3.9	A basic iDDPL AND/NAND gate with the equivalent circuits of the evaluation network.	102
3.10	Time diagram of the evaluated signals at the output of a basic DDPL AND/NAND cell for all possible inputs.	103
3.11	A basic iDDPL AND/NAND gate with the equivalent circuits of the evaluation network.	104
3.12	Time diagram of the evaluated signals at the output of the improved iDDPL AND/NAND cell for all possible inputs, when $t_2 > \tau_1$	105
3.13	Time diagram of the evaluated signals at the output of the improved iDDPL AND/NAND cell for all possible inputs, when $t_2 < \tau_1$	105
3.14	A basic DDPL XOR/NXOR gate with the equivalent circuits of the evaluation network.	107
3.15	Time diagram of the evaluated signals at the output of a basic DDPL XOR/NXOR cell for each input, for the case $t_2 > \tau_1$	107
3.16	Time diagram of the evaluated signals at the output of a basic DDPL XOR/NXOR cell for each input, for the case $t_2 < \tau_1$	107
3.17	Early evaluation free iDDPL AND/NAND (left), OR/NOR (middle) and XOR/NXOR gate (right) with minimum number of transistors.	108
3.18	Logic gates implementation of balanced iDDPL multiplexer (left) and full adder (right).	108
3.19	Waveforms of the differential signals at the output of a BUFF/INV gate for different MF (up) and current peaks in evaluation/postcharge (down).	109
3.20	Distribution of the output delay δ for the combinational gates after 500 iterations of Montecarlo simulations.	112

3.21	Superimposition of current traces for a basic (a) and an optimized (b) AND/NAND gate.	115
3.22	A combinational multi-level logic with 5 cascaded AND/NAND gates.	115
3.23	Superimposition of current traces for the multi level logic implemented with basic (a) and optimized (b) AND/NAND gates.	115
3.24	Superimposition of filtered current traces for the multi level logic implemented with basic (a) and optimized (b) AND/NAND gates.	115
3.25	Block scheme of the iDDPL master-slave flip-flop.	118
3.26	Timing diagram of the signal processed in the DDPL master-slave flip-flop.	118
3.27	Scheme of the input converter (left) and working principle (right).	120
3.28	Timing diagram of the signals elaborated by the input converter.	120
3.29	Scheme of the differential CMOS XOR gate which controls the transmission gates of the input converter.	120
3.30	A p-type master latch and working principle of a differential half circuit.	121
3.31	Timing diagram of the signals elaborated by the master latch.	121
3.32	A n-type slave latch and working principle of a differential half circuit.	123
3.33	Timing diagram of the signals elaborated by the slave latch.	123
3.34	Scheme of the improved input converter.	123
3.35	Superimposition of current traces for all possible input data in a CMOS (a), WDDL (b), SABL (c), TDPL (d), DDPL (e), and iDDPL (f) 4-bit register ($MF = 3$).	124
3.36	Coefficient of variation (CV) for a CMOS (a), WDDL (b), SABL (c), TDPL (d), DDPL (e), and iDDPL (f) 4-bit register as a function of the time samples in a clock cycle under different mismatch factors.	125
3.37	Screenshot of the Coefficient of Variation for the master-slave iDDPL 4-bit register around the delay time δ under different mismatch factors.	126
3.38	A comparison of the distribution of the average current for 100 clock cycles for the DPL flip-flops under test ($MF = 3$).	126
3.39	Variation of the delay δ along combinational logics.	129
3.40	Timing of the signals at the input/output of a generic combinational path.	131
3.41	Modelization of the routing of two standard cells: the differential wires in red must be low-leakage.	136
3.42	Layout of some combinational iDDPL cells: BUFF/INV (a), AND/NAND (b), OR/NOR (c), and XOR/NXOR (d).	136
3.43	Screenshot of the SERPENT-block after placement, with highlighted the differential interconnect wires between two iDDPL cells.	136
3.44	Symmetric layout of the iDDPL master-slave flip-flop.	138
3.45	A detailed description of a secure semi-custom design flow to implement a TEL circuit using iDDPL gates.	140
4.1	Macro-blocks composing the SERPAES chip: the AES-block and the SERPENT-block	142
4.2	Data path of a DPL-featured 4-bit unit implementing the first round of <i>Serpent</i> processor.	144

4.3	Schematic view of the S-Box S_0 from Serpent algorithm implemented only with combinational CMOS gates; the critical path is also indicated in figure.	144
4.4	Clock tree of the iDDPL core.	147
4.5	Deviation of the clock signals at the output of the buffers (pre-layout).	147
4.6	Polarization circuit to generate the static voltage V_{bias} in the iDDPL core.	150
4.7	Dependence of the delay δ with the control voltage V_{bias} , for $V_{DD} = 1.2V$	150
4.8	Waveforms of the input/output, clock and test signals of the iDDPL circuit before layout.	152
4.9	Variation of the tuning range curve for the different corners of the circuit.	154
4.10	Layout of the iDDPL core of the SERPENT-block.	157
4.11	Mismatch factor for the differential nets of the S-Box block of the iDDPL circuit: combinational gates (blue) and of flip-flops (red) output wires.	157
4.12	Waveforms of the input/output, clock and test signals of the iDDPL circuit after layout.	160
4.13	Deviation of the clock signals of the circuit due to the clock skew after layout.	161
4.14	Distribution of the value of δ on each differential combinational net of the iDDPL implementation of the S-Box for every input data.	161
4.15	Waveform of V_{bias} signal after post-layout simulations.	161
4.16	Layout of the SABL core of the SERPENT-block.	163
4.17	Complete layout of the SERPENT-block.	163
4.18	General architecture of the SERPENT-block.	164
4.19	LEF file of the SERPENT-block (screenshot).	165
4.20	Simplified schematic diagram of the AES-block of the SERPAES chip.	167
4.21	Block diagram and clock signal of the basic AES encryption unit (AES-0).	168
4.22	Block diagram and synchronization signals of AES-1.	168
4.23	Block diagram and synchronization signals of AES-2.	169
4.24	Block diagram and synchronization signals of AES-3.	170
4.25	Block diagram and synchronization signals of AES-4.	171
4.26	Block diagram of the basic synthesis flow in Synopsys DC [119].	172
4.27	Report of the area occupation of the AES-block	182
4.28	Time report of the critical path associated to the clock Sys_clk	183
4.29	Time report of the critical path associated to the clock $clk2$	184
4.30	Screenshot of the VHDL file: assignment of the pins of an input pad (left) and an output pad (right).	187
4.31	Report of the area occupation of the SERPAES chip.	191
4.32	Schematic view of the SERPAES chip after synthesis	193
4.33	Screenshot of the post-synthesis NCSim simulation.	193
4.34	Typical layout of a digital circuit, where the standard-cells are placed side-by-side.	194
4.35	Step-by-step design of the floorplan.	201
4.36	Final floorplan of SERPAES.	202

4.37	Description of the clock <i>clk2_AES</i> in the file <i>SERPAES_clock.ctstch</i>	208
4.38	Post place and route layout of the SERPAES chip in SoC Encounter.	216
4.39	Final layout of SERPAES for full-chip verification in Cadence Virtuoso.	216
4.40	Photograph of the SERPAES chip.	217
4.41	Photograph of the board designed to perform measurements on the SERPAES chip.	217
4.42	Photograph of the workstation in our laboratory for the evaluation of SERPAES.	218
4.43	Power trace from the core AES0 inside the SERPAES chip measured by the scope.	219
4.44	Correlation coefficient curves for one byte (15) of the word at the output of the SubstitutionBox layer of round 1 of AES-0 (490k plaintexts).	219
4.45	Output bit on the critical path of the iDDPL (left) and the SABL (right) blocks of the SERPAES chip.	219
4.46	Power trace from the iDDPL block inside the SERPAES chip (three cycles) measured by the scope.	220
4.47	Power trace from the SABL block inside the SERPAES chip (three cycles) measured by the scope.	220
5.1	Current samples as a function of the operating frequency: static measures are feasible in the case (a), whereas in (b) dynamic power is predominant.	224
5.2	Block diagram describing the LPA procedure [5].	226
5.3	Leakage current variability for CMOS gates.	230
5.4	Simulation testbench to measuring the leakage of DPLs.	231
5.5	Leakage current variability for WDDL gates.	231
5.6	Leakage current variability for MDPL gates.	232
5.7	Simulation testbench to measure the leakage of dynamic DPLs.	234
5.8	Leakage current distribution for SABL gates.	234
5.9	A generic circuit composed of two cascaded blocks: one bit-slice (registers) and one non-bit-slice (combinational).	236
5.10	One bit-slice of the Serpent encryption processor.	238
5.11	Data path of the first stage of a bit-slice (SERPENT-block) for DPLs.	238
5.12	Leakage current as a function of the Hamming weight of the input (dotted line) and the output (straight line) words of the S-Box.	240
5.13	PBCs between leakage and output bits of DPL S-Boxes.	240
5.14	PBCs for the DPLs implementation of SERPENT-block (correct key in bold red line)	243
5.15	Leakage current as a function of the selection function $w = f(IN, Key)$	243
5.16	Minimum number to disclosure as a function of the SNR.	245
5.17	Correlation coefficients as a function of the number of measurements for CMOS (the curve of the correct key is in red).	246
5.18	Correlation coefficients as a function of the number of measurements for WDDL (the curve of the correct key is in red).	246
5.19	Correlation coefficients as a function of the number of measurements for MDPL (the curve of the correct key is in red).	247

5.20	Correlation coefficients as a function of the number of measurements for SABL (the curve of the correct key is in red).	247
5.21	Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 CMOS sample circuits.	251
5.22	Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 WDDL sample circuits.	251
5.23	Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 MDPL sample circuits.	252
5.24	Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 SABL sample circuits.	252
5.25	Statistical distribution of the correlation coefficient under high intra-die variations (success rate < 1) (left) and moderate intra-die variations (success rate = 1) (right).	252
5.26	Distribution of the MTD for the successful LPA attacks against the CMOS (left upper), WDDL (right upper), MDPL (left lower), SABL (right lower) sample circuits.	253
5.27	Measure of static power on the current traces of a TEL circuit when the static voltage V_{bias} is unchanged (a) and changed (b).	254
5.28	Leakage current distribution for iDDPL gates.	255
5.29	Data path of the cryptographic circuit under test and clock signal diagram during static measures.	256
5.30	Circuit for the conversion of a CMOS signal into the TEL domain for the iDDPL style.	257
5.31	Correlation coefficients as a function of the number of measurements for DDPL (the curve of the correct key is in red).	259
5.32	Correlation coefficients as a function of the number of measurements for DDPL using the output of the XOR as selection function (the curve of the correct key is in red).	259
6.1	Block diagram of the measurement workstation for standard PAAs.	265
6.2	A photograph of the FPGA board used for the experiments.	265
6.3	Working principle of an inductive current probe.	268
6.4	Block diagram of the <i>SCLab</i> workstation in the laboratory.	269
6.5	Photograph of the <i>SCLab</i> workstation in the laboratory.	269
6.6	Current trace of the FPGA correctly displayed in Labview.	272
6.7	Screenshot of the GUI of Gemini for PAAs against an AES core.	272
6.8	Correlation coefficient curves displayed by Matlab for a single byte of the key (case of successful attack).	272
6.9	Block diagram and synchronization signals of AES-0.	273
6.10	Block diagram and synchronization signals of AES-3.	273
6.11	Block diagram and synchronization signals of AES-5.	273
6.12	Floorplan of AES-0 on the FPGA, in the case of synthesis with RAM cells (left) and without RAM cells (right).	274
6.13	Experiment 1: correlation coefficient curves for the 16 bytes of the word at the output of the <i>SubstitutionBox</i> layer of round 1 of AES-0 (100k plaintexts).	277

6.14	Experiment 1: correlation coefficient curves for the bytes 2, 5, 11 of the word at the output of the <i>SubstitutionBox</i> layer of round 1 of AES-0 (750k plaintexts).	278
6.15	Experiment 1: correlation coefficient curves for one byte of the word at the output of the <i>AddRoundKey</i> layer of round 1 of AES-0 (750k plaintexts).	278
6.16	Experiment 2: correlation coefficient curves for the 16 bytes of the word at the output of the <i>AddRoundKey</i> layer of round 1 of AES-3 (800k plaintexts).	279
6.17	Experiment 2: correlation coefficient curves for one byte of the word at the output of the <i>AddRoundKey</i> layer of round 0 of AES-0 (800k plaintexts).	280
6.18	Experiment 3: correlation coefficient curves for the 16 bytes of the word at the output of the <i>SubstitutionBox</i> layer of round 1 of AES-3 (1.2M plaintexts).	281
6.19	Experiment 3: correlation coefficient curves for the 16 bytes of the word at the output of the <i>AddRoundKey</i> layer of round 1 of AES-3 (1.2M plaintexts).	282
6.20	Experiment 4: correlation coefficient curves for one byte of the word at the output of the <i>SubstitutionBox</i> layer of round 1 of AES-3 (750k plaintexts).	283
6.21	Experiment 4: correlation coefficient curves for one byte of the word at the output of the <i>AddRoundKey</i> layer of round 0 of AES-3 (750k plaintexts).	283
6.22	Experiment 5: correlation coefficient curves for the 16 bytes of the word at the output of the <i>AddRoundKey</i> layer of round 0 of AES-5 (30k plaintexts).	284
6.23	Experiment 5: current trace of AES-5 during one elaboration.	285
6.24	Experiment 5: correlation coefficient curves for one byte of the word at the output of the <i>SubstitutionBox</i> layer of round 1 of AES-5 (30k plaintexts).	285
6.25	Experiment 6: correlation coefficient curves for one byte of the word at the output of the <i>SubstitutionByte</i> layer of round 1 of AES-5 (1M plaintexts).	285
6.26	Experiment 6: correlation coefficient curves for one byte of the word at the output of the <i>AddRoundKey</i> layer of round 0 of AES-5 for 100k, 200k and 1M plaintexts.	285
6.27	Experiment 6: current trace of AES-5 during the elaboration for a clock frequency equal to 1MHz (left) and 100kHz (right).	286
6.28	Experiment 7: drift effect of the current traces after an increasing number of elaborations for AES-5, when the probe TCP202 is used.	288
6.29	Experiment 7: correlation coefficient curves for one byte of the word at the output of the <i>AddRoundKey</i> layer of round 0 of AES-5 with 100k, 300k and 1M plaintexts, using the probe TCP202.	288
6.30	Experiment 8: correlation coefficient curves for one byte of the word at the output of the <i>AddRoundKey</i> layer of round 0 of AES-5 with 300k plaintexts, using the probe TCP202.	288

List of Tables

1.1	Output transition of a CMOS inverter and corresponding measured power consumption on the V_{DD} pin.	15
1.2	A classification of the most common hardware countermeasures according to the domain and the level of abstraction [72].	34
2.1	Description of the Return-to-Zero data encoding.	52
2.2	Description of the Time Enclosed Logic data encoding.	53
2.3	Model of the power consumption for a TEL and a SABL inverter cell with an unbalanced load.	57
2.4	Roadmap of the estimated propagation times for different submicron technologies.	60
2.5	NED as a function of the capacitance C_F	64
2.6	Power model at each clock cycle: the key dependence of the instantaneous power consumption can be detected in the 2 nd and the 3 rd clock cycles.	76
3.1	Delta as a function of the input data in presence of different MF for a BUFF/INV gate.	110
3.2	Delta as a function of the input data in presence of different MF for a AND/NAND gate.	110
3.3	Delta as a function of the input data in presence of different MF for a OR/NOR gate.	110
3.4	Delta as a function of the input data in presence of different MF for a XOR/NXOR gate.	110
3.5	Variation of the delay δ with the driving strength for different mismatch factors.	111
3.6	Output delay in the AND/NAND gates (in ps).	113
3.7	Power balancing metrics for the registers under test.	127
3.8	Performances of the DPLs in terms of occupied area and average energy under high output mismatch ($MF = 3$).	128
3.9	The DDPL065 cell library, designed using the CMOS065 technology.	139
4.1	Performances of the designed SABL and iDDPL circuits compared to CMOS.	162
4.2	Gate count for each core of the SERPAES chip after synthesis.	190

5.1	Correlation between static power of the S-Box and the Hamming weight of the input/output words.	239
5.2	Leakage currents measured for the DPL crypto-cores as a function of the input pattern.	242
5.3	Actual security metrics for the DPL crypto-cores (SNR = -20dB).	246
5.4	Coefficient of variation (%) for the static currents distribution measured for the 100 crypto-core sample circuits designed with DPLs.	251
5.5	Actual security metrics for the DPL crypto-cores successfully attacked with a LPA procedure (SNR = -20dB).	252
5.6	Leakage currents measured for the iDDPL crypto-core as a function of the output pattern.	256
6.1	Summary of the results of the experiments conducted on the AES cores implemented on the Cyclone FPGA in the workstation <i>SCLab</i>	287
A.1	Boolean equations describing the 4x4 S-Box S_0 of Serpent and the 4x4 S-Box of PRESENT.	295

List of Acronyms and Abbreviations

ADLBL	Asynchronous Directional Latch Based Logic
AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
BCDL	Balanced Cell-based Differential Logic
CAD	Computer Aided Design
CCWSR	Compile Characterize Write Script Recompile
CHES	Cryptographic Hardware and Embedded Systems
CMOS	Complementary Metal Oxide Semiconductor
CPA	Correlation Power Analysis
CPLD	Complex Programmable Logic Device
CPPR	Clock Path Pessimism Removal
CQFP	Ceramic Quad Flatpack
CRPR	Clock Reconvergence Pessimism Removal
CTS	Clock Tree Synthesis
CV	Coefficient of Variation
DB	Database
DC	Design Compiler
DDPL	Delay-based Dual-rail Precharge Logic
DES	Data Encryption Standard
DLL	Delay Locked Loop
DPA	Differential Power Analysis
DRC	Design Rule Check
DRSL	Differential Random Switching Logic
DTS	Dynamic Timing Simulation
DUA	Device Under test
EDA	Electronic Design Automation
EEPROM	Electrically Erasable Programmable Read Only Memory
EMA	Electromagnetic Analysis
FED	Frequency Energy Deviation
FFT	Fast Fourier Transform
FN	Feistel Network
FPGA	Field Programmable Gate Array
FUB	Fondazione Ugo Bordonni
GE	Gate Equivalents
GIDL	Gate Induced Drain Leakage
GUI	Graphic User Interface

HD	Hamming Distance
HDL	High Description Language
HVT	High Voltage Threshold
HVTGP	High Voltage Threshold General Purpose
HW	Hamming Weight
iDDPL	improved Delay-based Dual-rail Precharge Logic
IO	Input Output
ISCOM	Istituto Superiore delle Comunicazioni e dell'Informazione
LEF	Library Exchange Format
LFSR	Linear Feedback Shift Register
LIB	Liberty
LPA	Leakage Power Analysis
LUT	Look Up Table
LVS	Layout Versus Schematic
LVT	Low Voltage Threshold
MDPL	Masked Dual-rail Precharge Logic
MF	Mismatch Factor
MIA	Mutual Information Analysis
MTD	Measurements To Disclose
NED	Normalized Energy Deviation
NSD	Normalized Standard Deviation
PAA	Power Analysis Attack
PBC	Point Biserial Correlation
PC	Personal Computer
PD	Pull Down
PDA	Personal Digital Assistant
PLL	Phase Locked Loop
PSD	Power Spectrum Density
PSN	Power Supply Network
PU	Pull Up
RAM	Random Access Memory
RFID	Radio-Frequency Identifier
RNG	Random Number Generator
ROM	Read Only Memory
RTL	Register Transfer Logic

RTZ	Return To Zero
SABL	Sense Amplifier Based Logic
SAFF	Sense Amplifier Flip Flop
SCA	Side Channel Attack
SCLab	Smart Card Laboratory
SDB	Synopsys Database
SDC	Synopsys Design Constraints
SDF	Standard Delay Format
SIMD	Single Instruction Multiple Data
SNR	Signal to Noise Ratio
SoC	System on Chip
SPA	Simple Power Analysis
SPN	Substitution Permutation Network
SR	Single Rail or Set Reset or Success Rate
STA	Static Timing Analysis
SVT	Standard Voltage Threshold
TA	Template Attack
TDPL	Three-phases Dual-rail Precharge Logic
TEL	Time Enclosed Logic
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
WDDL	Wave Differential Dynamic Logic

Introduction

Scientific context

The main features of modern cryptography are entangled in these words [40]:

"A security system is as strong as its weakest part".

Every cryptographic design should be conceived with these words in mind. In general, a security system is composed of several parts, and the property of the weakest element fixes the constraints of the entire system. In order to guarantee a minimum level of security, identifying the weakest element(s) and improving its (their) strength are crucial issues.

By the end of 1970s when the *Diffie-Hellman exchange protocol* was proposed [80] and the public key cryptography was introduced, the main challenge of cryptographers was to develop new algorithms to encrypt secret data in the most secure and effective way. In the same years, the *Data Encryption Standard (DES)* was invented [35]. With the standardization of DES, symmetric key cryptography started to become established. DES was the standard for several years, until its mathematical strength was demonstrated to be low for modern computational machines. By the end of 1990s, a second generation of cryptography was developed: efficiency and speed were now considered as fundamental as security. The *Advanced Encryption Standard (AES)* is the supreme outcome of that period [2]. AES links together the benefits of a secure encoding with high performances, without sacrificing elegance, therefore it is widely considered as the best block cipher ever invented. Even if AES represents an efficient solution still nowadays, in very last years with the technology progress cryptography has taken a new direction, leading to the beginning of a novel era: increasingly, everyday items are enhanced to pervasive devices by embedding computing power. Smart-cards, RFID tags, medical devices, transport and payment cards, car immobilizers, PDA's are common examples. The interconnection of these pervasive devices leads to the famous vision of ubiquitous computing (*ubicomp*) introduced by Mark Weiser [128]. A widely shared view is that ubiquitous computing is the next paradigm in information technology, thus encapsulating information in ever more cheap, lightweight, and easy to carry out devices has become a trend. If early cryptographers thought that the strength and the effectiveness of a cryptographic application could be reached by designing a robust mathematical algorithm, in modern cryptography the perfection associated to the mathematics is leaving ever more space to the imperfect pragmatism of physical implementations.

Nowadays, mathematical design has evolved to such an extent that the weakest

element of the system is not anymore the cryptographic algorithm, but the implementation which runs the algorithm itself. This had the effect of changing the vision of security, such that today the physical implementation is considered as the weakest element of the system: in other words, an ideally secure cryptographic algorithm can be insecure in practical applications because of the vulnerability of the host device. The latter is an electronic device which maps the message and an internally stored key into a ciphertext. During the encryption, data are physically manipulated within the device in form of electric signals, bringing part of information to leak out during the computation through additional unpredicted information channels which fluctuate in a hostile environment and can be captured by an adversary. These information channels are widely known as *side channels*. Noise, light, consumed power, electromagnetic emissions, and any other perturbation are examples of side channels [112]. The dangerousness of these unintentional physical emissions is in the fact that they can be exploited by the adversary to extract secret information from the device. An attempt to steal information by a cryptographic device is called in the context of cryptography as *Side Channel Attack (SCA)* [61].

In last years, limiting the amount of information leaked by side channels has become a preeminent issue in the design of cryptographic circuits. Very often, the presence of side channels is investigated and possibly prevented through critical and penetrating analysis. Several research groups have started to investigate algorithmic and physical countermeasures to these threats, as well as new possible attack methodologies in order to demonstrate the effectiveness of an implementation or the vulnerability of another one. In this context, the technology progress plays a crucial role, given that several implementations, formerly defined secure, were proved to be highly unsafe, with a negative commercial and economical impact for the manufacturing companies. In some sense, it is just like an arms race between hackers and manufacturers, as well as researchers and researchers, in a sort of everyone against everyone. Clearly, research in this field is of primary interest for all the actors.

Objective and motivation to this work

In accordance to the above discussion, this thesis work has been conceived in the scenario of *physical observable cryptography* [82]. Physical observable cryptography is a relative recent discipline dealing with the problems of *physical security*, which were not considered by traditional cryptography [115]. It links together the issues related to the design of digital VLSI circuits with security aspects, under the perspective of providing a final product which exhibits a relatively low physical leakage. According to this definition, physical observable cryptography is intended with the purpose of building secure and efficient cryptographic devices, by integrating a specific cryptographic algorithm with a series of circuit improvements in order to ensure a sufficient level of security of commercial devices. Although no general theory has been still rigorously formulated, a lot of sub-fields have been developed, ranging from the design of digital cryptographic circuits, up to the definition of advanced statistical tools for the exploitation of the information leakage.

Therefore, physical observable cryptography covers a wide range of different and

heterogeneous disciplines, and involves the collaboration of experts of different areas: on a side, digital designers have the role of investigating every possible origins of physical leakage in modern technologies; on the other side, security experts must be able to integrate statistical and mathematical metrics for the evaluation of the leakage already during the earlier steps of the design, in order to provide the digital experts with efficient tools to define the design constraints. The question arises when different technologies are used to implement a cryptographic primitive: in other words, the level of security of a specific implementation designed some years ago can be dramatically different from the level of security of the same implementation designed today using a different technology. Furthermore, an implementation evaluated and defined as secure some years ago, can be cracked today with more powerful and advanced attack instruments.

Actually, it is impossible to face all the issues related to physical observable cryptography. For this reason, in this thesis work we focus on a specific aspect: the analysis and the design of cryptographic integrated circuits for portable embedded devices (e.g. smart-cards), designed with modern electronic nanotechnologies, which are able to counteract SCAs through an improved hardware security. More specifically we discuss and define novel methodologies and design-time metrics to build circuits able to thwart a specific subset of SCAs, *Power Analysis Attacks (PAAs)*, which are a class of attacks targeting the power consumption of a device as side channel [62].

In this context, this work has been addressed towards the design of hardware countermeasures against PAAs, which have the purpose of improving the level of security of a circuit already at physical level. A key problem in the design of cryptographic circuits in modern technologies is linked to the unavoidable electrical mismatches arising in submicron region, which are the physical reasons that make the instantaneous power consumption of a CMOS circuit dependent on the internal processed data. The electrical mismatches are expected to increase with the technology scaling, and require to re-define the level of security of previously proposed countermeasures as well as adequately address the strength of novel countermeasures. In this regard, capacitive and timing mismatches on the internal interconnect wires represent of course the most important leakage sources. In literature many efforts have been made towards the design of well-balanced interconnections in cryptographic circuits in order to mitigate these effects and equalize the power consumption. For this purpose several layout optimizations and assisted routing methods have been presented, but these techniques have the disadvantage of requiring a customized design flow; furthermore, they proved to be suboptimal, because process mismatches are very difficult to be controlled; moreover, the electrical mismatches are expected to increase with the technology scaling.

In addition to the problem of the electrical mismatches, recent research in the field of PAAs demonstrated that also the static power consumption, which was not considered in early works on PAAs, can be adopted as actual leakage source in submicron circuits. Leakage power in CMOS devices is expected to increase with the technology scaling, posing new requirements not only on the design of low-leakage electronic circuits, but also on the assessment of the level of security of nanoscaled crypto-devices.

Description of the adopted tools

The main part of the work presented in this thesis has been performed in simulation, using some of the most known CAD tools widely adopted for the design of electronic circuits. The simulation results and the discussion presented in this thesis prove the relevance of electronic simulations in the context of security design.

The most important outcome of this work has been the design of a cryptographic ASIC in CMOS 65nm using a semi-custom design flow in simulation. The chip contains a number of cryptographic cores (e.g. AES and Serpent) implementing some countermeasures developed in our department in last years, which have been extensively evaluated in simulations.

Most of the simulations presented in this work have been executed in the *Cadence Virtuoso Analog Design Environment* on Linux Fedora operating system. Layout, physical verification, extraction and back-annotation of the critical parts have been executed with the *Virtuoso Layout Editor Suite*, using the integrated *Calibre* processor for parasitics extraction.

The digital design flow for the cryptographic ASIC has been executed using *Synopsys DC Compiler* for Register Transfer Level (RTL) synthesis (front-end flow) and *Cadence SoC Encounter* for the place and route (back-end flow). Furthermore, functional simulations in the early steps of the project have been performed using *Altera Quartus II*, and repeated in post-layout using *Cadence NCSim*.

In addition, *Matlab* has been used for post-processing and calculation, and *Labview* has been adopted to control the oscilloscopes in practical activities.

Description of the technology process node

The electrical scheme as well as the layout presented in this thesis work have been designed using the 65nm Bulk CMOS (CMOS065) Process provided by *ST Microelectronics*. In this paragraph we provide some general information about the technology and the transistor models used in this work [117].

The design kit provided by the semiconductor vendor allows to choose the standard or the advanced process. We have used the standard option, which is composed of two possible alternatives: the Low Power (CMOS065_LP) and the Low Power with access to General Purpose (CMOS065) transistors models. The CMOS065_LP process serves battery operated and wireless applications; it is characterized by a single IO oxide plus a single core oxide dual V_t process, and uses copper metallization with 7 metal layers (5 thin plus 2 thick) and low-k dielectrics. The CMOS065_LP GP is a single IO oxide plus dual core oxide dual V_t process, and uses copper metallization with 7 metal layers (5 thin plus 2 thick) and low-k dielectrics.

Both the above described processes give access to the following add-on options: the Low V_t (LVT), the Standard V_t (SVT), and the High V_t (HVT) transistors. They differ from the value of the threshold voltage. The LVT devices switch faster, but have the penalty of exhibiting higher static leakage power. On the contrary, the HVT devices switch slower, but have the advantage of reducing the static leakage power. The SVT devices represent an efficient compromise for general applications.

The power supply of the transistors differ according to the processes: the

CMOS065_LP process has a 1.8nm thin oxide, is designed for 1.2V($\pm 10\%$) applications (max 1.32V) with 1.8V or 2.5V capable IO pads; the CMOS065_LPGP process has a 1.3nm thin oxide, is designed for 1.2V($\pm 10\%$) applications (max 1.32V) with GP devices limited to 1.0V($\pm 10\%$) (max 1.1V) and with 1.8V($\pm 10\%$) or 2.5V($\pm 10\%$) capable IO pads. The 1.8V IO pads have a 2.8nm thick oxide, and are targeted for a maximum voltage of 1.98, whereas the 2.5V IO pads have a 5.0nm thick oxide, support an overdrive supply voltage of 3.3V, and are target for a maximum voltage of 3.63V.

The natural drift devices are targeted for nominal power supply voltage of 8V on the drain ($|V_{ds}| \leq 8V$), and, accounting for a tolerance of ($\pm 10\%$), they are qualified for a maximum voltage of 8.8V on the drain.

Outline of the thesis

This thesis is divided in 7 chapters, which deal with electronics and cryptography issues and are strongly related.

In Chapter 1, an in-depth discussion about the topic of hardware security is provided. More specifically, we introduce the reader to the concept of physical security in nanoscaled cryptographic circuits. For this purpose, after discussing some basic foundations related to the design of integrated circuits for specific applications (ASICs) in deep-submicron technologies, the theory on SCAs is recalled, with a particular emphasis on PAAs. The analysis of PAAs and the design of countermeasures against PAAs is the guideline of this work. In this chapter we discuss also a basic standard procedure to design secure cryptographic circuits for hardware-oriented cryptographic primitives.

In Chapter 2 we present the *Time Enclosed Logic (TEL)* circuits as a hardware-level countermeasure to counteract PAAs in submicron circuits. This countermeasure is based on a novel digital data encoding and a specific design methodology, which can be integrated using common CAD tools for SPICE-level simulations and layout. Furthermore, we present a new design-time metric to assess the intensity of the leakage emission of cryptographic circuits. A case study cryptographic circuit, named in this thesis *SERPENT-block*, has been implemented using this countermeasure, and the PAAs resistance of such an implementation has been validated through extensive SPICE simulations.

In Chapter 3 we describe a full custom implementation of the TEL concept: the *improved Delay-based Dual-rail Precharge Logic (iDDPL)* style. A prototype standard-cell library of iDDPL gates has been used to design the SERPENT-block. More specifically, the logic gates composing the library are based on a specific circuit template and have been optimized down to the layout level by overcoming the electrical mismatches arising in submicron devices, which are well known to reduce the level of security of cryptographic circuits. In this chapter a discussion about the adoption of this prototype digital library in a standard semi-custom design flow is also provided.

In Chapter 4 we present the design of a prototype cryptographic ASIC, the *SERPAES chip*. This chapter is more digital design oriented, and provides an accurate description of the design of the ASIC, executed using standard industrial

CAD tools. Different independent cryptographic cores developed in our department and previously evaluated with respect to PAAs resistance are implemented on the chip. Among them, the SERPENT-block evaluated in Chapter 2 and implemented with the standard-cell library presented in Chapter 3 is described in minute detail. All the design steps as well as the design choices are discussed and described in order to understand the main issues of the project.

In Chapter 5 we recall the topic of *Leakage Power Analysis (LPA)* attacks. LPA attacks exploit the leakage arising from the static power consumption of deep-submicron circuits, which is correlated to the processed data and is expected to further increase with the technology scaling. We demonstrate on some specific case studies the feasibility of this new class of attacks and their dangerousness for embedded devices. As additional contribution, we investigate the LPA resistance of the countermeasure presented in previous chapters, and we demonstrate that it represents a promising solution also to prevent LPA attacks.

Chapter 6 is dedicated to the description of some experimental activities executed as an integration of this research. We have implemented two measurement workstations for implementing PAAs. The first setup has been mounted in order to evaluate the security of the SERPAES chip designed during the research activity. The measurement workstation is located in the laboratory of our department, where the SERPAES board on which the chip is hosted has been connected. At the moment of writing this work, the SERPAES chip has been proved to be fully functional, whereas the PAAs experiments are going to be performed. The second setup has been designed in the laboratories of the Ministry of Economic Development, where other countermeasures designed in our department have been evaluated on a FPGA.

In Chapter 7 we draw the conclusions to this work and discuss some possible developments as future improvement to these activities.

Contents

Introduction	xvii
1 Physical security in submicron technologies	1
1.1 Introduction	1
1.2 Foundations on ASIC design	2
1.2.1 Digital design flow strategies for VLSI circuits	2
1.2.2 ASIC vs FPGA	3
1.2.3 Description of the semi-custom design flow for ASIC	4
1.2.4 Main issues in the design of submicron integrated circuits	5
1.3 Physical security of cryptographic circuits: a review of Side-Channel Attacks (SCAs)	8
1.4 Origin of leakage in submicron technologies	10
1.4.1 Power consumption of CMOS circuits	10
1.4.2 Electromagnetic field irradiated by CMOS circuits	15
1.5 Power Analysis Attacks (PAAs)	17
1.5.1 Standard methodology to implement PAAs	17
1.5.2 Differential Power Analysis (DPA)	21
1.5.3 Correlation Power Analysis (CPA)	21
1.5.4 Template Attacks (TA)	22
1.6 Symmetric cryptography for SCAs evaluation	23
1.6.1 Block ciphers as cryptographic case study	23
1.6.2 Review of two basic block ciphers: Rijndael and Serpent	24
1.6.3 Lightweight cryptography: the PRESENT block cipher	29
1.7 Design strategies for secure block ciphers	31
1.7.1 Hardware properties of cryptographic primitives	31
1.7.2 Building blocks of cryptographic circuits	32
1.7.3 Hardware countermeasures against PAAs: a survey	33
1.7.4 Metrics to compare the efficiency of hardware implementations	35
1.8 Overview of Dual-Rail Precharge Logic (DPL) styles to counteract PAAs	36
1.8.1 General description of DPLs	36
1.8.2 Limitations of DPL styles in submicron technologies	37
1.8.3 A comparison among some popular DPLs	40
1.8.4 The secure digital design flow for cryptographic ASIC based on DPLs	40
1.9 Conclusion	42

2	Time Enclosed Logic: a hardware countermeasure to overcome PAAs	45
2.1	Introduction	45
2.2	Security assessment of hardware countermeasures	46
2.2.1	Assumptions on the adversary model adopted for the PAAs procedure	46
2.2.2	Metrics for the evaluation of the physical leakage	47
2.3	A novel data encoding for differential dynamic cryptographic circuits	50
2.3.1	Brief review of Return To Zero (RTZ) logics	50
2.3.2	Basic principle of Time Enclosed Logic (TEL) circuits	51
2.4	Description of TEL circuits	53
2.4.1	Cell templates for TEL circuit implementations	53
2.4.2	A first-order model of the dynamic power consumption	55
2.4.3	Timing constraints of a TEL circuit	58
2.4.4	The fluctuation effect of the delay δ	60
2.4.5	Second order effects: transient leakage	61
2.4.6	Energy balancing and timing enclosing properties	64
2.5	A balancing act: frequency analysis of the current traces	65
2.5.1	Insertion of an on chip filter in a TEL circuit	65
2.5.2	A new frequency-based metric	66
2.5.3	Relation between δ and f_0 in a TEL gate	69
2.6	A cryptographic case study	70
2.6.1	The SERPENT-block	70
2.6.2	Description of the architecture of the circuit	72
2.6.3	Direct analysis of the power model of the pipeline	73
2.6.4	Estimation of the cutoff frequency f_0 of the circuit	76
2.7	Security evaluation of the TEL circuit	77
2.7.1	Design of the on-chip filter considering chip peripherals	78
2.7.2	Area estimation of the countermeasure	79
2.7.3	Evaluation of the leakage of the noise-free current traces	79
2.7.4	Correlation Power Analysis attacks with Gaussian noise	81
2.8	Conclusions	85
3	An ASIC-oriented implementation of TEL circuits	89
3.1	Introduction	89
3.2	The Improved Delay-based Dual-rail Precharge Logic family	90
3.2.1	A full custom circuit implementation of TEL	90
3.2.2	Limitations of the Delay-based Dual-rail Precharge Logic (DDPL) style	91
3.2.3	Cell template of an iDDPL gate	91
3.2.4	Conversion of the signal from the CMOS domain	94
3.2.5	Design-time metrics and simulation parameters	97
3.3	Design and characterization of iDDPL combinational gates	99
3.3.1	Fluctuation effect in iDDPL gates	99
3.3.2	Analysis of the early evaluation effect	100
3.3.3	Combinational gates with capacitive load imbalance	108
3.3.4	Presence of mismatch variations	111

3.3.5	Validation of the model in some case study combinational gates	112
3.4	Design and characterization of an iDDPL sequential element	116
3.4.1	Main issues in the design of DPL sequential elements	116
3.4.2	A master slave flip-flop for the iDDPL style	118
3.4.3	Simulation and comparison of some DPL 4-bit registers	122
3.5	The prototype iDDPL library	128
3.5.1	Architecture of a micropipelined iDDPL circuit	128
3.5.2	Analysis of the timing constraints	129
3.5.3	Layout of the iDDPL cells	134
3.5.4	Consideration on the layout of the flip-flop	135
3.5.5	The DDPL065 cell library	138
3.6	Conclusions	139
4	Design of the SERPAES prototype chip	141
4.1	Introduction	141
4.2	Design of the SERPENT-block	142
4.2.1	Full custom design methodology	142
4.2.2	Data-path of the circuit	143
4.2.3	Design of the iDDPL sub-block	144
4.2.4	Design of the SABL sub-block	159
4.2.5	Design of the complete SERPENT-block	162
4.3	Description of the AES-block	164
4.3.1	From FPGA to ASIC	164
4.3.2	Architecture of the AES-block	166
4.3.3	The basic AES encryption unit (AES-0)	167
4.3.4	The random precharged interleaved pipeline countermeasure (AES-1)	168
4.3.5	The random interleaved pipeline countermeasure (AES-2)	169
4.3.6	The XOR-series countermeasure (AES-3)	169
4.3.7	The XOR-parallel countermeasure (AES-4)	170
4.4	Logic synthesis of SERPAES	170
4.4.1	General information on the synthesis methods in DC	171
4.4.2	Synthesis design flow for SERPAES	173
4.4.3	Preliminary steps before the synthesis	174
4.4.4	Logic synthesis of the AES-block	178
4.4.5	Logic synthesis of the SERPAES chip	182
4.5	Place and route of SERPAES	192
4.5.1	Standard-cell layout	192
4.5.2	Digital back-end flow	192
4.5.3	General settings	194
4.5.4	Floorplan	195
4.5.5	Static Timing Analysis	200
4.5.6	Placement	204
4.5.7	Post-placement optimization	206
4.5.8	Clock Tree Synthesis (CTS)	207
4.5.9	Post-CTS optimization	209
4.5.10	Routing	210

4.5.11	Post-route optimization	211
4.5.12	Signoff and final verification	213
4.6	Testing the functionality of SERPAES	215
4.6.1	Design of the SERPAES board	215
4.6.2	First measurements on the chip	215
4.7	Conclusions	218
5	Leakage Power Analysis attacks against nanoscaled DPL circuits	221
5.1	Introduction	221
5.2	Review of Leakage Power Analysis attacks	223
5.2.1	Context of LPA attacks	223
5.2.2	Leakage model for bit-slice structures	225
5.2.3	Standard procedure for LPA attacks on bit-slice circuits	225
5.2.4	Some considerations about noise in practical LPA attacks	226
5.2.5	Security metrics to assess the vulnerability on the static power	227
5.3	Leakage current in combinational gates in CMOS 65nm	229
5.3.1	Evaluation of the variability of the leakage of single logic gates	229
5.3.2	Standard CMOS logic	229
5.3.3	Dual-rail Precharge Logic styles	230
5.4	Leakage Power Analysis attacks and lightweight cryptography	233
5.4.1	Leakage model for a generic non-bit-slice structure	233
5.4.2	Static power analysis methodology for a lightweight cryptographic implementation	235
5.4.3	Analysis of the static power of the combinational S-Box S_0	237
5.4.4	Analysis of the static power of the SERPENT-block	241
5.4.5	LPA attacks against the crypto-processor considering noise	242
5.5	Evaluation of the LPA effectiveness with power variability issues	248
5.5.1	Impact of intra-die and inter-die process variations on the actual security metrics adopted in LPA	248
5.5.2	Impact of intra-die variation on the leakage model	249
5.6	Investigation on the static power variability in TEL circuits	251
5.6.1	Practical considerations on LPA attacks against TELs	251
5.6.2	Leakage distribution on single iDDPL gates	254
5.6.3	Leakage model of an iDDPL circuit	254
5.6.4	LPA attacks	257
5.7	Conclusion	258
6	Practical evaluation of PAAs against cryptographic circuits	261
6.1	Introduction	261
6.2	<i>SESAMO</i> project: evaluation of the hardware security of cryptographic devices	261
6.2.1	Objective of this activity	261
6.2.2	Description of the measurement setup <i>SCLab</i>	262
6.2.3	Description of the cryptographic cores used in the experiments	271
6.2.4	Description of the experiments	274
6.3	Conclusions	287

7	Conclusions and final remarks	289
7.1	Relevance of full-chip simulations and design-time evaluation in chip-card manufacturing	289
7.2	Summary of the research contribution	290
7.3	Future directions and improvements	292
A	Implementation of 4x4 S-Boxes	295
B	Description of the Matlab scripts for PAAs attacks	297

Chapter 1

Physical security in submicron technologies

1.1 Introduction

Following the considerations done in the introduction to this thesis, the most important question in physical observable cryptography is how the security level of a cryptographic device re-maps in modern technologies. The trend of reducing dimension and power consumption of a circuit leads to the need of re-evaluate and possibly update the level of security of the entire system. Indeed, physical leakage of a certain implementation designed some years ago can be dramatically different from the leakage of the same implementation designed with a more modern technology. The reason is that the devices are optimized under the perspective of improving the performances in terms of area and power, but these improvements do not take security into account.

In this chapter we recall some introductory concepts about digital design flow and cryptography, with the purpose of highlighting the main issues of hardware security for practical applications. Under this perspective, we first provide a brief introduction to the digital VLSI design; more specifically, we will focus on the ASIC design, which is probably the most useful design methodology to describe, in order to better understand and address the most relevant issues about hardware security. After describing the standard design flow for digital circuits, we introduce the topic of physical attacks against cryptographic circuits, with a particular emphasis on *Side Channel Attacks (SCAs)* [61]. Among SCAs, we focus on *Power Analysis Attacks (PAAs)* [62], which are the most common and popular examples of SCAs and have been chosen as guide line theme of this thesis work. Then, we will recall some well-known *symmetric key encryption schemes* as case studies, and the hardware design strategies to build secure digital implementations. Under this perspective, we will provide a brief discussion on the most common countermeasures against PAAs, with a particular focus on circuit level countermeasures, which are based on the adoption of a specific circuit architecture at cell level. For this purpose, we will show that in order to obtain a secure cryptographic circuit, the standard digital design flow needs to be rearranged by introducing some additional steps, which allow to enhance the level of hardware security of the chip, conducting to the so called *secure*

digital design flow.

1.2 Foundations on ASIC design

1.2.1 Digital design flow strategies for VLSI circuits

The exponential growth of the scale of integration, exemplified by the Moore's law [84] (i.e. the increase in the number of transistors per unit area, or, more generally, the performance of the devices, doubling every 18 months), led to a radical innovation in the design flow of digital systems. If in the early 70's the project was being carried by hand drawing the layout on large sheets of paper in accordance to the geometrical representation of the circuit on silicon, today this is no longer feasible due to the huge amount of active devices in the integrated circuits [101], which is represented by the acronym VLSI, *Very Large Scale Integration*.

The VLSI design represents the combination of two types of skills, architectural and circuital:

1. The architecture design skills correspond to the design of block diagrams modeled at various abstraction levels; they can comprise parts of software that must be performed by dedicated microprocessors or described in a hardware description languages (HDL). In this case, only the logical functionality is considered.
2. The circuit design skill consists in the connection of many transistors or logic cells, which are implemented at layout level through CAD tools. In this case, the electrical operation is taken into account.

In general, there are two typical approaches for the realization of digital circuits: full-custom and semi-custom.

In the full-custom (or, simply, custom) design the chip is implemented *from scratch*, so that the individual blocks composing the system are designed up to the transistor level. More specifically, in this design strategy no third parts modules are used, and all the entities are modeled and designed for that application. There are specific software tools which guide the designer along the custom project, allowing to semi-automate the layout of the chip and reduce the design time. Custom design has the advantage of optimizing the design in terms of power and performance, at the expenses of time and money. For this reason this approach is adopted mostly for the design of few functional blocks (e.g. floating point units and memory cells) or in general for a large number production.

The semi-custom design instead makes use of logic circuits that have already been implemented, usually by third parts. They can be divided into two categories: *cell-based* or *pre-fabricated* units and *array-based* or *pre-designed* elements. FPGA, Sea of Gate, PLD, CPLD are all examples of array-based elements. Among the cell-based units, three different design strategies can be categorized: standard-cells based, macro-cells based and compiled-cells based.

In the standard-cells based design, each cell corresponds to an elementary component, such as a logic gate or a flip-flop, or slightly more complex blocks such as multiplexers or other arithmetic circuits (i.e. full-adder). For every technological

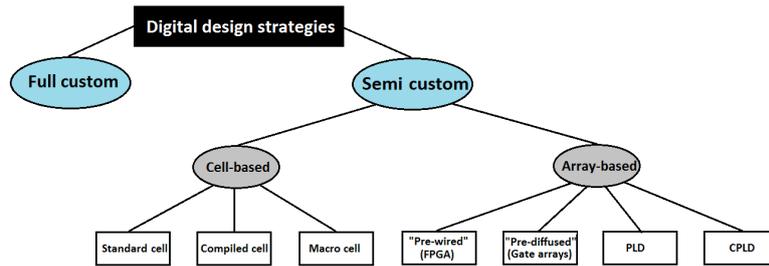


Figure 1.1. A schematic representation of the different possible methodologies to design digital VLSI circuits.

process there is a cell library composed of a number of logic gates characterized up to the layout level. In macro-cell based design, each macro-cell corresponds to a complex block such as an Arithmetic Logic Unit (ALU) or a register-file, but also small CPUs or memory banks. Finally, the compiled-cell based design is characterized by the usage of a particular class of standard-cells and macro-cells, which are automatically generated by means of suitable design tools in accordance to the technological process that the designer wants to use, but not described at layout level. The above discussed design strategies are represented in the diagram of Fig. 1.1.

1.2.2 ASIC vs FPGA

Cryptographic primitives as any other digital circuit can be implemented in both software and hardware, according to the target applications. In this paragraph an overview on the possible cryptographic implementations is briefly discussed [63].

Software implementations are designed and coded in programming languages, such as C, C++, Java, and assembly language, to be executed, among others, on general purpose microprocessors, digital signal processors, and smart-cards. Implementing a cryptographic algorithm in software was typical in the first era of cryptographic smart-cards, given that the flexibility of microcontrollers is suitable for portable applications.

The technology scaling as well as new issues in the field of hardware security facilitated a new hardware-oriented approach, with the development of new optimized algorithms for ultra-constrained devices (e.g. RFID tags). Hardware implementations are in general designed and coded in hardware description languages, such as VHDL and Verilog HDL, and are intended to be realized using ASIC or FPGA.

ASICs are designed all the way from the behavioral description down to the physical layout and then sent for a fabrication in a semiconductor foundry. FPGAs can be bought off the shelf and reconfigured by designers themselves. With each reconfiguration, which takes only a fraction of a second, an integrated circuit can perform a completely different function. FPGA consists of thousands of universal reconfigurable logic blocks, connected using reconfigurable interconnects and switches. Additionally, modern FPGAs contain embedded higher-level components, such as memory blocks, multipliers, multipliers-accumulators, and even microprocessor cores. Reconfigurable input/output blocks provide a flexible interface with the

outside world. Reconfiguration, which typically lasts only a fraction of a second, can change a function of each building block and interconnects among them, leading to a functionally new digital circuit.

As anticipated in the introduction, in this thesis work we focus on the design of cryptographic ASICs. Basically there are two strategies to design an ASIC: using a full custom design flow or a semi custom design flow. According to the discussion in previous paragraph, in the first case the circuit is implemented by designing each single functional units down to the transistor level, whereas in the second case the standard-cells available in a specific technology library are used.

In the following section we are going to describe more in detail the design of an ASIC following a semi-custom design approach.

1.2.3 Description of the semi-custom design flow for ASIC

An ASIC corresponds to a digital system conceived to perform a specific task. In general, it differs from SoC (System on Chip) by the fact that, while in an ASIC the designer tends to design "from scratch" the large part of the system, in the SoC he/she tries to reuse macro-blocks that are available from foundry handbooks. Furthermore a SoC can be used to carry out multiple functions, whereas an ASIC is used to execute a single activity (such as numerical signal processing, fingerprints detection, etc). Today, thanks to the technological improvements, the boundaries between these two device typologies have largely vanished, so many modern SoCs integrate some functional blocks conceived for specific task execution. At the same time, most ASICs contain CPUs and other general-purpose parts, so a modern ASIC is also a SoC and viceversa.

The design flow of a digital ASIC consists of the following key points [101]:

1. Architectural and electrical specifications: represented by an algorithm describing what the system must do. At this level ("system level"), this attempts to identify what are the functional blocks which will make up the device, what they must to do, but there is no information on how they will do it. At this level also information relating to delays, power consumption, cost and clock frequency are known.
2. RTL coding in HDL: the RTL (Register Transfer Level) also called the architecture level, lies a step below the system layer. After defining the system architecture, the designer will see in more detail how each module must carry out its own operations. The RTL specific returns, for each block, the processing performed on data and data transfers between the memory elements with the accuracy of the clock cycle. Today the specifications at RTL level are written using an Hardware Description Language (HDL).
3. Architecture Dynamic Simulation: it allows to evaluate the functionality of the specifications described in HDL. In particular, through the use of an appropriately designed testbench it will provide stimuli to the block to be simulated, and it will show if the data obtained are congruent to those expected. Dynamic simulation is performed with specific software, NCSim or ModelSim.

4. Design constraints and synthesis with standard cell: the design constraints such as frequency, maximum fan-out of the gates, delays, maximum area, etc. are defined. Then, each sub-block composing the system is synthesized using a specific software such as Design Compiler. The constraints are used to guide this tool in the synthesis.
5. Static Timing Analysis (STA) on each block: corresponds to the heart of the design of digital integrated circuits. The timing analysis allows to identify critical paths and improve them, estimate the maximum clock frequency achievable by the chip, thus identifying the "bottlenecks" of the architecture in order to eliminate them. Also for this step, there are specific programs.
6. Pre-layout static timing analysis: the timing analysis is performed on the whole project.
7. Initial floor-plan through cells guided placement: by means of STA a report file is generated with all constraints necessary to meet the specifications; this file allows to help the layout tool to place and route the cells.
8. Insertion of the clock tree.
9. Extraction of delays from the layout after the global routing, and consequent STA.
10. Detailed routing of the cells.
11. Extracting the actual delays by specific software, such as NCSim and Prime-Time.
12. Post-layout STA.

All these steps must be performed iteratively, so, for example, if the architectural simulation (step 3) is not satisfactory, the designer must return to the RTL coding (step 2) and reiterate the flow. So, in general, chip design is an iterative process, and it may happen that a single step must be repeated several times, until all constraints are satisfied and everything is working properly. The block diagram of the digital flow is shown in Fig. 1.2.

1.2.4 Main issues in the design of submicron integrated circuits

The use nanometer CMOS technology encountered several problems when the active devices have been used to create complex digital systems such as ASIC and SoC using processes below 90nm. While I am writing this thesis, for the most advanced devices the scaling of transistors has arrived beyond the threshold of 20nm. The reduction of the size of active devices, combined with the increase in the number of levels of metal to be used for the interconnections, led to a higher density in the active area of integrated circuits and to a decrease of the switching delays of the logic elements. For this reason, now it is possible to integrate multiple functions in the same chip and achieve higher frequencies.

However, designing digital circuits in deep submicron technologies has revealed

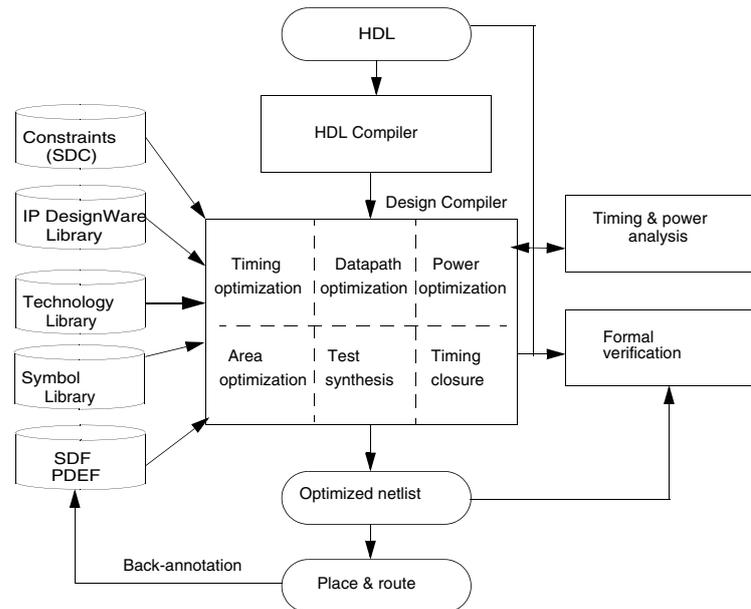


Figure 1.2. Block diagram of a standard digital design flow [119].

some outstanding drawbacks which must be adequately identified and discussed. The main disadvantage of scaling is that also the power supplied to the digital system must unavoidably decrease for physical reasons, and this has a negative outcome on the switching delays, as it takes longer to charge and discharge load capacitances. This increase of delay, however, is partially mitigated by the decrease of the threshold voltages. Another problem related to the high scaling of integration is the increase of process variability, which can impact the functionality of the test-chip and must be properly addressed. In following paragraphs we briefly describe crosstalk, loss of synchronization, leakage currents, and voltage drop, which represent the most common issues in deep submicron design and are described in every book of digital VLSI design [101].

Crosstalk on the local interconnections

Crosstalk is the interference on the signal propagating on a wire due to the coupling with an adjacent wire. It is a critical issue in the design of nanoscaled circuits, because with the technology scaling the metal wires are designed with increasingly reduced distances, and therefore this issue must be adequately described and assessed. Crosstalk has a strong impact on the local interconnection wires of the standard cells, where the most important problem is provided by the capacitive coupling between the wires. Local interconnect wires are typically short and numerous per area unit, leading to a great density; therefore, the amount of cross-coupled capacitance on each wire can be noticeable.

The physical reason of cross-talk is the presence of dielectric (SiO_2) which separates different metal and causes the appearance of parasitic capacitances; when a signal propagates along a wire, the cross coupled capacitance between that wire

and a wire in proximity is charged, and it may create a charge flow in the near wire which represents noise for the signal propagating there. This phenomenon can be critical in particular for dynamic circuits, where wires are floating during a certain period of time. The "victim" wire may also assume another logic value according to the amount of charge injected by the "killer" wire, leading to errors of functionality in the circuit.

The problem of cross-talk must be addressed already during the design phases of dynamic circuits; for this purpose design tools are provided with specific analysis algorithms. A possible technological solution to reduce the effect of crosstalk is to use new dielectric materials, which have a reduced dielectric constant and are called for this reason *low-k materials*. Moreover, a good design strategy is to reduce the number of floating dynamic wires in the points of high density within the chip, and possibly reducing the cross-coupling by separating them by a sufficient distance, even if this unavoidably impacts the area overhead; also the width of the metal wires can impact the amount of cross-coupled capacitances.

Delay and loss of synchronization on the global interconnections

Another problem related to the technology scaling is the increase of probability of having loss of synchronization among signals, which is further emphasized by the fabrication defects of the devices. In general, the problem of synchronization is critical on the global routing interconnect wires, which are long and are characterized by a big amount of overall parasitic capacitances. Cross-talk as well as parasitic capacitances to other metal layers are causes of this problem. Furthermore, differential dynamic circuits are more critical, because the reciprocal delay between two differential signals may impact the functionality of the circuit, but can also reduce the level of security of a circuit, as it will be clear in next section.

In order to avoid such problem, logic gates must be designed so to balance the amount of input capacitance to the driving gate, and prevent that a signal is loaded with a much more different capacitance than its dual. Furthermore, long interconnections must be in general avoided; from this perspective, small design allows to obtain a better balance of the propagation times of the signals. Finally, process variations must be adequately studied and analyzed during the design steps through statistical simulations (e.g. Montecarlo, statistical delay modelization, etc.).

Leakage currents

Prior to the submicron era, the overall amount of power consumption was dominated by the dynamic power consumption, given that as a first approximation the load of a CMOS logic cell is the input capacitance of another CMOS logic cell, which is an infinite impedance at low frequencies and does not draw current. Moreover, there is no conductive path between V_{DD} and GND. The overall effect is that there is no static power consumption in CMOS.

However, the reduction of the dimension of the transistors led to the decrease of the thickness of the gate oxide and a certain amount of charge can be injected in the gate region, creating a current flow directly proportional to the width W of the

device and inversely proportional to the oxide thickness. As a first approximation, it has been estimated a minimum gate oxide thickness of about 30Å.

A possible solution to reduce this phenomenon is to substitute silicon oxide with novel dielectric oxides, named *high-k materials*, with a higher dielectric constant. Nevertheless, static power consumption is given also by the sub-threshold currents and by the reverse-biased currents of a pn junction; as it will be shown in next sections, the overall amount of leakage in submicron devices is inversely proportional to the length L , and therefore it can reach noticeable values. In general, higher is the number of transistors and lower is the technology node, higher is the static power consumption of a circuit.

Voltage drop

Voltage drop is the percent variation of the nominal value of the supply voltage due to high currents drawn by the devices: higher is the number of devices, higher is the drop. The reduction of the supply voltage of digital circuits is a direct consequence of the technology scaling. In order to have a perspective of how much voltage supplies decreased in last 10 years, we can consider that a $0.6\mu m$ was supplied with a voltage of 5V, passing through 1.2V for the $0.13\mu m$ technology, down to less than 1V of modern 18nm technologies.

Furthermore power consumption increased considerably due to the fact that the supply voltage decreases less than the increase of the adsorbed current, given that the number of transistors in a chip continues noticeably to grow with the reduction of dimensions. The adoption of higher working frequencies is another important factor.

For all these reasons, the voltage drop has become an important issue in submicron circuits. Increasing the width of the power supply wires, the number of metal layers for the power supply wires, and the thickness of the power supply nets are possible solutions for the problem of voltage drop.

1.3 Physical security of cryptographic circuits: a review of Side-Channel Attacks (SCAs)

After having introduced the topic of hardware design in deep submicron technologies, in this section we are going to discuss the problem of physical security of cryptographic devices.

A cryptographic primitive can be considered from two points of view: on a side, it can be seen as an abstract mathematical object or black box, which elaborates a specific amount of data through a secret key; on the other side, this primitive is implemented on a real device, with a specific circuit architecture on which a set of instructions are executed. Classical cryptanalysis has the purpose of reverse engineering an algorithm (i.e.: recovering the key of the algorithm with reasonable computational efforts) by studying its mathematical properties. Several efforts have been done with the aim of implementing more robust algorithms able to resist to all the most known cryptanalytic attacks, up to the point that very strong schemes have

been presented and standardized. The *Advanced Encryption Standard (AES)* [2] is an outstanding example in the field of symmetric key cryptography. In last years, a new approach has been introduced to break the security of a specific implementation: instead of trying to find vulnerabilities in the cryptographic algorithm, information leakage can be detected on the physical implementation of the algorithm itself.

Physical attacks on cryptographic devices take advantage of implementation specific characteristics to recover the secret parameters involved in the elaboration. Therefore, they are much less general than cryptanalytic attacks, being effective on a specific implementation, but at the same time they are more powerful because depends on the resistance of the implementation itself. For this reason, these attacks are also called *implementation attacks*.

Several kinds of implementation attacks exist. In literature they are classified among two orthogonal axes [115]:

1. *Invasive vs non invasive*. Invasive attacks require to depackage a chip in order to have direct access to the internal components (e.g. memory, processor, bus, etc.). On the contrary, non invasive attacks do not require to destroy the chip, and exploit only the external information leaked by the device during its normal activity.
2. *Active vs passive*. Active attacks interact directly with the device, trying to change the functionality of the device to extract information from it. On the contrary, passive attacks are based on monitoring the behavior of the device without adding any disturb.

The object of this thesis work is a special class of attacks, named *Side-Channel Attacks (SCAs)* [116], which fall in the category of non invasive, passive attacks. SCAs represent a serious threat for the security of cryptographic circuits, because they aim at extracting information (e.g. the key of a cryptographic algorithm) by exploiting the unintentional physical emissions of the device (e.g. power, electromagnetic field, light, etc) without leaving any trace of their activity. Their dangerousness is in the fact that they can be mounted with relatively low cost equipment by any electronic laboratory. Since their first appearance in scientific literature [61], research in this field have been conducted towards different directions: the discovery of novel side-channel sources, the development of increasingly stronger attacks methodologies, and the implementation of new countermeasures at each level of abstraction. This explains why today security in modern cryptographic applications is considered as a multidimensional issue that can be only guaranteed through a symbiotic work among experts of different areas.

A generic SCAs scenario is depicted in Fig. 1.3, where the terminology adopted in this work is explained. There are two actors involved in the side-channel evaluation: the cryptographic target implementation and the adversary. Two remarks can be done by looking at the figure: first, the target implementation leaks information independently on the fact that the adversary may exploit it; second, the strength of the adversary determines whether this information can be exploited for recovering the attack or not. The purpose of a physical security designer is to design implementations with a reduced (ideally, with no) physical observable leakage, irrespective of the strength of the adversary. Namely, the strength of the adversary determines only

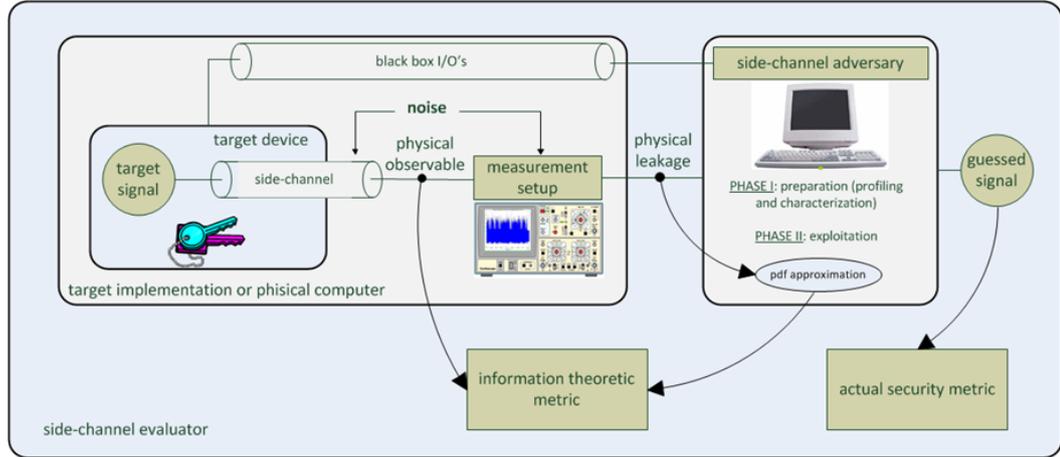


Figure 1.3. Description of a standard side-channel scenario [116].

the velocity (i.e. the minimum number of observation) with which a key can be recovered.

According to the figure, SCAs are closely related to the existence of physical observable phenomena caused by the execution of a specific task, which is always accompanied by a charge flowing in the circuit implementation. It must be pointed out that physical emissions like electromagnetic emissions, current adsorption, heat dissipation, noise, light, are generated by electricity. As a matter of fact, there are several sources of leakage from actual cryptographic devices, which can be exploited by malicious adversaries. In next section, we provide a description of the two most known and important leakage sources in modern submicron CMOS circuits: power consumption and electromagnetic emissions.

1.4 Origin of leakage in submicron technologies

1.4.1 Power consumption of CMOS circuits

Power Analysis attacks (PAAs) [62] represent one of the most effective and dangerous class of SCAs, being simple to be performed in practical applications and relatively low cost. Basically they allow to recover the information processed in a circuit by monitoring the instantaneous power consumption of the circuit itself.

The feasibility of PAAs is based on a physical reason: the instantaneous current drawn from the power supply voltage by a digital circuit in CMOS technology during its operations is strongly dependent on the internally processed data. Namely, under the assumption that an attacker can measure and store the current samples through a specific measurement setup (e.g. an oscilloscope), PAAs exploit the correlation existing between the current samples and a specific logic property of a processed bit.

When a CMOS circuit works, the supply voltage is constant and equal to V_{DD} ; thus the physical observable leakage represented by the instantaneous current is just directly related to the instantaneous power consumption of the circuit:

$$p(t) = v(t) \cdot i(t) = V_{DD} \cdot i(t) \quad (1.1)$$

and the average power is given by the integral on a period of elaboration divided by the period:

$$P_{AV} = V_{DD} \cdot \frac{1}{T} \int_0^T i(t) dt = V_{DD} \cdot I_{AV} \quad (1.2)$$

The power consumption of a CMOS circuit is given by the sum of two main contributions: the dynamic and the static power.

$$P_{tot} = P_{dyn} + P_{stat} \quad (1.3)$$

The total power consumption of a CMOS circuit is given by the sum of the power consumption of all the logic cells composing the circuit, therefore it depends on the number of gates, on their circuit topology, and on the interconnections between them.

In the following paragraphs we describe these two contributions in the typical case of a CMOS inverter gate, highlighting the physical reason beyond these leakage sources.

Dynamic power consumption

The dynamic power consumption of a CMOS circuit is the fraction of power originated from the transitions of the signals propagating along the circuit, and it is related to the dynamic behavior of the devices. The dynamic consumption is given by the sum of two main contributions: the switching and the short circuit power [101]:

$$P_{dyn} = P_{switch} + P_{sc} \quad (1.4)$$

The most important contribution is the switching power, which is due to the current needed for charging and discharging the capacitances of a gate. The dynamic behavior of a logic cell is very often studied through a lumped-C model; the model considers the output capacitance as a single element, which considers all the contributions due to the parasitic capacitances of the devices. The lumped-C model is a good estimation of the total capacitance and will be widely adopted in next chapters. The different contributions will be described in more detailed in next sections.

The switching current arises each time a gate switches from a logic state to another, and therefore is inherently linked to the dynamic behavior of the cell. There are two main contributions to the switching current in the normal operation of a gate: the current $I_{0 \rightarrow 1}$ for charging the output capacitance on the output transition $0 \rightarrow 1$, and the current $I_{1 \rightarrow 0}$ for discharging the output capacitance on the output transition $1 \rightarrow 0$. In the first case, the current flows from the voltage supply into the output capacitance, in the second case the current flows from the output capacitance into the ground.

We now calculate the average switching power of a CMOS gate on an operation period T . Let us suppose there is a set of N possible configurations of signals at the input of the cell. For each data input configuration, the switching power consumption is given by the integral of the instantaneous power on T , which leads to the formula:

$$P_{switch}^{(i)} = \frac{1}{T} \int_0^T V_{DD} \cdot i_{switch}(t)^i dt = V_{DD}^2 C_L f P_{0 \rightarrow 1}^{(i)} \quad (1.5)$$

for $i = 1 \dots N$. $P_{0 \rightarrow 1}^{(i)}$ is the switching activity at the output line for a given input data, and it can assume the value 1 if a transition from 0 to V_{DD} occurs or 0 if not. The probability factor of a gate is given by the frequency of occurrences of a transition, and for N possible data inputs it is:

$$\alpha_t = \frac{\sum_{i=1}^N P_{0 \rightarrow 1}^{(i)}}{N} \quad (1.6)$$

The switching power consumption of the CMOS gate is the average of the power for each possible input:

$$P_{switch} = \frac{1}{N} \sum_{i=1}^N P_{switch}^{(i)} = \frac{1}{N} \sum_{i=1}^N V_{DD}^2 C_L f P_{0 \rightarrow 1}^{(i)} = V_{DD}^2 C_L f \alpha_t \quad (1.7)$$

This formula is valid for any CMOS gate, but usually it is difficult to calculate P_{switch} in practical cases for the presence of the probability factor, which depends on the data pattern statistics. Accordingly, the switching power of a CMOS gate is proportional to the operating frequency, the supply voltage, the switching factor (and therefore to the input statistics) and the output capacitance, but doesn't depend on the dimensions of the devices.

Instead, the short circuit power is given by the temporary short circuit which occurs in the CMOS cell during the switching of the output due to the finite rising and falling times of the signals. It is generated by the current flowing from the voltage supply to the ground terminal. In the simple case of the inverter, there is a short period in which the nMOS and pMOS devices are switched on simultaneously, both for the $0 \rightarrow 1$ and the $1 \rightarrow 0$ transition. Considering again an operation period T , the average short circuit power is given by the integral of the instantaneous short circuit current on T , multiplied by the supply voltage. The short circuit current flows only during a time interval t_{sc} , leading to a current peak with an amplitude equal to $I_{cc,max}$, and can be approximated by a triangle. Under this assumptions, the average short circuit power is:

$$P_{sc} = \frac{1}{T} \int_0^T V_{DD} \cdot i_{sc}(t) dt = V_{DD}^2 I_{cc,max} f t_{sc} \alpha_t \quad (1.8)$$

Similarly to the switching power, the short circuit current depends on the operating frequency, the supply voltage, and the switching factor (and therefore to the input statistics); furthermore, it depends on the time t_{sc} , which in turns depends on the rising and falling time of the signals, and on the current peak $I_{cc,max}$, which in turns depends on the saturation current of the devices, and therefore on the transistors dimensions.

According to the formulas, the dynamic power consumption exhibits always a strong dependance with the input data. This is the reason for PAAs are effective against CMOS circuits. A graphical representation of the dynamic current contributions is shown in Fig. 1.4.

Static power consumption

The static power consumption of a CMOS circuit is defined as the fraction of power linked to the leakage currents flowing in the devices independently from

the signals commutations. In more detail, the leakage currents can be detected after all transient effects linked to the dynamic currents are passed away (steady state condition). The static consumption is given by the sum of the following main contributions [105]:

$$P_{stat} = V_{DD} \cdot I_{leak} = P_{sub} + P_d + P_{gidl} + P_{pt} + P_g = V_{DD} \cdot (I_{sub} + I_d + I_{gidl} + I_{pt} + I_g) \quad (1.9)$$

The first component I_{sub} represents the sub-threshold leakage current occurring when the gate-source voltage, V_{gs} , exceeds the weak inversion point but is still below the threshold voltage V_{th} . In this region, the MOSFET behaves similarly to a bipolar transistor, with an exponential trans-characteristic. The current in the sub-threshold region is given by:

$$I_{sub} = K \frac{W}{L} e^{\frac{V_{gs} - V_{th}}{nV_T}} (1 - e^{-\frac{V_{ds}}{V_T}}) \quad (1.10)$$

where n and K are technology parameters, V_{ds} is the drain-source voltage, and V_T is the thermal voltage and is equal to $\frac{kT}{q}$. Scaling down the supply voltage in CMOS requires also to scale down the threshold voltage V_{th} , in order to maintain the performance of the scaled down logic. From the equation above, it becomes clear that the reduction of the threshold voltage increases the sub-threshold leakage current significantly.

The second component I_d represents the reverse biased pn junction current of a diode in inverse polarization. Diode leakage occurs when a transistor is turned off and another active transistor charges up/down the drain with respect to the former's bulk potential. For example, in the case of an inverter with a low input voltage (Fig. 1.5), the output voltage is high, and the pMOS is on. The nMOS transistor is turned off, but its drain to bulk voltage is equal to $-V_{DD}$ since the output voltage is at V_{DD} and the bulk of nMOS is connected at 0. For the n-well to bulk diode, the leakage current is given by:

$$I_d = I_S (e^{\frac{V}{V_T}} - 1) \quad (1.11)$$

where I_S is the reverse saturation current, and V is the diode voltage. This current is especially significant for an application which spends much of its time idle, since this power is always being dissipated even when there is no switching.

The contribution I_g is the gate oxide tunneling current, which is present when the electric field at the gate is high enough to tunnel through the gate oxide layer some high energy carriers (hot carriers). This causes a decrease of the threshold voltage, and a current flowing into the gate region. The phenomenon of gate tunneling is common in scaled down devices with reduced oxide thickness.

The punchthrough current I_{pt} is the current originating by the phenomenon of punchthrough; this effect is due to the fact that in nanometer technologies the threshold voltage varies as a function of V_{ds} ; for high V_{ds} , the drain and source depletion region approach each other and electrically "touch" deep in the channel. The current arising from this effect, the punchthrough current, varies quadratically with the drain voltage.

Finally, the gate induced drain leakage (*GIDL*) current (I_{gidl}) arises in the high electric field under the gate/drain overlap region causing deep depletion. The *GIDL* occurs at low V_g and high V_d , and generates carriers into the the substrate and drain

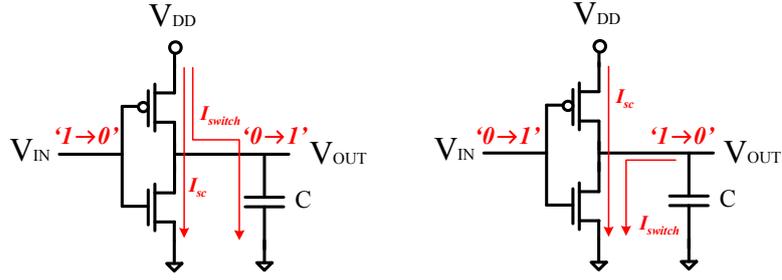


Figure 1.4. Origin of the dynamic currents in a CMOS inverter when output switches from low to high (left) and from high to low (right).

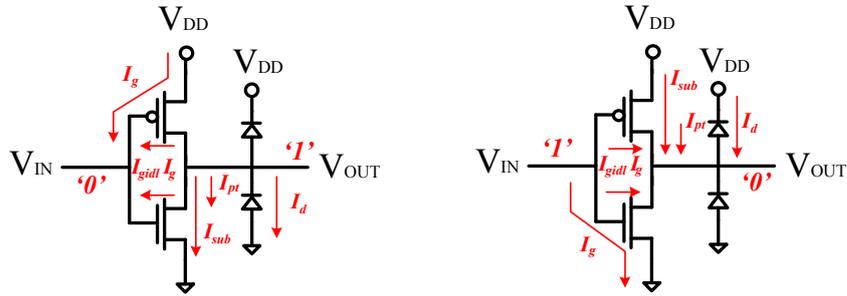


Figure 1.5. Origin of the leakage currents in a CMOS inverter when input is low (left) and high (right) in steady state condition.

from surface traps or band-to-band tunneling.

In common submicron technologies, the sub-threshold current and the reverse biased pn junction current are the most important components of the overall leakage current, and the other contributions are usually neglected in normal operation. Therefore, according to Eq. 1.10 and 1.11 the leakage currents are strongly dependent on the dimension of the transistors, and on the variations of temperature and process in the elements I_S and V_{th} . Furthermore, leakage is strongly dependent on the circuit topology. For all these reasons, static consumption is another important source of leakage for submicron circuits, as it will be shown more specifically in the following. All the contributions of static power consumption are depicted in Fig. 1.5 for the case of a CMOS inverter.

By summarizing, PAAs against CMOS circuits are effective thanks to the strong dependence of the overall power consumption on the input data. According to the formulas presented in this section, this dependence can increase when the load capacitance varies, which impacts the dynamic power consumption, or when the process and temperature variations are taken into account, which impacts the dynamic as well as the static power consumption.

In the general case of a CMOS inverter, during an operating cycle four data configurations are possible:

- $0 \rightarrow 0$: the input signal is a logic '0' and does not switch during the time of observation: no dynamic current is consumed, and the leakage contribution in

Table 1.1. Output transition of a CMOS inverter and corresponding measured power consumption on the V_{DD} pin.

Input transition	Power consumption
0 → 0	$I_{leak,0}$
0 → 1	I_{sc}
1 → 0	$I_{switch} + I_{sc}$
1 → 1	$I_{leak,1}$

steady state condition (i.e. after all transient effects due to the last transition have faded out) is $I_{leak,0}$.

- 0 → 1: the input signal switches from a logic '0' to a logic '1': during the transition, no static power is instantaneously consumed, and the dynamic contribution is given by the short circuit component I_{sc} .
- 1 → 0: the input signal switches from a logic '1' to a logic '0': during the transition, no static power is instantaneously consumed, and the dynamic contribution is given by short circuit and switching currents $I_{sc} + I_{switch}$.
- 1 → 1: the input signal is a logic '1' and does not switch during the time of observation: no dynamic current is consumed, and the leakage contribution in steady state condition (i.e. after all transient effects due to the last transition have faded out) is $I_{leak,1}$.

If for example the current consumption is measured on the power supply pin of the inverter, four possible physical leakages are observed, as shown in Table. 1.1.

1.4.2 Electromagnetic field irradiated by CMOS circuits

The electromagnetic (EM) emanations of digital circuits is another important source of leakage to be described. Any movement of electric charges is in fact accompanied by an electromagnetic field, therefore the EM emanations are caused by the current flowing in any part of the circuit. Every currents arising in a circuit module inside a chip not only produce their own emanations based on the physical and electrical characteristics of the module, but also affect the emanations from other components due to coupling and circuit geometry. Between them, the EM signals emitted by the data processing operations are critical for what concerns security issues. Indeed, similarly as power consumption, also EM radiation is data-dependent. From a theoretical point of view, electromagnetic leakage is usually explained by the Biot-Savart law, which puts in relation the variation of the magnetic induction vector with the variation of the current:

$$d\mathbf{B} = \frac{\mu \mathbf{I} d\mathbf{l} \times \bar{\mathbf{r}}}{4\pi r^2} \quad (1.12)$$

where μ is the magnetic permeability, I is the current carried on conductor of infinitesimal length $d\mathbf{l}$, $\bar{\mathbf{r}}$ is the unit vector specifying the distance between the current element and the field point, and r is the distance from the current element to the

field point. Even if this equation is simplified and does not describe all the physical effects related to the radiation, it reveals that the irradiated field is data-dependent because depends on the current intensity; furthermore, the field orientation depends on the current direction. This data-dependent radiation is just the origin of leakage.

Usually, these kinds of emissions can be categorized in two groups: direct (or intentional) and unintentional [3]. Direct emanations result from intentional current flows; many of these consist of short bursts of current with sharp rising edges which result in emanations observable over a wide frequency band. Often, components at the higher frequencies prove more useful to the attacker due to overwhelming noise and interference prevalent in the lower frequency bands. In a complex circuits, isolating direct emanations can be difficult and may require use of tiny field probes positioned very close the signal source and/or special filters so as to minimize interference from other signal sources; getting good results may necessitate having to decapsulate the chip packaging [41] [100].

On the contrary, unintentional emanations are more interesting regarding the security evaluation. They are mainly caused by the technology scaling, which increases the complexity of modern CMOS devices resulting in electrical and electromagnetic coupling between components in proximity. Even if these electromagnetic couplings are not dangerous for the functionality of a digital circuit, from a security point of view they represent an important leakage source. Unintentional currents result in many compromising emanations in several unintended ways. Such emanations carry information about the currents and hence the events and relevant state of the device. These emanations manifest themselves as modulations of carrier signals generated, present or “introduced” within the device. One strong source of carrier signals is the harmonic-rich “square-wave” clock signal propagated throughout the device. In CMOS devices, a current flows when there is a change in the logic state of a device, as seen in previous paragraph. In addition, all data processing is typically controlled by the clock signal. Each clock edge triggers a short sequence of state changing events and corresponding currents in the data processing units. The events are transient and a steady state is achieved well before the next clock edge. At any clock cycle, the events and resulting currents are determined by a small number of bits of the logic state of the device, i.e., one only needs to consider the active circuits during that clock cycle. These bits, termed as relevant bits in [27], constitute the relevant state of the device.

This class of side-channel attacks is named *Electromagnetic Attacks (EMA)*. In literature there are many example of the EMAs, first introduced by Quisquater and Samyde [100], and further developed in [42] [97]. Basically they exploit the EM emanations by placing coils in the neighborhood of the chip and studying the measured electromagnetic field. EMAs may also provide much more information and are therefore very useful, even when power consumption is available. As a matter of fact, 3D positioning of coils might allow to obtain much more information from the device’s components. Moreover, Agrawal et al. [3] showed that EM emanations consist of a multiplicity of signals, each leaking somewhat different information about the underlying computation.

EMA measurement phase is much more flexible and challenging than power measurement phase, and the observed leakage offers a wide spectrum of potential information, making EMAs even more dangerous than PAAs. On the other hand,

this information may be exploited using the same basic or advanced techniques as for power analysis, even if optimal decision models can be adapted in accordance to the peculiarity of EMA. In essence, EMA is a non-invasive, passive attack, since it consists in measuring the near field during the normal activity of the device without interacting with it. However, this attack is made much more efficient by depackaging the chip first, to allow nearer measurements and to avoid perturbations due to the passivation layer.

1.5 Power Analysis Attacks (PAAs)

1.5.1 Standard methodology to implement PAAs

On the basis of the leakage analysis done in previous section for CMOS circuits, we are interested in the leakage originated by the power consumption. For this purpose we recall some basic concepts related to *Power Analysis Attacks (PAAs)* by introducing the algorithmic attack procedure.

The first PAAs procedure adopted for attacking cryptographic devices was *Simple Power Analysis (SPA)* [72]. Basically SPA is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations. In SPA attacks, the instantaneous power consumption of a device is monitored in order to extrapolate information like the type of algorithm, the beginning and the end of the encryption, the number of rounds, and so on. Usually SPA attacks are based on the knowledge of the implemented algorithm, but the monitoring phase is followed by the collection of a small number of traces, and this makes it difficult to apply more advanced statistical techniques.

In contrast to SPA attacks, *Differential Power Analysis (DPA)* [62] represents a more advanced attack technique which is based on the collection of a large number of traces. In DPA attacks, it is assumed that the device under attack is physically owned by the attacker. According to the previously introduced SCAs model, the adversary records a large number of traces on the basis of the available sources, builds a model of the power consumption of the circuit, and through a statistical distinguisher he/she is able to recover the correct key of the algorithm. More precisely, DPA attacks may assume different names according to the adopted statistical distinguisher, as it will be shown in next paragraphs. In the following, we will use the generic name of PAA to indicate the exact procedure, irrespective of the chosen power model as well as the adopted distinguisher.

The procedure of PAAs is characterized by some steps, which are described below. The assumption is that the attacker has knowledge of the implemented algorithm as well as the circuit architecture of the device.

1. **Choosing an intermediate result of the executed algorithm.** At this step, the attacker selects a specific point of the data-path of the circuit. The output word w of the selected block represents the so called *target function* of the attack, which indicates that the selected word is a function of the key of the algorithm and the input datum, according to the data-path of the algorithm: $w = f(k, d)$.
2. **Selection of the sub-key to be determined.** If we suppose that the

algorithm elaborates blocks of N-bits, the attacker divides the key block length according to a *divide and conquer* strategy; therefore, if for example a N-bit key block is divided in sub-blocks of L bits, the number of sub-keys K is equal to 2^L . Usually, the length of a sub-block k_j , with $j = 1, 2, \dots, K$, is equal to 8 (a byte), and the number of possible keys is 256.

3. **Measuring the instantaneous power consumption.** PAAs are an example of *known plaintexts attacks*: during the normal operations of the device, the attacker stores the values of the N-bits input data together with the actual instantaneous power consumption of the device. Suppose that the number of input plaintexts is D , and the number of current samples for each elaboration is T : a $D \times T$ matrix containing the current samples t_m (with $m = 1, 2, \dots, T$) is then stored, where each row corresponds to a specific input data d_i (with $i = 1, 2, \dots, D$). This matrix is called *actual power matrix* \mathbf{T} , where the generic element t_{im} is the current sample related to the input plaintext i at the time instant m .
4. **Calculating the hypothetical intermediate value.** For each input plaintext d_i , the output word is calculated according to the data-path of the algorithm, using all possible sub-key guesses. Given a sub-key guess k_j , a $D \times K$ matrix is built; it is the so called *hypothetical power matrix* \mathbf{V} , where the element v_{ij} is the hypothetical intermediate value of the data word w : $v_{ij} = f(k_j, d_i)$.
5. **Mapping the intermediate values to power consumption values.** The attacker selects a specific power model for the hypothetical intermediate value, exploiting a logic property of the data word v_{ij} . There are different possible power consumption models for a CMOS circuit; the Hamming weight and the Hamming distance have been widely used in literature and are usually good choices. The Hamming weight model relies on the fact that the dynamic power consumption of a CMOS gate is approximately directly proportional to the amplitude of the current peak: according to the analysis done for the inverter, when the output is a logic '1' the current peak is dominated by the charge current, whereas when the value is a logic '0' the peak is given only by the short circuit current, which is smaller. According to this property, the data word v_{ij} is mapped into a specific power consumption prediction value h_{ij} which approximates the actual power consumption: $h_{ij} = H(v_{ij})$. This way, the *prediction power matrix* \mathbf{H} is built, with dimensions $N \times K$.
6. **Comparing the hypothetical power consumption with the actual power consumption.** As a final step, the actual power matrix \mathbf{T} and the prediction power consumption matrix \mathbf{H} are compared: more specifically, each row h_j of the prediction power matrix \mathbf{H} is compared to each column t_m of the actual power matrix \mathbf{T} , with $j = 1, 2, \dots, K$ and $m = 1, 2, \dots, T$. This means that the attacker compares the hypothetical power consumption values of each sub-key guess to the actual current samples. The result of this comparison r_{jm} is an element of the *comparison matrix* \mathbf{R} , with dimensions $K \times T$. The element r_{jm} contains information about the level of correlation between the

sub-key k_j and the current sample t_m occurring at the time instant m . The attacker selects the maximum value $r_{j^*m^*} = \max[r_{jm}]$ in the matrix; the indexes of this element provides information regarding the correct sub-key, k_{j^*} , and time instant at which the word w is physically processed by the device, m^* .

The above described procedure is depicted in Fig. 1.6 [72]. The steps are repeated for each sub-key block, using the strategy *divide and conquer*, until the entire N-bit key block is determined.

In order that the probability of success of the attack is as high as possible, the traces must be aligned, otherwise no correlation between the actual power consumption and the predicted model can be detected. Moreover each current trace must be composed exactly of the same number of samples T , and they also must be aligned: this means that a specific sample at the time instant m must be originated by the same physical process.

On the basis of the above described procedure, PAAs may differ according to:

- The power model for building the prediction matrix.
- The statistical distinguisher for doing the comparison between the actual and predicted power matrix.

In the PAAs presented in this thesis work we will use the Hamming weight power model. As already mentioned, there are several other techniques for describing the logic property of a n-bit word. We have chosen the Hamming weight because it is very simple to be used in practical application, and is also suitable for hardware implementations based on the adoption of several combinational gates.

The adoption of a specific statistical distinguisher is very important for practical applications, because it allows to reduce the time of an attack, enhancing the probability of success. Anyway, it has been demonstrated [116] that the physical observable leaked by a specific device does not depend on the adopted statistical distinguisher: the probability of success of PAAs against a specific implementation mounted using different statistical distinguishers tends asymptotically to the same value. Thus, the usage of a statistical gauge impacts only the velocity to arrive to the asymptotic value, that is the minimum number of observations.

In the following paragraphs, we describe the most known and used PAAs techniques which adopt a different approach and make use of three different statistical distinguishers: *Differential Power Analysis (DPA)* [62], *Correlation Power Analysis (CPA)* [17], and *Template-based Attacks (TA)* [28]. Several improved techniques have been implemented in last years which exploit the physical leakage in very efficient ways. One of the most important is *Mutual Information Analysis (MIA)* [44] which makes use of information theory concepts; very often, this techniques are also accompanied by more advanced statistical methodologies to reduce noise on the set of measured samples, like *Multivariate Analysis* [114] and *Wavelet Transform* [29]. However, the issues related to the strength of the statistical discriminator is outside the scope of this thesis. For our purpose, it is sufficient to consider that the only difference among different discriminators is how the matrix \mathbf{H} is built and the elements compared, whereas the amount of physical leakage of an implementation does not depend on it.

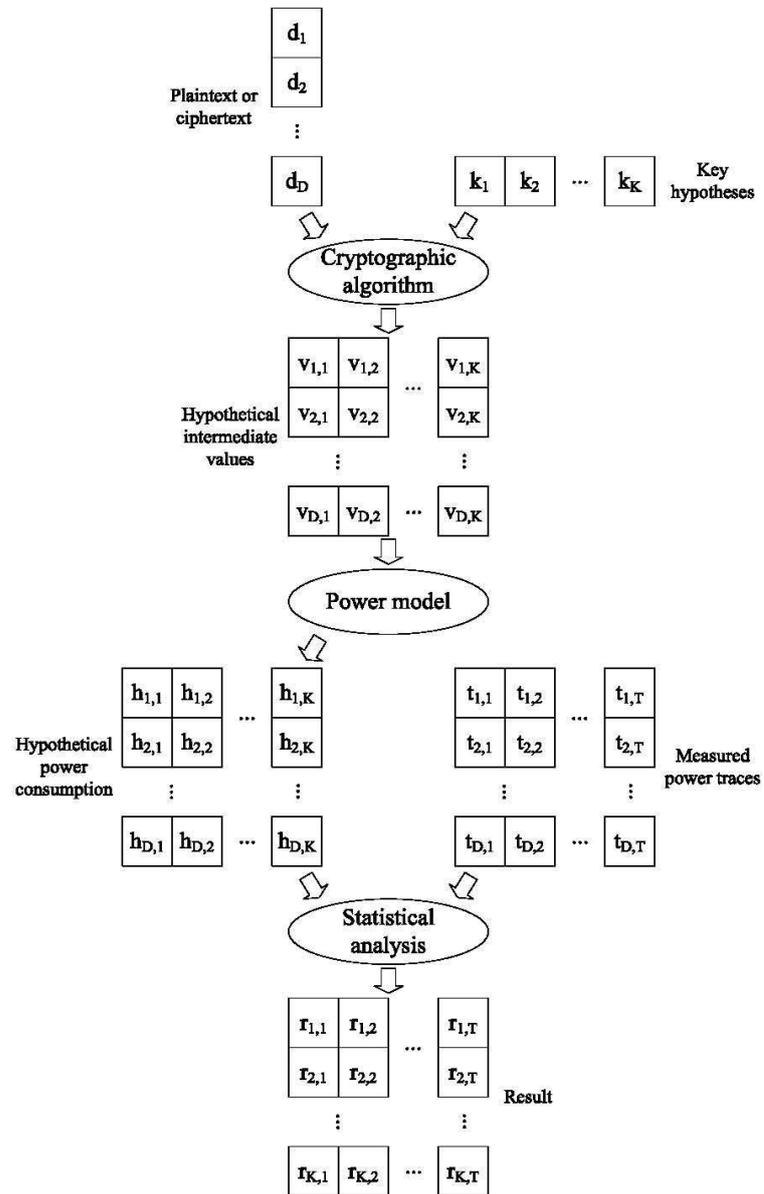


Figure 1.6. Block diagram describing the complete PAAs procedure [72].

1.5.2 Differential Power Analysis (DPA)

Standard *Differential Power Analysis (DPA)* [62] attacks are based on the adoption of the *difference of mean* between the traces as statistical test. The basic idea of this method is to determine the relationship between matrices \mathbf{V} and \mathbf{T} according to the values of the single bits of the word h_{ij} , say bit $h_{ij}(b)$, with $b = 1, 2, \dots, L$.

The set of the m -points vectors \mathbf{t}_i , containing the T current samples of each input plaintext, is divided into two groups, according to the value of the bit h_{ij} : if $h_{ij}(b) = 0$, the current vector \mathbf{t}_i is located in the group S_0 , whereas if $h_{ij}(b) = 1$, the current vector \mathbf{t}_i is located in the group S_1 . Then, for both the subsets, a vector containing the mean values of the T current samples at each time instant is calculated; we refer to the mean vectors related to a specific key j as \mathbf{M}_{0j} and \mathbf{M}_{1j} . The difference $\mathbf{D}_j = \mathbf{M}_{1j} - \mathbf{M}_{0j}$ between these two vectors represents one row \mathbf{r}_j of the comparison matrix \mathbf{R} . In formulas, each element of the comparison matrix \mathbf{B} can be calculated as:

$$m_{1jm} = \frac{1}{\sum_{i=1}^D h_{ij}(b)} \cdot \sum_{i=1}^D h_{ij}(b) \cdot t_{im} \quad (1.13a)$$

$$m_{0jm} = \frac{1}{\sum_{i=1}^D (1 - h_{ij}(b))} \cdot \sum_{i=1}^D (1 - h_{ij}(b)) \cdot t_{im} \quad (1.13b)$$

$$r_{jm} = m_{1jm} - m_{0jm} \quad (1.14)$$

The correct key $k^* = k_{j^*}$ is chosen as the key of index $j = j^*$ which maximizes the amplitude of the values r_{jm} . Furthermore, the time instant $m = m^*$ at which this happens together with its neighbor samples correspond to the period of time when the target word w is elaborated within the device. If the rows of R are plotted for each key guess, some peaks can be detected in correspondence to this time instants only for the correct key k^* ; ideally, for the other keys there are no peaks.

1.5.3 Correlation Power Analysis (CPA)

Correlation Power Analysis (CPA) [17] is an improvement of standard DPA. In literature, these two acronyms are used indifferently as reference to PAAs; however it must be pointed out that they refer to two different methodologies based on two conceptual different foundations.

Standard DPA attacks are based on partitioning the set of current samples measured by the device according to the value of a specific target bit; for this reason, DPA attacks are very often classified as *partitioning attacks*. The elements of the comparison matrix \mathbf{R} are found by calculating the difference of means between these two sets of elements.

On the other side, CPA attacks are based on the calculation of the covariance matrix between the actual power matrix \mathbf{T} and the prediction power model matrix \mathbf{H} :

$$\rho_{H,T} = \frac{\text{cov}(H, T)}{\sigma_H \sigma_T} \quad (1.15)$$

Each element of the matrix ρ_{jm} represents the *Pearson's product moment correlation coefficient* between the predicted power value calculated for the key of index j and the actual value at the time instant m .

The correlation coefficient is the most common way to determine linear correlation between two sets of data. Thus it represents an adequate choice to calculate the comparison matrix \mathbf{R} . In practical applications, a generic element of \mathbf{R} can be calculated by the following empirical formula:

$$r_{jm} = \frac{\sum_{i=1}^D (h_{i,j} - \bar{h}_j) \cdot (t_{i,m} - \bar{t}_m)}{\sqrt{\sum_{i=1}^D (h_{i,j} - \bar{h}_j)^2 \cdot \sum_{i=1}^D (t_{i,m} - \bar{t}_m)^2}} \quad (1.16)$$

and the values r_{jm} represent a good estimation of correlation coefficients ρ_{jm} .

Similarly to DPA, the correct key $k^* = k_{j^*}$ is chosen as the key of index $j = j^*$ which maximizes the amplitude of the values r_{jm} . Furthermore, the time instant $m = m^*$ at which this happens together with its neighbor samples correspond to the period of time when the target word w is elaborated within the device. Again, if the rows of \mathbf{R} are plotted for each key guess, some peaks can be detected in correspondence to this time instants only for the correct key k^* ; ideally, for the other keys there is no correlation. These plots are called *correlation coefficient curves* and in practical applications they are used for choosing the key which generate the highest peaks. Usually, in the correlation coefficient curve of wrong keys many *ghost peaks* can occur, which may misdirect an attacker towards a wrong decision. In general, and it will become clear in the following chapters of this work, a good strategy to prevent ghost peaks is to select a good target function; very often, the choice falls on the output word of a non-linear block of the cryptographic algorithm (e.g. the S-Box block), but in general it depends on the implementation.

1.5.4 Template Attacks (TA)

Template-based Attacks, or more simply *Template Attacks (TA)* [28] are based on a different approach. Standard PAAs attacks as DPA and CPA are based on the collection of a several number of noisy traces, which are in a case partitioned in a subset of values and averaged according to the value of a target bit of the predicted power model (DPA), and in the other case are compared using the correlation coefficient estimator (CPA). In both situations, the statistical distinguisher aims at finding correlation between key and leakage according to the fact that only one part of the leakage is correlated to the key, whereas the remaining part is noise and must be eliminated. For this purpose, several techniques exist for reducing the redundancy of the collected traces in order to reduce noise.

TAs are based on one observation: instead of being averaged, the statistical properties of noise can be estimated in order to build a template of the power consumption of the device. More specifically, each power trace can be characterized by the sum of a deterministic vector, containing the leakage samples, and a random vector, containing noise values; namely, the current samples vector have a multivariate Gaussian distribution with mean \mathbf{m} , given by the deterministic leakages, and covariance matrix \mathbf{C} , given by the variances of the noise samples. The pair (\mathbf{m}, \mathbf{C}) is called template h of the power trace, and it completely characterizes the trace.

Under the assumption that an adversary has available a *clone* of the target device, for several elaborations he/she can build templates $h_{ij} = (\mathbf{m}, \mathbf{C})$ for each pair of input data and key (d_i, k_j) . According to the multivariate Gaussian assumption, current traces can be described by the following probability density function:

$$p(\mathbf{t}; h_{ij}) = \frac{\exp[-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m})]}{\sqrt{(2\pi)^T \cdot \det(\mathbf{C})}} \quad (1.17)$$

This phase of the attack is called *profiling phase*

After having defined a set of templates in the profiling phase, the attacker builds the actual power matrix \mathbf{T} by measuring the current of the device under attack, and calculates the conditional probability $p(k_j|\mathbf{T})$ for each key guess, under the condition that the current traces are statistically independent:

$$p(k_j|\mathbf{T}) = \frac{\prod_{i=1}^D p(\mathbf{t}'_i|k_j) \cdot p(k_j)}{\sum_{l=1}^k [(\prod_{i=1}^D p(\mathbf{t}'_i|k_l)) \cdot p(k_l)]} \quad (1.18)$$

This phase is called *exploiting phase*; the conditional probabilities $\mathbf{p}_j = p(k_j|\mathbf{T})$ are then calculated for all the key guesses, and represent just the rows of the matrix \mathbf{R} . The correct key k_j^* is that which maximizes this set of probabilities. The *a priori* conditional probabilities $p(\mathbf{t}'_i|k_j)$ are calculated by using the Maximum-Likelihood (ML) criterion, so that:

$$p(\mathbf{t}'_i|k_j) = p(\mathbf{t}'_i; h_{ij}) > p(\mathbf{t}'_i; h_{il}), \forall l \neq j \quad (1.19)$$

whereas the probability $p(k_j)$ is equal to $\frac{1}{K}$ under the hypothesis that all the keys are equally likely.

1.6 Symmetric cryptography for SCAs evaluation

1.6.1 Block ciphers as cryptographic case study

Symmetric-key algorithms are a class of algorithms for cryptographic applications that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys may be identical or, alternatively, one key can be obtained through simple transformations on the other key; the key(s) is (are) kept secretly by the device where the encryption and the decryption are executed. According to the Kerckhoffs's principle, "a cryptosystem should be secure even if everything about the system, except the key, is public knowledge", reformulated by Shannon in "the enemy knows the system" [113]; this means that the secrecy of a cryptographic algorithm is completely determined by the secrecy of its key(s). A cipher must be built in order that it is not possible for an attacker to recover the secret key(s) with reasonable computational efforts.

There are two different kinds of symmetric-key encryption algorithms: *stream ciphers* and *block ciphers*. The main difference is that stream ciphers encrypt the digits (typically bytes) of a message one at a time, whereas block ciphers take a number of bits and encrypt them as a single unit, padding the plaintext so that it is a multiple of the block size.

The goal of this thesis work is to study the performances of cryptographic circuits on resisting to SCAs when an attacker tries to recover the secret key by measuring its physical emission. For this purpose, we have considered block ciphers as cryptographic case studies because they operate on large blocks of digits with a fixed and unvarying transformation, which allows to easily implement the PAAs procedure described in previous section; anyway all the results of this work could be extended to the case of stream ciphers. We point out that a good block cipher must be fast and secure, i.e. it must be impossible for an adversary with realistic sources to retrieve the key used even if he/she has access to a black-box capable of encrypting and decrypting the plaintext of her choice, which is in accordance to a SCAs evaluation.

Basically, the aim of block ciphers is to provide a keyed pseudo-random permutation which is then used as the building block of more complex protocols. Block ciphers are usually coupled with a particular mode of operation, with which they can be used to encrypt data. The analysis of the different modes of operation of block ciphers goes beyond the purpose of this work, since we are only interested in assessing the security of block ciphers on a single elaboration unit.

There are two main families of designs for block ciphers: *Substitution-Permutation Networks (SPN)* and *Feistel Networks (FN)*. A SPN block cipher takes a block of the plaintext and the key as inputs, and applies several alternating "rounds" or "layers" of invertible substitution boxes (*S-Boxes*), which confer confusion, and permutation boxes (*P-Boxes*), which confer diffusion to the ciphertext block. On the contrary, a FN block cipher is based on an internal *round function*, which processes two blocks of data word through a substitution box and combines them with a linear operation; between each round, the two sub-blocks are inverted. The advantage of FN block ciphers is that the encryption and the decryption phases are identical and there is no need of having non-invertible S-Boxes, whereas the SPN structure has the property of being inherently parallelized, which makes SPN more adoptable for hardware implementations.

Earlier block ciphers like the *Data Encryption Standard (DES)* [35], a pioneer in the field of symmetric-key encryption, were conceived for software implementations, and they were implemented using a FN. Its successor [2] was also conceived for running on software, but the adoption of a SPN structure allowed it to be implemented also in hardware. With the technology scaling, the need of ever more scaled cryptographic devices has become an important issue in the design of novel block ciphers, which must be equally efficient regarding security and hardware performances (area, power, throughput). Therefore, in last years several novel block ciphers have been specifically designed for running on ultra-constrained devices, leading to the so called *lightweight cryptography* [38]. In order to have an idea about the trend of modern lightweight cryptography, in Fig. 1.7 a list of the most interesting and attractive block ciphers for hardware-oriented lightweight applications is reported [14]. In figure, the most important hardware properties are shown.

1.6.2 Review of two basic block ciphers: Rijndael and Serpent

In this section we provide a brief description of two of the most popular and widely known block ciphers: the *Rijndael* and the *Serpent* algorithms. These two algorithms

Name	AES	CLEFIA	DESX	GOST revisited	HIGHT	ITUbee	KLEIN	KATAN	
Designers	Rijmen et al.	Shirai et al.	Leander et al.	Poschmann et al.	Hong et al.	Karakoç et al.	Gong et al.	De Camière et al.	
Reference	AES conference 98	FSE 2007	FSE 2007	CHES 10	CHES 06	LightSec13	SAP 12	CHES 09	
Cryptographic Properties	Block size	128	128	64	64	64	64	32, 48, 64	
	Key size	128, 192, 256	128, 192, 256	184	256	128	80, 64, 80, 96	80	
	Structure	SPN	GFS	FN with whitening	FN	GFS (rotation)	FN	SPN	Stream-cipher like
	S-Box	1 (8x8)	2 (8x8)	1 (6x4)	1 (4x4)	-	1 (8x8)	1 (4x4)	-
	Rounds	10, 12, 14	18, 22, 26	16	32	32	20	12, 16, 20	254
	Target	HW & SW	HW & SW	HW	HW	HW	SW	HW & SW	HW
Name	KTANTAN	lBlock	LED	mCrypton	Piccolo	PRESENT	PRIDE	PRINCE	
Designers	De Camière et al.	Wu et al.	Guo et al.	Lim et al.	Shibutani et al.	Bogdanov et al.	Albrecht et al.	Borghoff et al.	
Reference	CHES 09	ACNS 11	CHES 11	ISA 06	CHES 11	CHES 07	CRYPTO 14	ASIACRYPT 12	
Cryptographic Properties	Block size	32, 48, 64	64	64	64	64	64	64	
	Key size	80	80	64, 128	64, 96, 128	80, 128	80, 128	128, 128	
	Structure	Stream-cipher like	FN	SPN	SPN	GFS	SPN	SPN	
	S-Box	-	8 (4x4)	1 (4x4)	4 (4x4)	1 (4x4)	1 (4x4)	1 (4x4)	
	Rounds	254	32	32, 48	12	25, 31	31	20, 10	
	Target	HW	HW & SW	HW & SW	HW	HW	HW	SW	HW
Name	RCS	SEA	Serpent	SIMON	SPECK	TWINE	XTEA	Zorro	
Designers	Rivest	Standaert et al.	Anderson et al.	Beaulieu et al.	Beaulieu et al.	Suzuki et al.	Needham et al.	Gérard et al.	
Reference	FSE 95	SCRAA 06	AES conference 98	eprint.iacr 13	eprint.iacr 13	Workshop on LC 11	Internet	CHES 13	
Cryptographic Properties	Block size	32, 64, 128	96	128	32, 48, 64, 96, 128	32, 48, 64, 96, 128	64	64, 128	
	Key size	0, 2040	96, 128, 192, 256	64, 72 / 96, 96 / 128, 96 / 144, 128 / 192 / 256	64, 72 / 96, 96 / 128, 96 / 144, 128 / 192 / 256	64, 72 / 96, 96 / 128, 96 / 144, 128 / 192 / 256	80, 128	128, 128	
	Structure	FN (ARX)	FN	SPN	FN	Tweaked FN	GFS	FN	SPN
	S-Box	-	1 (3x3)	8 (4x4)	-	-	1 (4x4)	-	1 (8x8)
	Rounds	1, 255	93	32	32, 36, 42 / 44, 52 / 54, 68 / 69 / 72	22, 22 / 23, 26 / 27, 28 / 29, 32 / 33 / 34	36	64	24
	Target	SW	HW & SW	HW	HW	SW	HW & SW	SW	µ-controller

Figure 1.7. A list of 24 block ciphers adoptable for lightweight applications.

are considered as pioneers of hardware-oriented symmetric-key cryptography. Both of them were presented in the AES contest in 1999, only one of them was chosen as standard for the AES block cipher, that is the Rijndael algorithm, thanks to its efficiency and speed which made it suitable for software implementations. On the other hand, Serpent was certainly the most robust and secure scheme, but also the slowest algorithm among those presented.

In this work we will use both of them for two different applications, therefore it is useful to give a panoramic on their working principle and properties. As it will be clear in next pages, Serpent is explicitly conceived for hardware applications, thanks to its architecture where eight different 4x4 S-Boxes elaborate in parallel a single nibble of data. For this reason it can be considered a precursor of lightweight cryptography, and several block ciphers in Fig. 1.7 are inspired by it.

AES

The *Advanced Encryption Standard (AES)* [2] is one of the most famous and widely adopted block ciphers, used for a very big number of cryptographic devices, such as smart-cards and embedded coprocessors. AES was introduced as a standard cryptographic algorithm by the "National Institute of Standards and Technology" (NIST) in 2001, after a contest among 15 candidates and became effective as a federal government standard on May 26, 2002, after approval by the Secretary of Commerce. The strength of AES is confirmed by the fact that it is included in the ISO/IEC 18033-3 standard, and is currently used by the USA government as standard encryption algorithm to protect SECRET and TOP SECRET documents, with 192 or 256 bit key blocks.

The Rijndael block cipher is the name of the winner of the contest which took the name AES. Even if today AES and Rijndael refer to the same algorithm, actually they are quite different, because originally the Rijndael algorithm was proposed by authors for variable sizes of the data and key blocks: indeed, in AES input data block is fixed and equal to 128 bit, with a key block equal to 128, 192, or 256 bit; instead, Rijndael specifications [32] state that the cipher must have data and key blocks multiple of 32 bit, at least equal to 128 bit and at maximum 256 bit, without specifying the exact dimensions. In the remaining part of this work, we will refer to the AES-128 algorithm, which is the widely used in practical applications and represents the algorithm working on key and data blocks of 128 bit.

The AES-128 processor is composed of an encoding and a decoding algorithm working on data blocks of 128 bit. In the following, we will refer only to the encryption phase, given that the decryption is based on an inverted procedure with respect to the encryption and therefore they are dual.

The encryption is based on a different number of iterated operations on a 4×4 column-major order matrix of bytes, named as *state matrix*. Each one of the 16 bytes of the data block is organized as an element of the state matrix. The operations of the algorithm are named "round", each one is composed of some sub-operations on the state matrix which work on a single byte of the matrix and are named "layer". These operations involve the formalism and the algebra of "Galois Field" on the finite field $GF(2^8)$, where each byte can be represented as a polynomial of degree 7; two operations are defined, the sum and the multiplication of polynomials on $GF(2^8)$. The sum between polynomials in $GF(2^8)$ is just the exclusive sum of the single coefficients, whereas the multiplication is the multiplication between the coefficients and the product polynomial of degree more than 7 is then divided modulo an irreducible polynomial (modular reduction). In this work we are not interested in an in-depth dissertation of the precise mathematical steps of the single phases, which can be found in more advanced cryptology books or in [33].

The AES-128 encoder is composed of 11 rounds, which are in turns composed of 4 layers (except round 0 and round 10):

1. **SubstituteBytes**: each byte of the state matrix is mapped into another byte according to a specific calculation in $GF(2^8)$ which involves a non-linear operation: the inverse element calculus in $GF(2^8)$; actually, it is possible to describe this step as an associative table 16×16 , where each input byte is splitted in two nibbles: the first nibble is the row index i and the second nibble is the column index j of the table; the element ij is then the output byte of the substitution phase. This table is called substitution box (S-Box), and in hardware it can be efficiently implemented using a look-up table.
2. **ShiftRows**: in this layer, the rows of the state matrix are shifted on the left: in general, the i -th row is shifted of $i - 1$ positions on the left ($i = 1, 2, 3, 4$).
3. **MixColumns**: the bytes of the state matrix are combined using an invertible linear transformation in $GF(2^8)$. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides the propriety of confusion of the algorithm in order to resist to high order cryptanalytic attacks.

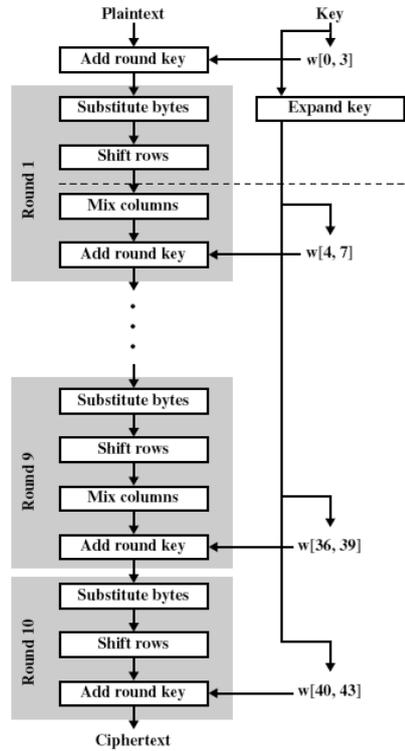


Figure 1.8. Different elaboration steps of the encoding procedure of AES-128.

4. **AddRoundKey**: in this layer each byte of the matrix is exclusively summed to each byte of the 128 round key. A round key is calculated through the function **ExpandKey** of the *Key Scheduler* device, which takes as input the $k - th$ round key and generates the $(k + 1) - th$ round key using non linear operations which involve again the usage of S-Boxes.

The encryption process is shown in Fig. 1.8, where the different elaboration steps are shown. The first round of the encryption, named as round-0, is characterized only by the *AddRoundKey* layer, where the round-0 key of the algorithm is combined directly to the input block.

Rounds 1-9 are composed of all the above described four layers, executed in the same order as they are described; at each round, the XOR operation is executed between the bytes of the state matrix and the round key generated by the Key Scheduler. Finally, in Round 10 the *MixColumn* is not present.

As it will be shown in next chapter, the target function of PAAs is very often the output of the *AddRoundKey* layer of round 0 or the output of the *SubstitutionBytes* layer of round 1; if the round-0 is hardwired in the device, an attack is successful if the attacker can recover this secret key. Accordingly, the *AddRoundKey* and the *SubstitutionBytes* phases are the most critical ones from a security perspective: more specifically, the *AddRoundKey* block is the digital part in which the key is physically combined to the data; on the contrary, the S-Box of the *SubstitutionByte* block is a non-linear operation which confers the propriety of diffusion to the algorithm.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 1.9. S-Box of the *SubstitutionBytes* layer of the AES encoder.

Serpent

The Serpent block cipher was another one of the 15 candidates of the AES contest and was designed in 1999 by Biham, Anderson, and Knudsen [8]. Serpent was considered the most robust scheme, and was selected as one of the 5 finalists in the competition. Unlike AES which is characterized by an elegant and efficient architecture, Serpent was designed only for the sake of security, and resulted to be the slowest.

The standard implementation of the block cipher consists of three main phases: an initial permutation IP, 32 encrypting rounds, and a final permutation FP. The initial and the final permutations do not have any cryptographic significance. They were added to simplify the implementation of the cipher and to improve its computational efficiency, thus we will be focusing only on the encryption phase.

One of the most important property of Serpent block cipher is that it can be efficiently implemented using a bit-slice implementation. Recalling the intent of authors, the basic idea is that just as one can use a 1-bit processor to implement an algorithm by executing a hardware description of it, using a logical instruction to emulate each gate, so one can also use a 32-bit processor to compute 32 different blocks in parallel, actually using the CPU as a 32-way SIMD machine. This is much more efficient than other implementations, in which a 32-bit processor is mostly idle as it computes operations on 6 bits, 4 bits, or even single bits. Serpent has been explicitly designed so that all operations can be executed using 32-fold parallelism during the encryption or decryption of a single block. Indeed the bit-slice description of the algorithm is much simpler than its conventional description. No initial and final permutations are required, since the initial and final permutations above described in the standard implementation are just those needed to convert the data from and to the bit-slice representation.

The block cipher consists of 32 rounds, which make Serpent almost three times higher than the number of rounds of AES as a first approximation. Even for Serpent we will describe only the encryption processor, being dual the operations of decryption. A 128-bit plaintext is ciphered with a 128-bit key word generated

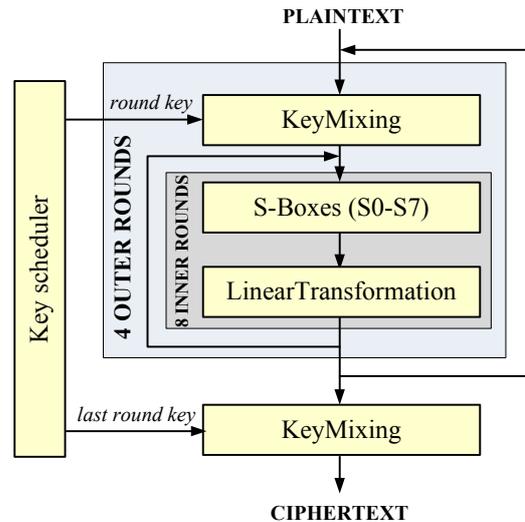


Figure 1.10. Different elaboration steps of the encoding procedure of Serpent.

by a key scheduler. Each round of the encryption is composed of three operations, which similarly to the case of AES can be called layers. The layers are repeated in the following order:

1. **KeyMixing:** a 128-bit data word is stored in the status register is XORed with a 128-bit round key.
2. **S-Boxes:** in this phase the word is processed by a 4x4 S-Box. There are eight different S-Boxes S0-S7, each of one is repeated after eight sub-rounds.
3. **LinearTransformation:** at the output of each S-Box, the 128 bit word in the status register is splitted into four 32 bit words which are linearly combined.

Similarly as AES, the round keys are generated by the the key scheduler. There is an additional 33rd round characterized only by an additional key mixing operation. The different phases of the block cipher are described in Fig. 1.10, where the 32 rounds are composed of 8 stages where the S-Boxes 0-7 are executed (inner rounds), which are repeated 4 times (outer rounds).

Serpent has very good properties which make it suitable for hardware implementations. As it will be shown in next chapters, the bit-slice architecture allows to exploit parallelism, and offers the possibility to implement the 4x4 S-Boxes through optimized combinational designs, which characterized by a small number of Boolean operations. Further information on Serpent can be found in the original paper [8].

1.6.3 Lightweight cryptography: the PRESENT block cipher

After the standardization of AES, the need for new block ciphers has been greatly diminished, given that AES proves to be an excellent and preferred choice for almost all block cipher applications. However, in the very last years the need of cryptographic primitives for extremely constrained environments such as RFID tags

S0:	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12
S1:	15	12	2	7	9	0	5	10	1	11	14	8	6	13	3	4
S2:	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2
S3:	0	15	11	8	12	9	6	3	13	1	2	4	10	7	5	14
S4:	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
S5:	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
S6:	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
S7:	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

Figure 1.11. S-Boxes of the Serpent encoder.

and sensor networks, has increased up to the point that block ciphers developed 15 years ago, just like AES which was designed for software efficiency on 8- and 32-bit processors, may be not more suitable for modern computing devices, even if several attempts of designing novel low-cost implementations of the AES has been done.

Modern cryptography benefits from the adoption of novel optimized block ciphers for lightweight applications: indeed, the decrease of the number of transistors leads to the reduction of the physical observable leakage inherently linked to the emissions of the devices (power, EM emissions, noise, etc), enhancing the constraint on the sensibility of the adversary; furthermore, a hardware efficient cipher is prone to be optimized for being implemented using novel hardware countermeasures in order to reduce the side-channels arising in submicron devices previously described, offering more freedom for the designer. There are also specific constraints when designing lightweight block ciphers: for example memory is very expensive so that implementing S-boxes as look-up tables can lead to a large hardware footprint; this explains why in Fig. 1.7 there are ciphers with 4x4 S-Boxes, 3x3 S-Boxes, or even ciphers without any S-Box at all.

For all these reasons, several ultra-lightweight block ciphers has been presented in last years, with the purpose of combining security performances with hardware efficiency. Among them, we decided to mention PRESENT as an example of hardware-optimized block cipher for new generation cryptographic devices. PRESENT is an ultra lightweight block cipher algorithm, developed by the Orange Labs (France), Ruhr University Bochum (Germany) and the Technical University of Denmark, and designed by Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. VIKKELSOE [15]. It has been carefully designed with the purpose of reducing area and power overhead, without compromising security issues, being 2.5 times smaller than AES [99], and thanks to its hardware performances it is intended to be used in situations where low-power consumption and high chip efficiency is desired. The International Organization for Standardization and the International Electrotechnical Commission included PRESENT in the new international standard for lightweight cryptographic methods as a part of the ISO-29192; this proves that the cryptographic community showed a big interest in this novel block cipher. As stated by authors, PRESENT is explicitly inspired by DES algorithm, the most important block ciphers which was widely used before AES arrival; unlike AES, it was designed with hardware efficiency in mind and complemented with the properties of Serpent, from which also it took the name.

PRESENT is another example of an SP-network block cipher, and consists of

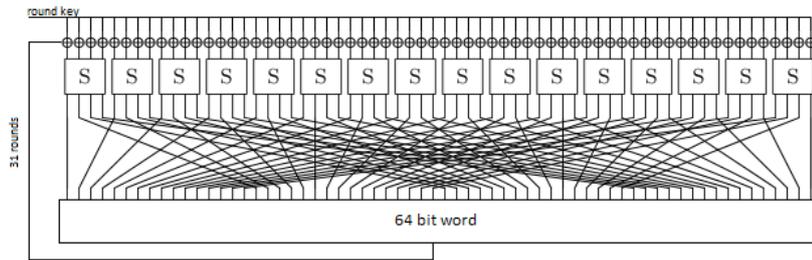


Figure 1.12. Different elaboration steps of the encoding procedure of PRESENT.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Figure 1.13. S-Box of the PRESENT encoder.

31 rounds. The block length is 64 bits and two key lengths of 80 and 128 bits are supported. Each of the 31 rounds consists of three simple layers:

1. **AddRoundKey**: data blocks are XORed bit-to-bit with the round key.
2. **sBoxLayer**: a 4x4 S-Box is repeated 16 times in parallel on a 64 bit word.
3. **pLayer**: the bits of the word are permuted.

The main goal of PRESENT is security and efficiency, but also simplicity, which made PRESENT primitives (e.g. the 4x4 S-Box) reused by other designs. Accordingly, PRESENT was explicitly designed for being implemented in hardware. Authors suggested a key block of 80 bit for a moderate level of security.

1.7 Design strategies for secure block ciphers

1.7.1 Hardware properties of cryptographic primitives

In this section we provide some basic concepts about the design of secure block ciphers. Basically, a block cipher takes a block of data and a key as input and transform it to a ciphertext, using round functions iterated for several times. While software implementations have to process single operations in a serial manner, hardware-oriented implementations offer much more flexibility for parallelism and serialization. Generally speaking, there are three major architecture strategies to implement a block cipher: *basic-iterative*, *serialized*, and *parallelized* [86].

In a basic iterative architecture one round of the cipher is implemented in a single combinational logic; at the output of the path there is an accumulator register, where data are stored and fed back as input of the combinational path for the next iteration. Each round is processed in a clock cycle.

Serialization is obtained by reducing the critical path of the combinational logic. In a serialized architecture only a fraction of a single round is processed in one clock

cycle, reducing the throughput of the circuit. This technique allows to reduce power and area of the circuit and thus represents an interesting solution for lightweight applications.

Parallelism is obtained through techniques like *loop unrolling*: the combinational path of the circuit is increased so that more than one round is performed during a clock cycle. This way the throughput of the circuit is slightly improved with respect to a basic iterative architecture, whereas the increase of power and area overhead is usually big. Indeed, a longer critical path impact the maximum clock frequency as well as the fanout of the logic cells.

In order to improve the performances of the circuit, *pipelining* is very often adopted in combination to one of the above described hardware techniques. Pipelining allows to increase the maximum frequency; if it is combined for example to a parallelized architecture, it allows to obtain an outstanding increase of the throughput.

An interesting discussion and a comparison between different possible hardware versions of a cryptographic circuits are done in [63]. In any case, the choice of the design strategy depends on the requirements of the application, and very often a combination of them is the best choice. In ultra-constrained devices, serialization is probably the best solution because it allows to obtain very low power and area overhead, at the expense of a low throughput.

1.7.2 Building blocks of cryptographic circuits

In accordance to the description presented in previous section, cryptographic block ciphers are characterized by three important phases: combination with the key to encrypt a datum, confusion of the bit through a non linear operation, and diffusion of the bit through a linear operations. Classic and lightweight block ciphers differ only on the complexity of the hardware: the principle of a lightweight cipher is doing the same as a standard cipher, but with less sources in terms of area and power. Therefore, modern lightweight ciphers must be designed with very good performances to resist to traditional cryptanalytic attacks.

In order to reuse the same hardware resources adopting one of the previously described design strategies, the data and the key blocks must be stored anywhere in the device. However, even if RAM and ROM memories were available in earlier microcontroller-based smart-card, the trend is to avoid to insert external memories in low-cost cryptographic embedded systems because occupy too much area and draw too much current. For this reason, the state of the logic circuit must be maintained in registers using conventional flip-flops, which have also a rather large area and power demand. In general, storage of internal data typically accounts for at least 50% of the area and power consumption of a circuit. As it will be shown in next chapters, designing a protected flip-flop for cryptographic applications is even more critical, due to the fact that it must have good performances and at the same time it does not have to leak information.

Combinational logic gates are characterized by the basic Boolean operations like NOT, NAND, NOR, XOR, and so on. Hardware implementations are typically build in a combinational fashion, as the case of 4x4 S-Boxes, exploiting the logic properties

of these gates which allow to implement any logic functions. Earlier cryptographic software implementations made use of look up tables or memories for mapping a complex logic function; this is the example of the 8x8 S-Box of the AES algorithm, which if implemented in hardware using combinational gates would require more than 100 logic cells, leading to a non-optimized solution. For this reason, lightweight cryptographic algorithms very often use small and compact 4x4 S-Boxes. It is clear that the choice of 4x4 S-Boxes is hardware driven, because they require less than a quarter of the area of 8x8 S-Boxes. The properties of the 4x4 S-Boxes have been recently studied [67] [106]; they must be selected very carefully because they may lead to a weaker level of security with respect to 8x8 S-Boxes; however, they are typically obtained by arranging the rows and the columns of the S-Boxes of previous block ciphers (e.g. DES and AES) exploiting their cryptographic properties.

In accordance to the description of PAAs, it is clear that the most critical part regarding the physical security of a block cipher is the S-Box layer. In this thesis work it has been decided to investigate combinational implementations of 4x4 S-Boxes, given their good hardware performances. More specifically, it will be made use of the 4x4 S-Box S_0 of the Serpent algorithm, shown in Fig. 1.11. In Appendix A it will be shown the logic description of the S-Box using Boolean equations; furthermore we present also a logic description of the S-Box of PRESENT for further hardware implementations. These S-Boxes have been chosen as case study of this thesis work, given the relevance of these two ciphers; thanks to their good mathematical performances, these S-Boxes have been also used for other cryptographic algorithms. We point out that implementing S-Boxes with combinational gates allows to occupy less area and optimized the layout, therefore it is the best solution in hardware. Any other 4x4 S-Box of other block ciphers can be implemented in a similar way.

1.7.3 Hardware countermeasures against PAAs: a survey

In this paragraph we provide a brief review of the most known methodologies to counteract PAAs at hardware level [72]. Earlier countermeasures were conceived to overcome PAAs through the insertion of additional dummy instructions at software level in the code running on the microcontroller. Hardware implementations are also based on the addition of redundant circuitry, similarly to the similar approach.

According to the formula in Eq. 1.8, the purpose of the countermeasures against PAAs is to balance the switching activity of digital logic circuits in order to prevent any correlation between data and dynamic power consumption. The identification of the correct key is made according to the amplitude of the coefficients r_{jm} of the matrix \mathbf{R} ; more specifically, by these values it is possible to deduce information on the amplitude of the correlation between prediction and actual consumption as well as the time instants in which correlation is higher. For this reason, so that PAAs can be effective, current traces must be well aligned. By this observation, it is clear that there are two ways to reduce correlation: reducing the amplitude of the correlation or introducing a misalignment on the traces.

In general, hardware countermeasures can be classified on the basis of two orthogonal axes: the abstraction level and the domain at which they act. According to the abstraction level, hardware countermeasures can be distinguished in system, architecture, logic, and transistor level. For what concerns the domain, hardware

Table 1.2. A classification of the most common hardware countermeasures according to the domain and the level of abstraction [72].

	System	Architecture	Logic	Transistor
Time	Random change of frequency	Insertion of random jitter	Asynchronous logic gates	
	Skipping clock pulses	Shuffling of operations	Desynchronization at cell level	
	Multiple clock domains	Insertion of dummy cycles		
Amplitude	Decoupling capacitances	Circuit duplication	Standard-cells DPLs	Full custom DPLs
	Shielding	Architecture masking	Logic masking	Current Mode Logics (CMLs)
	Insertion of noise	Dummy logics		Adiabatic circuits
	Hardware randomization			
	Current flattening circuitry			
	Supply current suppression			

countermeasures act in the amplitude or in the time domain. A survey of the most known and adopted hardware countermeasure is reported in Table 1.2 [72].

On a mathematical point of view, hardware countermeasures attempt to reduce the information leakage in two ways: reducing the intensity of the physical observable or increasing the environment noise. This is in accordance to the definition of signal to noise ratio, which will be described in more detail in the next chapter. Basically this can be done in two ways at physical level: some countermeasures have the purpose of breaking directly the link between processed data and power consumption, and fall in the category of *hiding* countermeasures, because they hide the observable leakage in the amplitude or in the time domain. Other countermeasures try to randomize the power consumption, so that the observed leakage is only poorly correlated to the data; this methodology is called *masking*.

System level techniques have the purpose of preventing the physical observation of the instantaneous power consumption by acting directly on the body of the chip. Decoupling, shielding, randomization, noise insertion, current suppression and flattening circuitry are the most known categories of system level countermeasures and attempt to reduce the amplitude of the physical observable. Among them, decoupling is certainly one of the most efficient techniques, which have been adopted since PAA were discovered for the first time [111]. A decoupling capacitor acts as an intermediate storage element which filters out the high frequency noise components superimposed on the supply voltage. Therefore, the presence of some capacitance on the power supply line of an integrated circuits is mandatory to guarantee the correct functionality, and each EDA tool provides to insert some decoupling capacitors during the back-end design flow. Since PAAs are based on the monitoring of the switching transitions at the power supply line, the presence of decoupling capacitors is an intrinsic countermeasure against PAAs. With the adoption of ever more scaled technologies, new countermeasures which adopt on-chip decoupling capacitors were presented [90][102][103]. On the contrary, time domain system level countermeasures act directly on the clock signal of the circuit, in order to insert random jitter or period variation with the aim of introducing misalignment in the measures.

Architecture level techniques are probably the most used and effective countermeasure for semi-custom designs. In general, architecture countermeasures are based on the insertion of additional circuitry, which aims at hiding or masking the observable leakage. Note that masking can be done also at logic level by the adoption of novel logic cells, which refer to the *Masked Dual-rail Precharge Logic (MDPL)* family [98].

More in general, lower is the level of abstraction, higher is the effectiveness of the countermeasure but also the design efforts. For instance, *Dual-rail Precharge Logic (DPL)* styles are the best strategy to protect a circuit directly at physical level [72]. DPLs are based on a specific compound with a differential architecture, and a two phase data-encoding; they can be implemented both at logic level, using pre-designed CMOS standard-cells, and at circuit level, using novel circuit topologies. Design of DPL styles is widely treated in this work, as we will present a novel hardware countermeasure based on a transistor level optimization. In next section the most known and interesting DPL styles are described.

1.7.4 Metrics to compare the efficiency of hardware implementations

Hardware implementations of a specific countermeasure can be characterized using several performance parameters. If a countermeasure should enhance the level of security of a cryptographic implementation, on the other side it does not have to impact too much the performances of the device itself, given the fact that cryptographic circuits are in general embedded in portable devices where sources are constrained.

The most important parameters to compare digital circuits are: area, power consumption, energy, throughput, time. Similarly to any other digital implementation, cryptographic circuits, and in particular the overhead due to a specific countermeasure, can be compared using these metrics.

- **Area.** Area is one of the most important parameters to assess the performances of a circuit. There are several ways to evaluate the area overhead of a specific countermeasure; very often the area overhead is calculated in comparison to the area occupation of standard-cells in a specific technology library. In ASIC applications, the area overhead is expressed as number of gate equivalents (GE), where a GE corresponds to the area of a CMOS NAND gate, in other cases simply by counting the number of devices. In same case, if a layout implementation of a countermeasure is available the area is expressed in μm^2 ; however this value depends on the fabrication technology and on how the layout has been designed. It must to be noted that differential circuits impact the area overhead of a factor at least equal to 2 with respect to CMOS logic.
- **Power consumption.** According to the description of PAAs countermeasures, power is another outstanding parameter. The resistance of a countermeasure against PAAs relies on the fact that some redundancy is introduced in the logic, with the purpose of balancing the power consumption of the circuit. Thus, this redundancy aims at increasing the overall power consumption. This increment is justified by the fact the security level of the circuit is enhanced, but in any case it must be kept as low as possible.
- **Energy.** The energy is linked to the average current over a time of elaboration. Balancing the average energy is not sufficient to design protected circuits, as it will be widely discussed in the following.

- **Throughput.** The throughput of a digital circuit is defined as the number of bits elaborated in a unit of time, and is expressed in bits per seconds (bps). Cryptographic circuits are characterized by dual encryption and decryption processors, with an almost equal throughput. The throughput depends on the size of the data block N to be encrypted, the number of simultaneously processed blocks NB (in case of parallel data-paths), and the latency of the circuit L , which is the time necessary to encrypt (decrypt) a single block. The throughput is given by the following formula:

$$TP = \frac{N \cdot NB}{L} \quad (1.20)$$

In low cost cryptographic applications, as for embedded devices (e.g. smart-cards), throughput usually doesn't represent a fundamental issue: inserting additional logic unavoidably impacts the speed (and the area) of the circuit.

- **Time overhead.** Time is also not critical in many low cost applications, as only a rather small amount of data is going to be processed. The amount of time for a certain operation can be calculated by dividing the amount of cycles by the working frequency. However, the energy is directly proportional to the amount of clock cycles required for an elaboration. Therefore the time overhead is a measure for portable constrained devices.

1.8 Overview of Dual-Rail Precharge Logic (DPL) styles to counteract PAAs

1.8.1 General description of DPLs

Among all the proposed countermeasure against PAAs, we are interested in an in-depth analysis of the Dual-Rail Precharge Logic styles, which have been introduced with the purpose of reducing the leakage arising from the dynamic power consumption directly at circuit level. The aim of DPLs is to make the dynamic power consumption of the logic cells in the device constant for each cycle of elaboration, which corresponds to guarantee that the switching factor α_t in Eq. 1.8 is always constant irrespective of the input data, and under the assumptions that all other parameters are constant. For this purpose, DPLs increase the amount of area and power of the non-protected counterpart by inserting dummy transistors and/or operations.

DPLs implement the concept of hiding directly at cell level. To do so, they are based on a data encoding which is compatible with a differential architecture. Indeed, in contrast to single-rail (SR) CMOS logics, DPLs are based on the duplication of the signal path like in a differential circuit topology. The same logic function requires more area and power with respect to the CMOS implementation (at least doubled). Moreover, the data encoding of DPL gates is based on the presence of a precharge phase: differential data are first set to the precharge value (first semi-period of the clock), and then are evaluated according to the circuit topology of the logic gate (second semi-period of the clock). The beginning and the ending of the precharge and the evaluation phases are typically controlled by the clock.

1.8 Overview of Dual-Rail Precharge Logic (DPL) styles to counteract PAA#37

Obviously, DPL styles may vary according to the way in which signals are evaluated; furthermore, DPL compounds can be built by using logic standard-cells (semi-custom design) or designing novel circuit templates *from scratch* (full-custom design). The implementation of new specific DPL styles offers an enhanced level of security, at the expense of an increased overhead of area occupation and power consumption. Furthermore, these architectures involve changes in the standard digital design flow, leading to a customized design which unavoidably enhances money and time requirements. Other DPL styles are based on the compound of static CMOS gates; they can be integrated in a standard digital flow, but have the drawback of being extremely sensitive to the electrical mismatches of deep submicron technologies, as it will be discussed in next paragraphs. In general, DPLs require also a double clock frequency in order to guarantee the same throughput of SR circuits, due to the fact that DPL sequential elements have a doubled latency with respect to CMOS flip-flops.

1.8.2 Limitations of DPL styles in submicron technologies

According to the discussion done in previous section, the model on which state-of-the-art DPLs rely suffers on different limitations, which have been revealed with the arrival of the era of submicron technologies. Furthermore, in literature only some DPLs have been evaluated on manufactured nanoscaled circuits, therefore many DPLs are effective only on a theoretical point of view. Other DPLs, such as *Charge Recovery Logics* [87], which exploit the principle of *adiabatic logics* [9], have been specifically conceived for low power applications and can be a promising solution.

In general, the trend of technology scaling in modern electronic circuits leads to novel subtle leakage sources with which a designer had to deal: for example, the electrical mismatches of the signals propagating inside the electronic circuit reveal a data-dependence that cannot be detected by earlier power models. Among them, we can identify at least four outstanding issues which must be adequately assessed during the design of DPLs: glitches of data, early propagation errors, mismatches of the interconnect capacitances, and static power consumption. In the last years, several works have been produced where these leakage sources have been successfully exploited to steal information from a device [43] [64] [109]; this encouraged cryptographic community to find novel mitigation techniques to eliminate, or at least reduce, their impact on the security of a circuit. In general, these techniques make the design flow longer and more difficult, but are in general unavoidable to implement a circuit with a specific DPL style.

In the following paragraphs we will briefly describe all these issues one by one. Static power consumption has been already described in previous sections, and will be investigated in Chapter 6, where a novel attack procedure, named *Leakage Power Analysis (LPA)*, already presented in [5], is adopted to attack a cryptographic circuit implemented using some of the DPLs of Fig. 1.15.

Glitches of data

In CMOS circuit, a glitch or dynamic hazard occurs when signals propagate in combinational paths so to have different arrival times at the input of a logic gate

[101]. The different arrival times lead to a temporary activation of the cell, changing the logic state of its output before all the signals arrive.

Glitches are innocuous in digital circuits, and usually don't affect the functionality of static CMOS circuits. In the case of cryptographic circuits, they are potentially dangerous because the spike in the current pattern originated by a glitch is data-dependent: if a gate switches at different time instants according to the input configuration, the current trace exhibits different patterns.

The effect of glitches occurs only in combinational logics. Using a dynamic circuit architecture, in which the clock signal is routed into combinational gates, solves this issue; this way all the signals are synchronized in two phases, the precharge and the evaluation, and no dynamic hazards can occur. Therefore, DPLs with a dynamic synchronization scheme don't suffer on glitches

Early propagation effects

Together with capacitive mismatches that will be described in next paragraph, timing mismatches [64][118] represent the most challenging aspects to deal with during the design steps of DPLs in modern submicron technologies.

Unlike glitches which affect the data-dependence of static CMOS logics, timing mismatches are effective also in dynamic logics. Timing in dynamic circuits is critical: any delay of propagation between signals can affect the functionality of a circuit. The timing mismatches occurring on two differential signals are the most dangerous, because even if the timing requirements of a specific implementations are met, even small variations of the times of propagation can affect the balance of the instantaneous power consumption.

In DPLs circuits early propagation can affect both the precharge (*early precharge*) and the evaluation (*early evaluation*) phase. Usually, in DPLs the precharge is global and it is executed at the same time (ideally) for all the logic gates; the evaluation is more sensitive on the times of propagation, thus it is more difficult to be mitigated.

Early propagation can be originated at different levels of abstraction: logic, transistor, physical. At logic level, it depends on the inherently imbalance of a specific logic function: for example, a n -input NAND gate evaluates when one signal goes to a logic '1' without waiting for the transition of the other $n-1$ signals. At circuit level, it depends on how a specific logic cell is built: indeed, also if a logic function is symmetric (e.g. XOR function), its circuit implementation may not to be equally balanced. At physical level, it may depend on skews between signals or on the jitter due to process mismatches among different area of a chip.

The effect of early propagation is certainly visible in standard-cells based DPLs, as *Wave Differential Dynamic Logic (WDDL)* [123] and *Masked Dual-rail Precharge Logic (MDPL)* [98] styles, which therefore have been very often implemented in enhanced versions with the presence of some pre-synchronization logics. Obviously this requires more area and power, but in any case it is not an ultimate design solution because early propagation is usually combined to the capacitive mismatches in submicron circuits; for this reason the overall effect must be controlled with a specific strategy based both on logic and circuit level optimizations. This phenomenon will be widely described in next chapter.

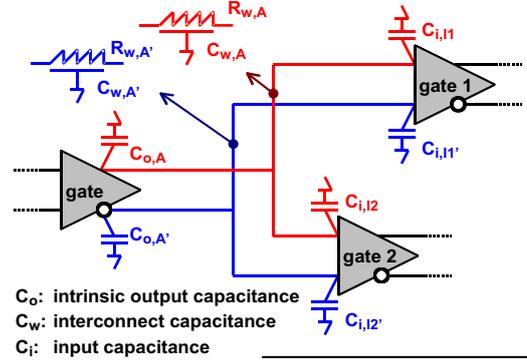


Figure 1.14. A model of the contribution of the load capacitance [124].

Capacitive mismatch of the differential interconnect wires

Capacitive mismatches have been identified as an outstanding leakage source already in earlier works on DPLs [124]. The reason is that the attempt of balancing the dynamic power consumption of a logic gate by equalizing the switching factor is in contrast with the assumption that the load capacitance C_L is constant (see Eq. 1.8), which this is not guaranteed in nanoscaled circuits.

According to the model depicted in Fig. 1.14, the total amount of capacitance C_L is given by the sum of three contributions: the intrinsic output capacitance of the diffusion region of the transistors (C_o), the interconnect capacitance (C_w), and the intrinsic input capacitance of the gate (C_i). The technology scaling leads to a reduction of the dimension of the transistors together with the reduction of the gate capacitance; at the same time, also the parasitic contribution is wanted to be reduced. Thus, the main contribution to the overall capacitance is dominated by the interconnect capacitance, which takes in consideration the contribution of the parasitics and the cross-coupled capacitances on the interconnection. In other words, in submicron region, $C_w \gg C_o, C_i$, and the equation is:

$$C_L = C_o + C_w + C_i \approx C_w \quad (1.21)$$

Therefore, in order to balance the instantaneous power consumption of a differential circuit, it must be guaranteed that $C_{L1} = C_{L2}$ for each differential interconnection. The specification on the matching precision is in contrast with standard full-custom design flow, in which the processor automatically executes the routing of the interconnect wires without considering the resulting mismatch. This represents an outstanding issue in design of modern cryptographic circuit.

In literature several techniques have been presented with the purpose of overcoming this effect at different abstraction level. The first technique to be introduced is masking data at logic level, which helps to randomize the power consumption in order to make it independent from the capacitive mismatch; among masked DPLs, the already cited MDPL is a pioneer, but according to the previous discussions it suffers on timing mismatches which led to the definition of some variants. Other examples of randomized DPLs are *Differential Random Switching Logic (DRSL)* [30] and *Balanced Cell-based Differential Logic (BCDL)* [91], which however have the

same drawback of MDPL. Another way to reduce the side-channel due to the capacitive mismatch is to integrate the digital design flow with a back-end optimization. Among them, we mention the *Differential routing and wire shielding* [122], the *Fat wire method* [124], and the *Back-end Duplication method* [47]. These techniques have the advantage of being adoptable in combination with other previously published countermeasures and are proved to increase the PAAs strength of a circuit, as for example in [53] for a WDDL AES processor designed in a CMOS $0.18\mu\text{m}$ technology with fat wire method, but at the same time have the drawback to require an ad hoc script to be integrated in the design flow, which unavoidably makes the design process more complex. Moreover obtaining a perfect balance during the design through a layout optimization is a very hard-working task which very often leads to imprecise results, in particular if the circuit is implemented into a very scaled technology. Namely, despite of the complexity of the design, the balance is suboptimal. More recently, an alternative is to implement new balanced logic styles which aim at overcoming the unbalance already at transistor level; this is the case of *Three-phase Dual-rail Precharge Logic (TDPL)* [21], *Asynchronous Directional Latch Based Logic (ADLBL)* [[65], and *Delay-based Dual-rail Precharge Logic (DDPL)* [20]. However this logics require a full custom implementation and in general are not immune from timing mismatches which make them useless for practical applications.

1.8.3 A comparison among some popular DPLs

As first step of our research we have executed an analysis of the state of the art DPL styles proposed in last 10 years in the cryptography community. In Fig. 1.15 we have reported a comparison between some selected DPLs of interest [34]. Some of these DPLs are implemented using a masking scheme, in which a mask is first generated by a RNG and then combined to the datum; masking allows to reduce the technological bias due to the differential capacitive mismatch, but cannot help to avoid timing errors due to the lack of synchronization of the signals. More specifically, synchronization is obtained by routing the clock signal also into the combinational gate, like in a dynamic circuit; this allows to avoid glitches and to reduce (not to eliminate) the early evaluation errors.

In table, the constraints column indicates the circuit primitives to use (standard-cells or full-custom) and the back-end optimization to execute in order to reduce the technological bias. To the best of our knowledge, each one of these DPLs has prove to be insecure against PAAs. Furthermore, the technology scaling stresses the main weaknesses because enhances the technological bias.

1.8.4 The secure digital design flow for cryptographic ASIC based on DPLs

The most interesting property of DPL styles is that they can be applied independently of the chosen cryptographic algorithm and the design strategy. Therefore the high-level design of a cryptographic ASIC can be executed irrespective of the specific DPL, and can be completely automatized in a semi-custom design flow. However, according to the discussion of previous paragraph this is true under the condition that some optimization steps are integrated in the flow in order to solve

1.8 Overview of Dual-Rail Precharge Logic (DPL) styles to counteract PAA#1

LOGIC FAMILY	MASK	SYNCHRONIZATION		CONSTRAINTS		EARLY EVALUATION	TECHNOLOGICAL BIAS
		PRECHARGE	EVALUATION	PRIMITIVES	BACK-END		
<i>Wave Dynamic Differential</i>	no	X	X	positive function gates	balanced P&R	yes	high
<i>Masked Dual-Rail Pre-Charge</i>	yes	X	X	majority function gates	no	yes	no
<i>Wave Dynamic Differential no EE</i>	no/yes	V	V	no	netlist post-processing	no	low
<i>Dual-Rail Random Switching</i>	yes	V	X	no	no	no	no
<i>Secure Triple Track</i>	no	V	V	no	delay on sync signals	not investigated	very low
<i>Balanced Cell-Based Differential</i>	no	V	V	no	balanced P&R	not investigated	low
<i>Sense Amplifier-Based</i>	no	V	V	specific library (domino)	balanced P&R	not investigated	no
<i>Charge Recovery</i>	no	V	V	specific library	frequency dependance	not investigated	low
<i>SecLib</i>	no	V	V	specific library	backend duplication	no	very low
<i>Dynamic Current Mode</i>	no	V	V	no	reference voltage distribution tree	not investigated	no
<i>Dual-Rail Transition</i>	no	X	X	no	area optimization	no	very low
<i>Three-phase Dual-Rail Pre-Charge</i>	no	V	V	specific library (domino)	two sync signals	not investigated	no
<i>Delay-Based Dual-Rail Pre-Charge</i>	no	V	V	specific library (domino)	no	not investigated	no

Figure 1.15. Performances and features of some selected DPLs styles.

the problems arising in submicron technologies. This means that when a DPL style is used, some extensions to the standard semi-custom digital design flow must be contemplated. For this purpose, in Fig. 1.16 a detailed description of the design steps for a standard semi-custom flow is provided. The design flow is composed of a front-end and a back-end part. The front-end flow extends from the high level specification to the synthesized RTL netlist, where the technology standard-cells are mapped. The back-end flow takes the RTL netlist as input and executes the place and route of the standard-cells to execute the layout of the circuit, according to the timing constraints and the delay information extracted by the synthesis. These phases are iterative and are executed using some specific scripts, choosing the logic cells by the technology library *TechLib*, where a detailed description of the cells is provided (e.g. maximum fanout, area, delay, power and layout information).

Usually, the design of a non-protected CMOS circuit using DPLs is executed in this way: first, a critical part is identified; then, the ASIC flow is modified by inserting some additional steps for designing a DPL-featured implementation of the critical part, usually implemented through specific scripts, because standard synthesizers (e.g. Synopsys DC Compiler) are specialized in using SR cells and don't support DPL styles.

The procedure of a secure semi-custom design flow for DPLs circuit is represented in Fig. 1.17. There are two main differences between the standard design flow in Fig. 1.16, one at front-end and the other at back-end level. First, the technology mapping is executed by using a secure library *SecureLib*, composed of DPLs, in substitution of the *TechLib*; for this purpose, the wires of the synthesized netlist are duplicated in accordance to a differential architecture; at the same time, some additional signals are routed into the logic gates, for example the clock signal is routed into the combinational cells in dynamic designs, as well as additional logic cells for the conversion of the signal from the SR to the DPL domain. The resulting netlist is then mapped into the *SecureLib*, and the constraints files are obtained by the DPL synthesis. This phase is mandatory in a secure digital design flow with DPL cells. The second difference is in the place and route phase: an additional set of scripts, generically named *optimize.tcl* in figure, is integrated in the design

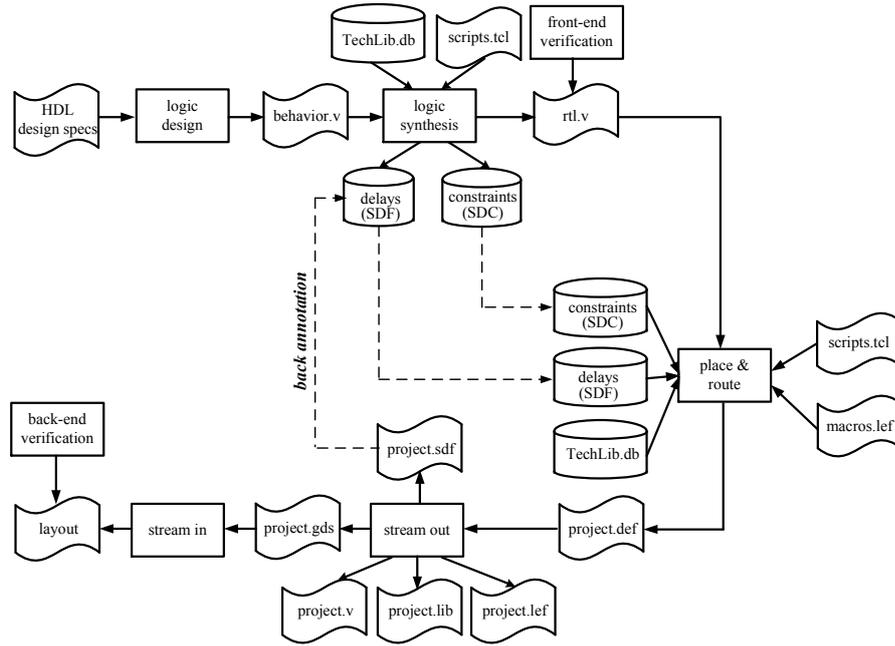


Figure 1.16. A detailed description of a standard semi-custom design flow.

with the purpose of optimizing the place and route in accordance to the issues described in previous paragraphs. Obviously, using dynamic logic gates poses more strict constraints on the timing and delays of the signals, as well as the analysis of cross-talks and capacitive mismatches, thus the *back annotation* as well as the final verification must be executed with big attention in order to guarantee the functionality of the circuit.

1.9 Conclusion

In this chapter we have introduced some basic concepts related to hardware security in submicron cryptographic circuits. After having described the most outstanding leakage sources in common CMOS technologies, we emphasized our discussion on Side Channel Attacks, and in particular on Power Analysis Attacks, which are simple in practice and relatively low cost, thus most dangerous in portable devices of common use like smart-card and RFID tags. We have then finalized the discussion towards the definition of a specific design methodology to design cryptographic ASICs, which differ from the standard design flow for the presence of Dual-rail Precharge Logic styles, probably the most efficient hardware countermeasure because act directly at circuit level.

The above described secure design flow represents just the traditional methodology followed for example in [49] and [125]. As discussed in this chapter, the electrical mismatches in submicron technologies are critical and must be adequately controlled by a back-end optimization phase, for example by controlling the routing with a

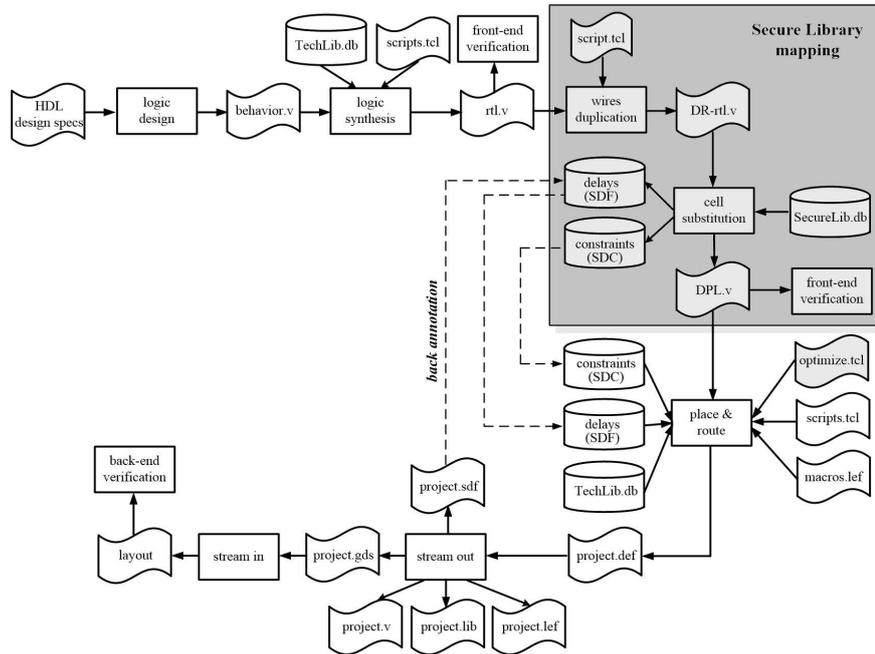


Figure 1.17. A detailed description of a secure semi-custom design flow for implementing a DPL circuit.

balancing procedure as done in [124] and [47]. However these techniques are in general suboptimal, and the complexity of the design very often is not justified by the resulting security level. For this reason, several efforts must be spent in order to develop a specific methodology which implements a hardware countermeasure with limited efforts and suitable for a semi-custom design.

Another parallel issue is to determine useful metrics to evaluate the level of actual security of a circuit during the design phases. Among the possible implementations, ASICs are the hardest to be validated in terms of SCA resistance: once a chip has been fabricated, it is actually impossible to patch any kind of vulnerability, thus a more precise way to define the weaknesses of the device already at simulation level must be adequately defined. Therefore, extensive tests of a prototype chip must be performed before the tape-out, in order to provide an effective validation of the security margin of the device.

In next chapter, we will present a novel hardware countermeasure which acts at different level of abstraction and allows to improve the PAAs resistance of a submicron circuit through a simple additional optimization in the layout. The purpose of this countermeasure is to implement the secure design flow depicted in Fig. 1.17.

Chapter 2

Time Enclosed Logic: a hardware countermeasure to overcome PAAs

2.1 Introduction

In previous chapter we have introduced the main issues related to physical observable cryptography, emphasizing on *Power Analysis Attacks (PAAs)* and the countermeasures to reduce their effect in current submicron technologies.

Many countermeasures against PAAs been proved to be ineffective in practical applications, due to the fact that the security level assessed during the design steps may be based on wrong assumptions. For example, the electrical mismatches which are intrinsic to the design of nanoscaled circuits and are caused by the impossibility to guarantee a perfect balance of the capacitances of the interconnect wires, must be considered and are expected to increase with the technology scaling.

In this chapter we propose a novel hardware countermeasure to reduce the data-dependence of the instantaneous power consumption, which is based on the combination of two methodologies at different level of abstractions: first, a novel logic data encoding is defined in order to balance the average power consumption in presence of capacitive mismatches of the interconnect wires; then a back-end optimization methodology, based on the insertion of an on-chip filter directly in the layout of the circuit, is also described in order to eliminate the residual dependence due to the circuit mismatches. In the following, we will refer to this countermeasure as *Time Enclosed Logic (TEL)*.

The purpose of this chapter is to present and validate through extensive simulations the robustness of this new countermeasure against PAAs when the electrical mismatches, always expected in nanoscaled designs, are taken into account. As first step, we describe the principle behind TEL circuits and provide an accurate description of the power model, considering both the capacitive and the timing mismatches. For this purpose, some assumptions on the adversary strength are done, in particular on the bandwidth and the resolution of the measurement setup.

Once the power model of a TEL implementation is adequately assessed, we discuss the impact of the electrical mismatches, which must be considered and

modelled already during the design of TEL circuit, and the need of inserting some decoupling capacitance in the layout of the circuit. For this purpose, we present a novel design-time metric, which is based on the investigation of the deviation of the frequency patterns of the simulated current traces, the *Frequency Energy Deviation (FED)*. Through this metric, a designer can have a first fundamental evaluation of the leakage property of the implementation already during the design steps, so that he/she can efficiently increase the level of balance of a TEL circuit with a simple additional back-end optimization.

This design methodology has been adopted to validate the leakage property of a case study cryptographic circuit, which implements a specific block of the SERPENT block cipher. The circuit is designed using the CMOS065 technology described in the introduction to this thesis, implemented as a TEL circuit and tested through extensive simulations in order to assess its level of security in terms of PAAs resistance. Simulated PAAs have been done using common statistical metrics as the *Point Biserial Correlation (PBC)* coefficient [50], which aims at assessing the DPA-resistance of a countermeasure, and the *Pearson's correlation coefficient* [17], which on the contrary assesses the CPA-resistance. Furthermore, the Gaussian noise is also taken into account in the experiments in order to make simulations more realistic according to the model presented in [28], which in turns agrees with the formalized analysis done in [116] and the simulation methodology applied in [104] for the case of nanoscaled chips. As a security metric, we have used the *Minimum number of Traces to Disclose the key (MTD)* [122], which is widely used in the context of both simulated and experimental attacks.

The cryptographic circuit has been designed with the purpose of being manufactured as an ASIC, using a specific circuit template designed in order to implement the TEL principle. The design steps of the circuit as well as implementation details of the adopted logic family will be described in more detail in next chapter.

All the experiments presented in this section have been done with SPICE level simulations performed in Cadence environment.

2.2 Security assessment of hardware countermeasures

2.2.1 Assumptions on the adversary model adopted for the PAAs procedure

The security evaluation of a countermeasure executed using SPICE level simulations deserves to be clarified in order to better understand advantages and limitations of such an analysis.

Simulations are fundamental during the design steps of a cryptographic circuit, and have an ever more increasing relevance because allow to predict the leakage of a device before fabrication, reducing verification and debugging costs [55]. They provide the designer with a fine-grain evaluation of the leakage of a countermeasure, which allows a very accurate estimation of the power consumption profile with a fine timescale; furthermore the collected traces are noise-free and perfectly aligned, which is actually impossible in practical measurements, but during the evaluation phase this usefully provides a conservative approach to validate the circuit implementation.

Moreover, it must be pointed out that SPICE level simulations are really meaningful if an adequate testbench which comprises also peripherals circuitry is taken into account, as discussed in [59]. Indeed, an important issue regarding the evaluation of the security of an implementation through SPICE simulations is related to the model adopted for the attacker. More specifically, in this chapter we use the model of *Perfect Attacker (PA)*, which is compatible with a conservative approach when SPICE-level simulations are adopted.

A Perfect Attacker is guessed to have unbounded sources in terms of time, bandwidth and memory to perform an attack. The model of Perfect Attacker [11] is in accordance to the precision provided by the simulated noise-free traces exported by Cadence. Following this model, the assumptions we do are the following: (1) the attacker has full knowledge of the data-path of the circuit; (2) the attacker can build an accurate power model of the leakage by knowing the power characteristics of the DPLs circuit and the exact instants in which leakage samples occur inside a clock cycle; (3) the attacker can profile the power consumption of the circuit using an unbounded profiling phase, which in simulation means collecting the noise free traces corresponding to each possible input. The strategy of considering a profiled acquisition phase corresponds to the situation in which the attacker owns a clone of the target device and builds a template of the device itself, similarly to a realistic PAAs methodology [28].

The only hypothesis we do is **to assume that the adversary has limited resources in terms of time resolution and bandwidth**, which can be reasonably supposed in standard low cost PA attacks setup. This assumption has been done starting by an observation: if a cryptographic device can be characterized through its power traces, the device cannot be considered secure any more, since an attacker can always model the power consumption of the device and build power templates in order to extract information at any time instant. But if we relax the constraints on the bandwidth of the acquisition, the physical leakage of our implementation could be hid in the time domain resulting invisible to the attacker setup. This is just the main purpose of TEL circuits.

Indeed **the effectiveness of TEL circuits is based on the hypothesis that in practical applications the attacker has a measurement setup with a limited resolution** [72]; therefore if the observable leakage is hid in the time domain beyond the time resolution of an oscilloscope, the acquisition fails at collecting relevant time samples and no leakage can be detected, irrespective of the strength of the statistical distinguisher. Thus, not even a perfect statistical distinguisher could detect information in the stored samples.

2.2.2 Metrics for the evaluation of the physical leakage

The evaluation of the security margin of a specific cryptographic circuit is a challenging issue and must be accurately performed before the chip fabrication. In this paragraph a brief review of the state-of-the-art security metrics adopted to quantify the physical observable leakage is provided together with a discussion on the model which is used to assess the strength of an adversary in PAAs.

In the context of PAAs, *Signal-to-Noise Ratio (SNR)* has been proposed as a theoretical system level metric to assess the physical leakage of a hardware

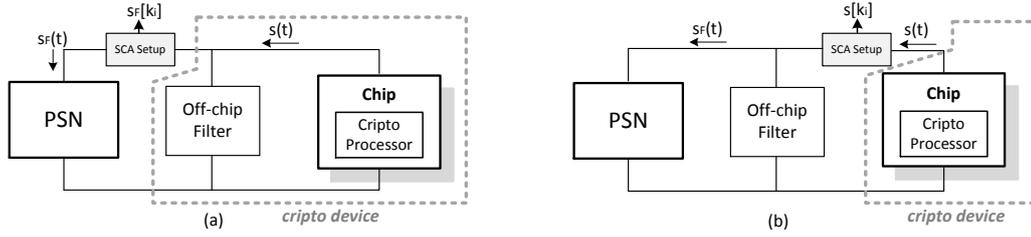


Figure 2.1. Two possible realistic scenarios for practical PAAs: an attacker can measure filtered (a) or non-filtered current traces (b).

implementation [72]. For digital circuits, SNR is defined as the ratio between signal and noise power:

$$SNR = \frac{P_{sig}}{P_{noise}} = \frac{P_{sig}}{P_{sw.noise} + P_{el.noise}} = \frac{\sigma_{exp}^2}{\sigma_{sw.noise}^2 + \sigma_{el.noise}^2} \quad (2.1)$$

In Eq. 2.1, P_{sig} represents the fraction of the overall power consumption which contains relevant information *potentially exploitable* by an attacker, and can be calculated as the variance of the leakage trace σ_{exp}^2 . P_{noise} is the overall noise in PAAs measurements and depends on unpredictable physical effects, as electronic noise power ($P_{el.noise}$) and switching noise power ($P_{sw.noise}$).

Based on the definition of SNR in Eq. 2.1, a hardware countermeasure has the purpose of reducing SNR. This can be done in two main ways: reducing the signal intensity or enhancing the noise power. Hiding in the amplitude or in the time domain, randomizing, or confusing the physical observable leakage by inserting noise are all common techniques that can be adopted at each level of abstraction [72].

Hiding means breaking the link between processed data and power consumption directly at physical level; usually hiding countermeasures reduce the physical observable leakage of a device forcing the current pattern to be as flattened as possible irrespective of the input data. On the contrary, randomization is obtained by masking the intermediate values processed inside the circuit through simple mathematical operations, keeping unchanged the current variance but de-correlating it from the data, and this can be done at system as well as circuit level. Similarly, adding noise means enhancing the variance of the switching noise by inserting dummy hardware circuitry or software instructions.

The main disadvantage of a system level metric as SNR is that it is too much generic and cannot be used to measure the intensity of the exploitable signal on a practical level. Consider for example two possible scenarios for practical PAAs (Fig. 2.1). In both cases, the off-chip filters (i.e. RLC circuitry) are inserted to remove the high frequencies components on the Power Supply Network (PSN) which arise in digital integrated circuits. The signal $s(t)$ is the physical side-channel emission from the cryptographic processor (e.g. current, electromagnetic field, etc). With reference to PAAs, $s(t)$ is the current drawn by the cryptographic chip which is directly related to the power consumption. The current $s(t)$ represents the physical observable that an attacker wants to measure in an ideal scenario, because it represents the non-filtered instantaneous current of the integrated circuit and contains

relevant information regarding a datum processed in the crypto-processor.

In a non-invasive attack scenario, the device under attack (DUA), namely the cryptographic device (e.g. a smart-card, a FPGA, an embedded system, etc), is considered as a black box, and an attacker wants to measure the emissions of the device as near as possible to the original source in order to minimize noise and directly acquire the original leakage $s(t)$. In the general case of Fig. 2.1(a), the attacker has only access to the external signal $s_F(t)$ filtered by the decoupling impedance inside the device, because he/she cannot/does not want to physically eliminate them. Instead in the case of Fig. 2.1(b), which is more realistic if we consider for example a smart-card as target DUA, the power is externally provided to the integrated device (i.e. by a card reader or tag), thus an attacker has direct access to the original signal $s(t)$. After the analogue to digital conversion during the acquisition phase, the discrete N-vectors s_F and s contain the side-channel information, but if we suppose to be in an univariate attack scenario, the exploitable signal power associated to the sampled point k_i is different in the two cases:

$$|s_F[k_i]|^2 \neq |s[k_i]|^2 \quad (2.2)$$

Therefore the side-channel information is different in the two cases, and this difference can lead a designer to underestimate the level of security of an implementation already during the design steps.

In the contest of the design and validation of circuit level solutions to overcome PAAs, the *Normalized Energy Deviation (NED)* has been adopted as a specific criterion in some early works [121] [123]. This metric is calculated as the deviation of the integral of the current traces (i.e. the energy), and represents an estimation of the ability of a logic gate to balance the average current on a clock cycle, i.e. the energy of the cell:

$$E_{AV} = V_{DD} \int_0^T I_{DD}(t) dt \quad (2.3)$$

$$NED = \frac{\max[E_{AV}] - \min[E_{AV}]}{\max[E_{AV}]} \quad (2.4)$$

E_{AV} is the average of the distribution of the energies per cycle, calculated for a number N of inputs where all possible data transitions are considered. The energy per cycle is calculated measuring the adsorbed current $I_{DD}(t)$ on the V_{DD} pin and integrating it in a clock cycle. NED relies on the fact that a decoupling capacitance is always present in the PSN of a device, therefore balancing the energy is sufficient to remove any data-dependence in the power consumption.

However, the adversary model depicted in Fig. 2.1 clearly shows that these assumptions can lead to wrong results. More specifically, in nanoscaled circuits several electrical mismatches can be detected in real measurements: timing mismatches, due to different propagation times [64] [118], and capacitive unbalances of the differential interconnect wires [49] [124] are today the most common sources of leakage in submicron cryptographic circuits, and cannot be assessed through an integral metric as NED. This explains why balancing the energy is very different from balancing the instantaneous power consumption, and the adversary model which is assumed in simulations plays a crucial role during the evaluation.

More recently, information theoretic metrics have been proposed as an efficient

tool to evaluate the strength of a countermeasure both in simulation and in real measurements [71] [116] in a more rigorous way. The methodology is basically divided in two steps: first, an information theoretic analysis of the leakage traces is performed in order to build a template of the power consumption; this can be done for example collecting a very high number of traces in practical measurements or simulating the circuit for each possible data input, and then considering a Gaussian model for the superimposed noise. Then, a security analysis with a specific statistical distinguisher is conducted. The validation of the circuit leakage is obtained by calculating the entropy of the key class K before any side-channel attack is applied (i.e. the conditional entropy) and the mutual information:

$$H[K|\mathbf{S}_q] = \mathbf{E}_k \mathbf{E}_{s_q|k} - \log_2(\Pr[K = k|\mathbf{S}_q = \mathbf{s}_q]) \quad (2.5)$$

$$I(K; \mathbf{S}_q) = H(K) - H[K|\mathbf{S}_q] \quad (2.6)$$

where \mathbf{S}_q is the random vector containing the side-channel observations generated with q queries to the target cryptographic implementation and s_q is a realization of this random vector. In practice, we have: $\mathbf{s}_q = [s_1, s_2, \dots, s_q]$, where each s_i is the side-channel trace corresponding to one given query. The term $\Pr[K = k|\mathbf{S}_q = \mathbf{s}_q]$ is the conditional probability of a key guess k given a leakage \mathbf{s}_q .

As seen in Fig. 2.1, the designer does not have control on the model of the measurement setup that an adversary can build, and a protected circuit can be really protected only for a particular situation. It is clear that the methodology and the contest of a PAAs evaluation must be adequately defined before testing the effectiveness of a countermeasure with a specific security metric. In general, a countermeasure can be considered as secure, if it is secure in the worst case situation, as that depicted in Fig. 2.1(b).

2.3 A novel data encoding for differential dynamic cryptographic circuits

2.3.1 Brief review of Return To Zero (RTZ) logics

Dual-rail Precharge Logic (DPL) [72] is a family of circuits with two basic properties: the information is encoded using two differential wires; the clock period is divided in a *precharge* and an *evaluation* phase allowing a dynamic data encoding. DPLs have been widely adopted as a countermeasure against PAAs thanks to their property of balancing the dynamic power consumption by guaranteeing that the switching factor of a logic gate is always 1. In this kind of logics, the clock of the circuit is routed into the flip-flops as well as the combinational gates, and the data-path is doubled. An interface circuit provides to convert the Single-Rail (SR) signal from the CMOS circuit section into the Dual-Rail (DR) domain.

The *Return To Zero (RTZ)* [66] logic circuits are a special class of DPLs widely used in the context of cryptography; in the RTZ data encoding both differential signals are reset to the minimum voltage supply (0V) during the precharge, and only one of them evaluates to V_{DD} according to the bit to be processed. The differential signals are represented using the formalism (A, \bar{A}) . The data encoding is done in the voltage domain, according to which line is charged, and each wire stays at a

voltage (0 or V_{DD}) for a period equal to $\frac{T_{CK}}{2}$: if $(A, \bar{A}) = (1, 0)$, wire A is charged at V_{DD} and a bit-1 is processed; on the contrary, if $(A, \bar{A}) = (0, 1)$ the processed bit is 0 (Fig. 2.2). Actually, the RTZ data encoding allows to hide a datum in the *voltage domain*, as shown in Table 2.1, and the time interval $\frac{T_{CK}}{2}$ in which a RTZ cell evaluates represents the relevant period of the data encoding, that is the interval of time in which the information is visible and can be potentially detected by an attacker.

Several implementations of RTZ logics have been proposed in literature, that differ according to the target circuit template. The most famous are the *Wave Dynamic Differential Logic (WDDL)* [123] and the *Sense Amplifier Based Logic (SABL)* [121]. The main difference is that WDDL gates are composed of CMOS standard cells and the clock signal is routed only into differential flip-flops; instead SABL gates are full custom logic cell built for dynamic circuits, therefore the clock signal is routed both into sequential and combinational gates.

Several papers in the last ten years have demonstrated that RTZ logics suffer on several issues whose effect is expected to increase with the technology scaling. These issues are inherent to the unavoidable electrical mismatches arising in nanoscaled technologies, already discussed in previous chapter. In order to overcome these problems, many modified versions of both WDDLs and SABLs have been published, as well novel and more advanced logic styles. However, to the best of our knowledge an ultimate solution have not yet been obtained.

2.3.2 Basic principle of Time Enclosed Logic (TEL) circuits

Time Enclosed Logic (TEL) circuits adopt a different methodology to encode a bit. A CMOS SR signal is converted into a differential signal pair so that both of them are set to 0 (*discharge phase*). At the clock rising edge, one of the signal of the differential pair (A, \bar{A}) , called *asserted* signal, has a rising edge and evaluates to V_{DD} (*evaluation phase*). After a time interval equal to δ , the other signal, called the *delayed* or *non-asserted* signal, also goes to V_{DD} , completing the transition (*postcharge phase*). From this time instant the gate remains in a frozen state until the beginning of the next clock period.

On the basis of the time diagram in Fig. 2.3, described in Table 2.2, a differential pair (A, \bar{A}) can have three possible states, according to the order in which the differential signals are charged: if rail A is charged before rail \bar{A} , a bit 1 is mapped; on the contrary, a bit 0 is mapped; no information is encoded when $\delta = 0$. It is straightforward to note that the relevant period of the TEL data encoding (δ) is lower than the relevant period of RTZ logics ($\frac{T_{CK}}{2}$).

The most important logic property of the TEL data encoding is the *functional completeness*. Basically, completeness is the property of a logic circuit to be represented by a symbolically complete logic expression. A set of Boolean functions is functionally complete if all other Boolean functions can be constructed from this set and a set of input variables is provided. Functional completeness allows to automatically synchronize the signals at the output of a combinational path, without incurring in timing mismatches. In literature, completeness has been already mentioned as a valid logic countermeasure to design balanced asynchronous logics [39]. Recently,

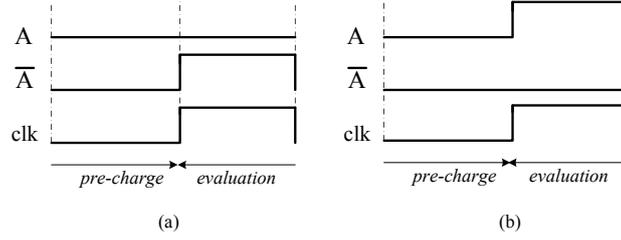


Figure 2.2. Timing diagram of logic-0 (a), logic-1 (b) signal in RTZ logics

Table 2.1. Description of the Return-to-Zero data encoding.

Voltage				RTZ domain	CMOS mapping
Precharge	Evaluation	A	\bar{A}		
A	\bar{A}	A	\bar{A}	(A, \bar{A})	IN
0	0	0	V_{DD}	$(0, 1)$	0
0	0	V_{DD}	0	$(1, 0)$	1
0	0	V_{DD}	V_{DD}	$NULL$	<i>Invalid</i>

in the context of PAAs the property of completeness has been exploited to build *delay-insensitive* circuits [16] [66], which are an example of balanced asynchronous logic style able to overcome the electrical mismatches. The property of completeness has important consequences in the design of TEL cells, as it will be discussed in next section.

From a security point of view, the most important property of the TEL data encoding is that the relevant information is enclosed inside a time period δ , and each electrical mismatch gives origin to a deviation of the current pattern only during this time window. When an attacker wants to monitor the instantaneous power consumption of a TEL circuit for example through an oscilloscope, if the sampling period of the oscilloscope is greater than δ , no relevant samples are captured during the acquisition phase, and thus PAAs are unfeasible. As an example, if the attacker uses an oscilloscope with a sampling rate of 2GS/s (with a maximum bandwidth of 1GHz due to the Nyquist's limit), which is rather common in a practical PAA scenario for low cost applications, an interval δ less than 500ps, reasonably achievable in common submicron technologies, is sufficient to prevent PAAs. The value of δ is chosen by the designer in order to guarantee a certain level of security in a given technology.

We point out that this property is in accordance to the attacker model described in previous section, where a limited bandwidth of the measurement setup is supposed.

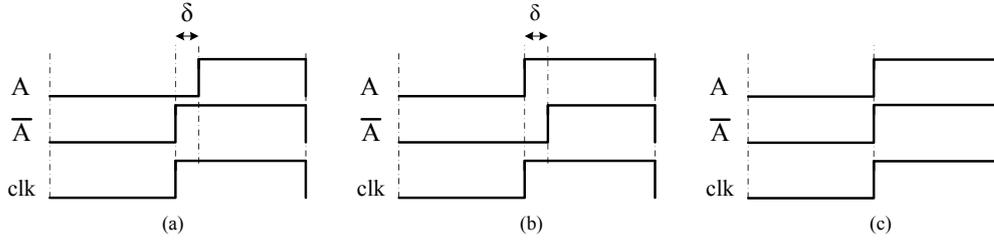


Figure 2.3. Timing diagram of logic-0 (a), logic-1 (b), invalid (c) signal in TELs

Table 2.2. Description of the Time Enclosed Logic data encoding.

Discharge		Voltage		Postcharge		TEL domain	CMOS mapping
A	\bar{A}	A	\bar{A}	A	\bar{A}	(A, \bar{A})	IN
0	0	0	V_{DD}	V_{DD}	V_{DD}	$(0, 1)$	0
0	0	V_{DD}	0	V_{DD}	V_{DD}	$(1, 0)$	1
0	0	V_{DD}	V_{DD}	V_{DD}	V_{DD}	$NULL$	<i>Invalid</i>

2.4 Description of TEL circuits

2.4.1 Cell templates for TEL circuit implementations

According to the property of completeness of TEL encoding, differential signals propagate with a reciprocal delay δ , but any mismatch on the propagation times can be balanced by exploiting the property of completeness. Indeed, each differential signal has one and only one complete commutation cycle in a clock period. On the contrary, in the RTZ protocol only one differential signal switches, whereas the other one stays at the precharge value 0.

This property has a direct consequence on the circuit topology of a logic cell. With reference to the *Wave Differential Dynamic Logic (WDDL)*, which is a standard-cell based instance of RTZ logic, each gate can be built as the compound of two functions, which must satisfy the following condition [72]:

$$F_1(D_1, D_2, \dots, D_n) = \bar{F}_2(\bar{D}_1, \bar{D}_2, \dots, \bar{D}_n) \quad (2.7)$$

The asserted signals are connected only to the function F_1 , whereas the non-asserted signals are only connected to F_2 . Therefore, these logic functions are compatible with the RTZ encoding if and only if these functions are *positive monotonic Boolean functions*. We recall that a positive monotonic Boolean function is a logic function that changes its output value in a monotonic way: when an input D_i has a transition $0 \rightarrow 1$ ($1 \rightarrow 0$), the output Q has a transition in the same direction $0 \rightarrow 1$ ($1 \rightarrow 0$) or remains in its state $0 \rightarrow 0$ ($1 \rightarrow 1$); the same is for the non-asserted input \bar{D}_i and output \bar{Q} . Further mathematical details can be found in [72].

The main drawback of WDDL circuits is that F_1 and F_2 are generic, thus they are implemented with different logic compounds, which are strongly sensitive on electrical mismatches, both capacitive and timing. On the contrary, according to the property of completeness of TEL encoding, it is possible to introduce some

redundancy on the input data pattern dependence and build balanced logic gates. As depicted in Fig. 2.4, the generic structure of a combinational TEL cell can be seen as the composition of two half circuits which implement two independent Boolean functions F_1 and F_2 , each one depending on all the input data set. The differential signals are set to complementary values on the basis of the TEL formalism, and are calculated according to the intended logic function of the cell. Namely, functions F_1 and F_2 must satisfy the following relation:

$$F_1(D_1, D_2, \dots, D_n, \bar{D}_1, \bar{D}_2, \dots, \bar{D}_n) = \bar{F}_2(D_1, D_2, \dots, D_n, \bar{D}_1, \bar{D}_2, \dots, \bar{D}_n) \quad (2.8)$$

A fundamental property of a positive monotonic function is that for all input values set to 0, the output value of the function is 0; similarly, for all input values set to 1, the output value is 1. In WDDL gates the possibility to have inputs equal to 1 is excluded by the RTZ data encoding. On the contrary, by exploiting the presence of a postcharge phase, missing in RTZ logics, both the differential TEL signals are characterized by the same voltage transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ in a clock cycle. For this reason, the differential TEL signals always switch according to a *strictly positive monotonic function*, because each time an input signal has a transition $0 \rightarrow 1$, the output responds with the same transition, first on the asserted and then on the non asserted wire, and cannot remain in its previous state. As a consequence, the logic functions F_1 and F_2 can always be designed as a compound of NAND functions. Indeed NAND is the minimum complete logic function, and thanks to its logic truth table it can be adopted to design self-synchronized balanced TEL circuits, given that when all inputs are 0, output is 0 (discharge), and viceversa when inputs are 1 output is also 1 (postcharge). The evaluation is admitted only during the time interval δ .

As an example, some basic TEL combinational templates composed of only NAND functions are depicted in Fig. 2.5. These circuits are fully compatible with the TEL encoding irrespective of how NAND functions are implemented. It is worth noting that a NAND function can be implemented with 4 transistors, but without its dual counterpart it can be implemented with a minimum of 2 transistors (e.g. in dynamic circuits), exploiting the dynamic behavior of the data encoding and reducing the redundancy of the circuitry. Note that, and this will be shown in next sections, there is no need of balancing the external capacitances in TEL circuits, which are inherently insensitive to the omni-present electrical mismatches. Furthermore, the propagation times are also balanced because signals propagate along the same logic path.

There are several possible circuit implementations for the cell template of Fig. 2.4. Basically, these implementations may differ according to:

- The target design.
- The synchronization scheme.

The target design represents the architecture on which the circuit must be designed. There are two basic strategies to implement a new logic style: a circuit template can be designed by using the pre-designed standard-cells in a certain technology library, or alternatively by designing a new topology *from scratch*. In the first case, the

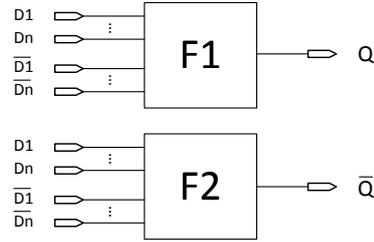


Figure 2.4. Cell template of a TEL gate composed of two independent circuit halves implementing the logic functions F_1 and F_2 .

design is suitable also for FPGA-oriented applications, where the performances of the circuit can be assessed faster and in a more flexible way than an ASIC-oriented design, which instead requires a more complex and longer procedure. In the second case, the security of a full-custom logic style SPICE-level simulations are crucial for security evaluation, and real experiments can be executed only after manufacturing the ASIC. Accordingly, the implementation of a TEL circuit can be executed by using CMOS standard cells, pass transistor logics, or full custom gates.

In regard of synchronization, the circuit may require the distribution of a clock signal or not. In the first case, the clock signal can be routed into the flip-flops, as in the case of static CMOS gates, or also into the combinational gates, as in the case of dynamic logics. In the second scheme, the circuit does not require a clock signal to be routed in the design and the signals propagate asynchronously in the circuit. Furthermore, the circuit may require the adoption of additional control signals to generate delays between the differential signals, according to the TEL encoding.

In Fig. 2.6(a), a specific full custom cell template for TEL circuits is presented, the *improved Delay-based Dual Rail Logic (iDDPL)* style. The iDDPL style is a full custom synchronous logic, implemented in a differential Domino circuit topology, where the clock signal is routed also into the combinational gates. The TEL data encoding is generated by using an additional control signal, as it will be shown in next chapter, implemented with a static voltage which controls the polarization of a delay element. The iDDPL style can be adopted for an ASIC implementation of TEL circuit, as described in next chapters.

With reference to Fig. 2.6(a), the differential TEL signals processed by the cell are encoded in the data formalism described in Table 2.2. The working principle of the cell will be widely described in next chapter, together with a circuit topology for combinational and sequential gates; furthermore, a circuit to convert the signal from the CMOS into the TEL domain will be also described.

In Fig. 2.6(b) also the circuit template of a *Sense Amplifier Based Logic (SABL)* inverter [121], as an example of full custom RTZ logic. SABL will be used as a reference template in order to have a fair comparison with TEL implementation.

2.4.2 A first-order model of the dynamic power consumption

In this paragraph a first order model of the dynamic power consumption of a TEL gate is calculated and compared to the case of RTZ logics. According to the data encoding defined in Table 2.2 and the timing diagram of the signals depicted

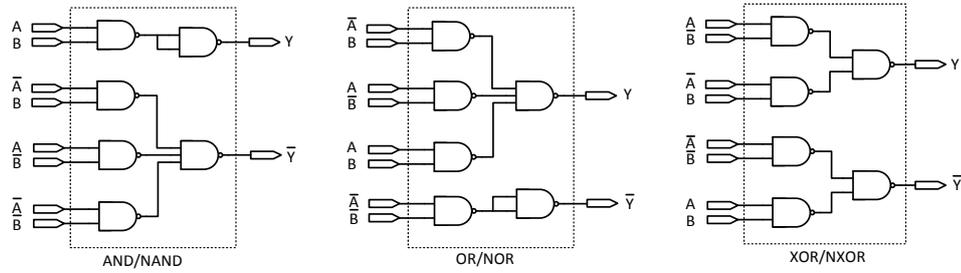


Figure 2.5. Cell templates of basic combinational gates, compatible with the TEL encoding and designed with only AND logic gates.

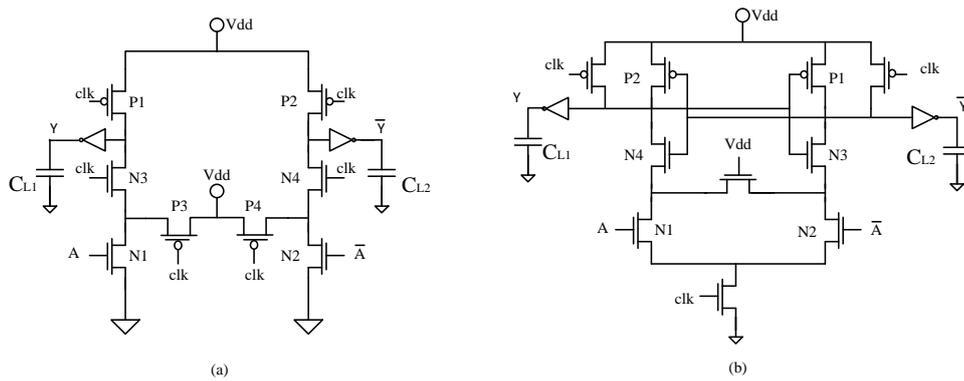


Figure 2.6. Full custom cell template of a TEL-compatible inverter (iDDPL style) (left) and a RTZ inverter (SABL style) (right).

Table 2.3. Model of the power consumption for a TEL and a SABL inverter cell with an unbalanced load.

	(Y, \bar{Y})	1 st semiperiod		2 nd semiperiod		$P_{dyn}^{(1)}$	$P_{dyn}^{(2)}$	P_{dynTOT}
		Y	\bar{Y}	Y	\bar{Y}			
RTZ	(0,1)	0 → 0	0 → 1	0 → 0	1 → 0	0	$V_{DD}^2 C_{L2} f$	$V_{DD}^2 C_{L2} f$
	(1,0)	0 → 1	0 → 0	1 → 0	0 → 0	$V_{DD}^2 C_{L1} f$	0	$V_{DD}^2 C_{L1} f$
TEL	(0,1)	0 → 1	0 → 1	1 → 0	1 → 0	$V_{DD}^2 C_{L1} f$	$V_{DD}^2 C_{L2} f$	$V_{DD}^2 (C_{L1} + C_{L2}) f$
	(1,0)	0 → 1	0 → 1	1 → 0	1 → 0	$V_{DD}^2 C_{L1} f$	$V_{DD}^2 C_{L2} f$	$V_{DD}^2 (C_{L1} + C_{L2}) f$

in Fig. 2.3, the current trace of a TEL circuit is composed of three dynamic peaks: the first occurs at the precharge, the other ones are related to the evaluation and the postcharge respectively, and are separated from a time equal to δ . Through the time enclosed data encoding, each capacitance is charged and discharged once in the clock cycle.

For a differential circuit, the dynamic power consumption is given by the sum of the power consumption of each differential half circuit. With the technology scaling, the interconnect wires have a strong impact on the overall capacitance, and thus under the perspective of an automatic routing procedure the differential load capacitances are expected to be different, as shown in Fig. 2.6 where $C_{L1} \neq C_{L2}$. The model is simplified considering the contribution of the load capacitances as lumped capacitors. In Table 2.3 the data transitions occurring during the two semi-periods on each output rail, the correspondent dynamic power consumption, and the overall dynamic power consumption are reported.

As reported in table, in RTZ logics there is always one transition on a rail, and the switching factor is equal to 1 irrespective of the input data; however, only one rail makes a transition at each clock cycle, whereas the other remains at the precharge value; thus the dynamic power consumption is unbalanced, and in presence of mismatched capacitances the overall dynamic power exhibits two different values.

On the contrary, in TEL gates *the switching factor is 1 for both the differential lines*, thus the dynamic power consumption is balanced irrespective of the input data and depends on the sum of the output capacitances, not on their single values.

In order to assess the level of unbalance of a specific logic circuit, which depends on how the interconnect wires are routed, in this thesis work we use the *Mismatch Factor (MF)* to indicate the degree of unbalance of the differential interconnect wires, defined in [47] as the ratio between the maximum and the minimum differential capacitances:

$$MF = \frac{C_{Lmax}}{C_{Lmin}} \quad (2.9)$$

The value of MF depends in general on the complexity of the layout and on how the routing is carried out. A standard automatic routing may generate high values of MF on the differential wires. Obviously, smaller is the area overhead of the circuit, smaller is expected to be the MF, but in high density regions MF can be much higher than 1. In cryptographic applications, area is one of the most important requirements, therefore for a small design a reasonable approximation is considering mismatch factors in the order of 2 (moderate mismatch) or 3 (high mismatch) for the differential wires already in pre-layout simulations.

2.4.3 Timing constraints of a TEL circuit

In this section we discuss and define the timing specifications of TEL circuits. According to the data encoding presented in last section, basically a TEL circuit is based on a hybrid synchronization scheme: on a side, the discharge is globally synchronized with the clock edge; on the other side, the evaluation is also synchronized with the clock but the postcharge is asynchronous and depends on the propagation times of the signal along the combinational path.

With reference to the multi-stage circuit in Fig. 2.7, the SR signal is first converted into a TEL differential pair with a nominal δ . In accordance to the data encoding defined in Table 2.2 and Fig. 2.3, the differential signals propagate along the pipeline at different time instants. The delay at the output of each gate is not equal to the nominal δ ; these timing mismatches are intrinsic to the asymmetry of the logic cells implementing a combinational function.

This phenomenon, that we name *fluctuation effect*, is described more in detail in next paragraph. The main drawback of this effect is the dynamic variation of the length of the time interval δ . In order to prevent any timing violations and guarantee at the same time that the level of security is preserved, the circuit in Fig. 2.7 must meet three fundamental requirements:

1. δ can decrease at the output of a combinational logic, but cannot increase ($\delta_{CL} < \delta$).
2. δ_{CL} at the output of a combinational logic must be always greater than the setup time of the flip-flop ($t_{su} < \delta_{CL}$)
3. δ must be regenerated by the flip-flop ($\delta_{FF} = \delta$) (neglecting the propagation time of the flip-flop).

These conditions represent the timing constraints of the circuit.

We call *propagation time* of a TEL combinational logic gate $\Delta\delta$ the variation of the output δ from the input value, and it is given by the difference $\delta_{IN} - \delta_{OUT}$. For a combinational circuit CL_i of Fig. 2.7, δ_{IN} is just the nominal δ_{FF} generated by the flip-flop and δ_{OUT} is δ_{CL_i} . The condition 1. reflects into a condition on the sign of $\Delta\delta$, that is $\Delta\delta > 0$. As it will be deeply analyzed in next chapter, it is always possible to build logic gates with balanced times of propagation of the differential signals through an adequate design of the pull down network of the logic gates, in order to satisfy condition 1.

Accordingly, the propagation time of the critical path $\Delta\delta_{CP}$ is defined as the maximum variation of δ along a combinational logic, and is calculated by the difference between the delay at the input of the first gate (i.e. the nominal δ) and the minimum delay δ_{CP} at the output of the path:

$$\Delta\delta_{CP} = \max[\delta - \delta_{CL}] = \delta - \min \delta_{CL} = \delta - \delta_{CP} \quad (2.10)$$

The propagation time of the critical path depends on the number of stages N , and can be calculated as the sum of the propagation times of the N gates of the path:

$$\Delta\delta_{CP} = \sum_{i=1}^N \Delta\delta_i = N \cdot \overline{\Delta\delta} \quad (2.11)$$

where $\overline{\Delta\delta}$ is the average propagation time of the TEL combinational gates in the path in a certain technology.

The setup time of the flip-flop t_{su} is defined as the minimum delay that the TEL signals at the output of the critical path (i.e. at the input of the following flip-flop) must have in order to be correctly sampled. The condition 2., together with equations 2.10 and 2.11, provide the timing constraint on the critical path:

$$\delta_{CP} = \delta - \Delta\delta_{CP} = \delta - N \cdot \overline{\Delta\delta} \geq t_{su} \quad (2.12)$$

Equation 2.12 poses a constraint on the maximum number of stages N_{MAX} that can be inserted in a combinational TEL circuit in a given technology node:

$$N_{MAX} = \frac{\delta - t_{su}}{\overline{\Delta\delta}} \quad (2.13)$$

In the calculation of the timing constraints of a TEL circuit, the designer should calculate the exact propagation time of each combinational stage considering also the interconnect impedance.

As stated before, the designer chooses the value of the nominal δ according to the level of security he/she wants. However the minimal value of δ is defined by the technology limits: considering a combinational path composed of a minimum of two combinational gates, for instance two AND/NAND gates, we can calculate δ_{MIN} as:

$$\delta_{MIN} = 2 \cdot \Delta\delta_{MAX} + t_{su} \quad (2.14)$$

where $\Delta\delta_{MAX}$ is the maximum propagation time of a given circuit implementation of a TEL combinational gate, and corresponds to a specific data input combination (critical path).

In order to have an idea of the timing performances of TEL circuits with the technology scaling progression, it is possible to define a road map for the dependence of δ_{MIN} with the technology; at each technology node the propagation time is estimated to be halved, as well as the setup time and thus δ_{MIN} (Table 2.4). The values are approximated and do not take into account the parasitic capacitances. The setup time is estimated considering the contamination time of a standard XOR gate.

The calculation of δ_{MIN} is important because poses a constraint on the maximum estimated frequency for the TEL circuit in a given node. According to the value in table, it is possible to design a 65nm TEL circuit with a level of security up to about 200ps, which corresponds to a minimum resolution required for a PAAs measurements setup of at least 5GSample/sec. With a 28nm technology, it is possible to obtain values in the order of 50ps, which correspond to a minimum resolution of at least 20GSample/sec, only usable for expensive measurements setups.

Furthermore, in practical attacks, both sampling noise and electronic noise superimposed on the current samples amplitude must be also considered, therefore the minimum resolution is even greater. Obviously, increasing the number N of logic gates in a combinational path increases also the actual value δ .

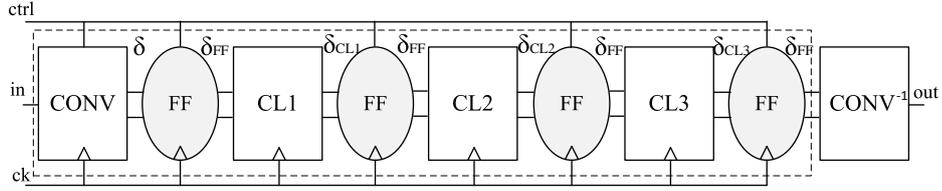


Figure 2.7. A pipelined circuit template in which the information is enclosed inside a time interval δ

Table 2.4. Roadmap of the estimated propagation times for different submicron technologies.

Tech mode [nm]	t_{pMAX} [ps] CMOS IVX2	$\Delta\delta_{MAX}$ [ps] TEL AND/NAND	t_{su} [ps]	δ_{MIN} [ps]
90	50	100	180	380
65	25	50	90	190
45	12	25	45	95
28	6	12	20	48

2.4.4 The fluctuation effect of the delay δ

In this section we define and describe the *fluctuation effect of δ* in TEL circuits, which has been mentioned in previous paragraph. According to the above described timing analysis calculations, a critical point of a TEL implementation is how to guarantee that the dynamic delay δ could be controlled inside the circuit. Indeed, the lack of synchronization between the differential signals and the lack of synchronization between a pair of differential signals and another one, are critical issues for the correct functionality of TEL circuits. More specifically, the value of δ fixed by a designer may suffer on random variations which depend on how the circuit is implemented. This phenomenon is just the fluctuation effect.

As seen in previous paragraphs, the TEL data encoding is characterized by two asynchronous phases, the evaluation and the postcharge. Combinational circuits are characterized by an asymmetric evaluation network due to the asymmetry of the logic function itself; for this reason, the differential signals have different propagation times, as visible in the pipeline model represented in Fig. 2.8.

We refer to the actual delay at the output of a combinational logic as δ_F . The output delay δ_F depends on the propagation time of the gates of the network, which, in turns, depends on the topology of the gates. In other words a fixed delay δ between a complementary pair at the input of a gate is mapped into a non-constant delay δ_F between the complementary pair at the output. This variation of the value of the dynamic delay can result into a positive propagation time ($\delta_F < \delta$) or negative ($\delta_F > \delta$). The fluctuation effect is potentially dangerous for the circuit when the variation is negative, because this means that the output delay δ_F is greater than the original, leading to a negative propagation time, which cannot be acceptable as stated by the timing constraints (i.e. point 1. in Sec 2.4.3).

As asserted by the point 2. in Sec 2.4.3, in TEL circuits the minimum allowable

value of δ_F is set by the setup time of the flip-flops, whereas the maximum value is set by the level of security chosen for the entire circuit (i.e. lowering the maximum δ increases the resolution required for the attacking measurement setup). Therefore TEL combinational circuits must be designed in order to avoid positive variations at the output of the path for not compromising the timing requirements of the circuit, but at the same the nominal value cannot be increased beyond the limit δ fixed for security requirements. For this purpose, it must be guaranteed that the delay δ_F does not increase randomly or uncontrollably at the output of each gate in order to avoid fluctuation, otherwise the timing constraints defined in Sec 2.4.3, in particular the first one, is not met.

Basically the fluctuation effect in TEL combinational networks is due to the asymmetry of the propagation times of the signal along the network. As it will be shown in next chapter for the case of the iDDPL circuit template, this phenomenon can be reduced if the gates are adequately designed. In particular, there are three main reasons which influence the balance of a logic gate:

1. the early evaluation of the data;
2. the different values of the capacitances at the output nodes;
3. the mismatch variations of the adjacent devices.

The first point is intrinsic to the inherent asymmetry of a combinational logic function. It is due to the fact that the data transition of a gate depends on the arrival times of the signals during the evaluation phase. More specifically, a gate can be activated at the arrival time of a signal, without waiting for the arrival of the other one. It represents a critical issue in the design of secure logic circuits, because can lead to a data-dependent power trace.

The second point is due to the imbalance of the output capacitances, which causes different time constants and propagation times. In modern submicron technologies, the amount of overall capacitance is dominated by the interconnect capacitances, whereas the contribution of the internal parasitic capacitances can be neglected under the assumption of using low area gates (i.e. W and L of the transistors are small). Actually, also the interconnect resistances may impact the propagation times of the differential signals, but in our first order analysis we suppose that the lengths of the wires are drawn as short as possible in order to neglect this contribution with respect to the wire capacitance.

The third point is due to the variations of the parameters of the transistors located close to each other, due to the process spreading, which in the nanometer region can strongly impact the symmetry of a logic cell.

All these effects will be analyzed in the following chapter dedicated to the design of iDDPL circuits.

2.4.5 Second order effects: transient leakage

The imbalance of the output capacitances on the differential wires may have a critical impact not only on the timing constraints of the circuit, but also on the leakage model defined in Sec. 2.4.2, in particular if combined to the PSN model, as discussed in Sec. 2.2.1.

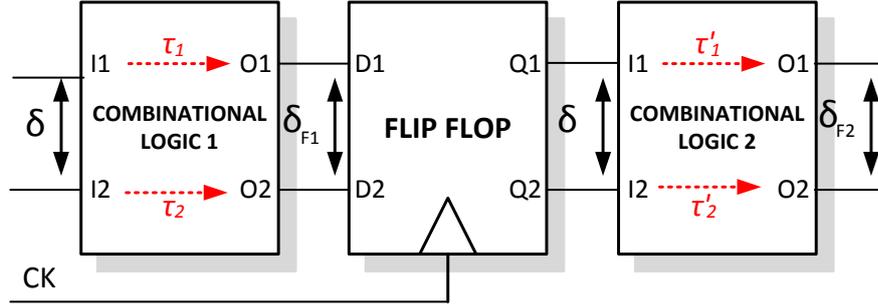


Figure 2.8. Variation of the delay δ along combinational logics before and after a register.

Indeed, the transient effects of the current traces due to the electrical mismatches in the circuit, for example the charge/discharge settling times of the output capacitances, are not taken into account by the model described in Sec. 2.4.2. When the load capacitance is charged by the output inverter of the Domino logic in Fig. 2.6, the current peaks of the current trace have a transient duration which depends just on the value of the single capacitances. If the output capacitances are mismatched, the current peaks of evaluation and postcharge have different transient durations.

From a security point of view, the current samples associated to the transient duration of a peak are relevant, because contain information regarding the processed datum. These means that, if we consider for example the frequency patterns of the current traces, they contain data-dependent components at higher frequencies. The presence of off-chip capacitances, omnipresent on the PSN of digital circuits, should help to filter out these peaks and mitigate the transient leakage.

However the main issue during the evaluation of the security of a specific circuit is: **to what extent the decoupling capacitance in the PSN reduces the transient leakage?** Earlier security metrics adopted to assess the data-dependence of a logic gate, like the NED, give an estimation of the ability of a circuit to balance the energy in a clock cycle by integrating the current traces in the period, doing the implicit assumption on the presence of a low pass filter. Furthermore, also the first order model described in Sec. 2.4.2 refers to the average energy. However, several papers demonstrated that depending on the device under attack, an attacker can even remove the off-chip capacitances and exploit these mismatches for attacking the circuit, as for example in [48]. This issue has been already treated in Sec. 2.2.2: balancing the energy in a cycle is not sufficient for enhancing the resistance of a circuit against PAAs, and NED can overestimate the actual level of security.

Consider for example the simple testbench in Fig. 2.9, in which the instantaneous current trace of the TEL inverter is measured considering a MF equal to 3: one capacitance, say C_{L1} , is fixed to 1fF, the other one, say C_{L2} , is equal to 3fF. The model of the PSN is simple: a capacitance C_F together with a source resistor R_S (100Ω), which act as low pass filter for the current drawn from the source generator. The clock frequency is chosen equal to 10MHz, which is typical for portable cryptographic applications, whereas the V_{DD} voltage is equal to 1V and the time window δ to 500ps.

Simulations are repeated using different values for the capacitance C_F (no ca-

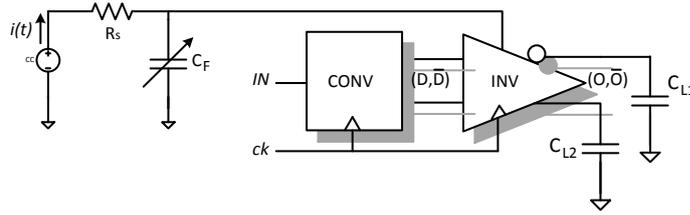


Figure 2.9. Testbench for the simulations of the TEL inverter cell with an unbalanced load and a variable RC filter on the PSN.

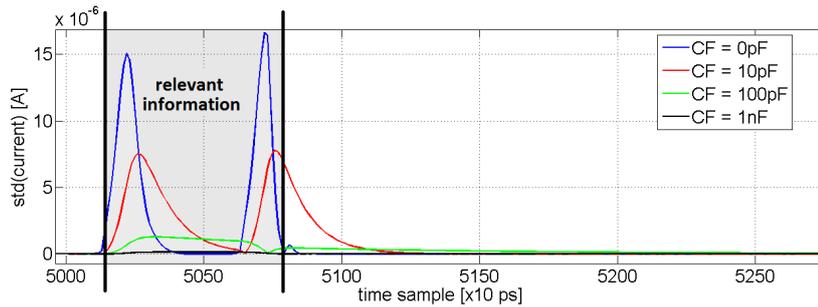


Figure 2.10. Distribution of the current samples during the evaluation phase for the unbalanced TEL inverter, with a variable filtering capacitance.

capacitance, 10pF, 100pF, 1nF). Current traces in the two cases $(1,0) \rightarrow (0,1)$ and $(0,1) \rightarrow (1,0)$ are then exported from Cadence Spectre, obtaining the sample sequences $i_{0 \rightarrow 1}[k]$ and $i_{1 \rightarrow 0}[k]$, with $k = 1, 2 \dots T$. The sampling period is equal to 1ps, which results in a number of samples $T = 100k$ per cycle. The filtered instantaneous current traces reported in Fig. 2.10 are the standard deviation of the sequences $i_{0 \rightarrow 1}[k]$ and $i_{1 \rightarrow 0}[k]$ in the evaluation and postcharge phases (i.e. when clock is high) for different values of C_F . It must be pointed out that the standard deviation of the current traces for the different data commutations is directly linked to the exploitable signal power P_{sig} , as indicated by Eq. 2.1.

From the figure, it can be seen that when no capacitances are inserted, the peaks are clearly visible and separated of about 500ps (blue curve); the rising/falling times are short, in accordance to the drive strength of the cell, and almost all the relevant samples are distributed within the security area in the window δ . Already with a relatively small decoupling capacitance of 10pF, the peaks extend outside the security area, beyond the nominal interval δ , creating the above described transient leakage (red curve), with several relevant points falling inside the dangerous area. Increasing C_F up to 100pF, the transient leakage is integrated in time, but it is still visible outside the interval δ (green curve), even if with a smaller amplitude. Finally, with a value of 1nF, the transient leakage is almost completely filtered and the standard deviation of the traces is completely flattened.

Even if the energy (i.e the area under the curve) is always the same and NED is almost constant (Table 2.5), the relevant period extends beyond δ as a function of the filtering capacitance. If we consider a pipelined circuit as that shown in Fig. 2.7, the contribution to the transient leakage is multiple and adds up along

Table 2.5. NED as a function of the capacitance C_F .

C_F	NED
0	0.855
10	0.807
100	0.869
1000	0.871

the combinational path, creating a current trace with an unpredictable outstanding variability outside the defined security area. The strength of power analysis attacks is to exploit this variability on the instantaneous power consumption, therefore the assumption done on the security of TEL circuits falls down if these transient effects are not adequately controlled.

2.4.6 Energy balancing and timing enclosing properties

The analysis done in previous paragraphs, more specifically the first order power consumption model of a TEL circuit and the description of the second order transient effects, poses the attention on two main issues regarding a TEL circuit, which are different but strictly linked.

The first issue is that the TEL encoding helps to overcome the data-dependence of the power consumption in presence of unbalanced load by guaranteeing that always the same transitions are done during a clock cycle. The second issue is that the timing mismatches due to the transient effect, which in turns depends also (but not only) on the differential capacitances, must be eliminated or at least limited instead of demanding this issue to the PSN filter. These two issues reflect into two important properties of the TEL circuits which must be guaranteed during the design of the circuit:

1. The average current on a clock cycle is balanced (property of *energy balancing*).
2. The instantaneous current exhibits a limited time interval in which the leakage could be potentially detectable (property of *timing enclosing*).

The first condition states that the total amount of energy provided to and subtracted from the output load is always constant irrespective of the values of the single capacitances; this condition is necessary but not sufficient to ensure that the instantaneous power consumption is data-independent and the circuit is protected for the whole elaboration period.

The second condition enforces the first one and, if satisfied, allows to hide the leakage in a very narrow time interval that cannot be detected by the adversary, making ineffective any timing mismatch. If the first condition is inherent to the TEL principle, the second condition must be adequately assessed by a correct design, as it will be described in next section.

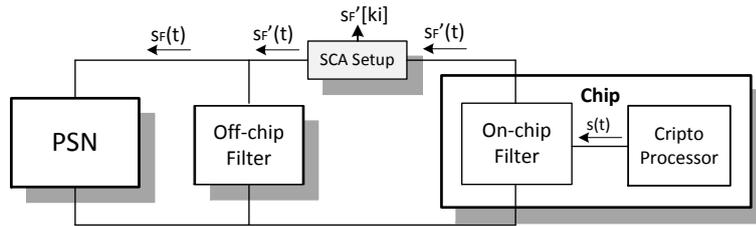


Figure 2.11. PAAs scenario for a TEL circuit, with the insertion of an on-chip filter for removing the high frequencies components directly at layout level.

2.5 A balancing act: frequency analysis of the current traces

According to the issues analyzed in previous sections, a TEL circuit can efficiently balance the instantaneous power consumption if and only if all the electrical unbalances resulting from capacitive and timing mismatches are mitigated. In this paragraph we present a novel methodology which can be adopted in combination with the usage of TEL data encoding in order to meet these requirements.

As described in previous section, the main issue is to eliminate, or at least to dramatically reduce, the transient effects arising in TEL circuits in order to meet the timing enclosing property. The design methodology we propose is based on the insertion of a on-chip filter whose purpose is to remove the high frequencies components of the transient leakage depicted in Fig. 2.10. This solution guarantees that the high frequencies are adequately filtered already at layout level, irrespective of the presence of an off-chip filter, in accordance to the scenarios depicted in Fig. 2.1.

2.5.1 Insertion of an on chip filter in a TEL circuit

With reference to Fig. 2.11, the idea is to insert a generic on-chip filter block directly inside the chip. In this scenario, the best situation for an attacker is when he/she has direct access to the pin of the package of a chip, under the hypothesis of having removed the off-chip capacitances as shown in the case (b) of Fig. 2.1. This way the attacker can measure **only** the trace $s'_F(t)$, which is already low pass filtered; thus, provided that the on-chip filter is adequately designed, the transient effects of the internal signal $s(t)$ cannot be detected outside the package of the chip.

This methodology represents an optimization which, if combined to the adoption of TEL data encoding allows to mitigate the effect of electrical mismatches and to efficiently flatten the instantaneous current irrespective of any influence of the external environment, like the peripherals of the chip or the strength of the measurement setup, as it will be shown in next sections.

The situation described in Fig. 2.11 implicitly relies on the presence of an additional back-end design step, based on the insertion of a layout filtering. This step can be efficiently implemented for example by inserting some capacitance in the layout of the chip. It must be pointed out that the presence of on-chip decoupling capacitors is

already implicit in the VLSI design, due to the fact that during the digital back-end flow some decoupling capacitors are always inserted by the automatic place and route processor in order to remove the noise on the internal power supply lines and protect signal integrity in specific chip areas.

Therefore polysilicon capacitances are available in common digital libraries as macrocells which are automatically inserted by the compiler. However they typically have a limited capacitance per area unit (in the order of some fF/um²), and in a standard design flow their purpose is to guarantee the functionality of the circuit, without any requirements regarding security issues. The presence of a specific block in Fig. 2.11 indicates that a minimum amount of on-chip capacitance must be guaranteed in order to filter off the transient leakage and make TEL circuit effective in balancing the instantaneous power consumption.

2.5.2 A new frequency-based metric

In this section we provide a method to estimate the minimum bandwidth of the filter in Fig. 2.11 during the design phases of the circuit. We use the *Fast Fourier Transform (FFT)* to deduce information on the energy distribution of the current traces in the frequency spectrum in order to understand which are the critical frequencies with respect to security issues. Previously published works exploited FFT as a novel leakage source, and novel PAAs based on the frequency domain have been also presented [73][110]. Following the results in [120], where authors propose a leakage frequency model to improve the strength of SCAs by selectively filtering the traces of a synchronous design, in this paragraph we use the properties of FFT to define a general metric to assess the leakage distribution during the design steps.

Consider again the results of Sec. 2.4.5. With reference to the testbench in Fig. 2.9, we measured the non-filtered current traces for the two data transitions and calculated their variance. Simulations were repeated for different values of C_{L2}, from 1fF (MF = 1, perfect balance) and 4fF (MF = 4, extreme unbalance), with steps of 1fF.

We call as \mathbf{FFT}_0 and \mathbf{FFT}_1 the one-dimensional vectors containing the points of the FFT of the current traces associated to the two possible input data (0, 1) and (1, 0), respectively. These vectors contain F points. The current traces have been exported with a sampling period of 10ps ($f_S = 100GS/s$) and according to the Nyquist condition the maximum frequency of the FFT is 50GHz. The number of points F is around 2M, which leads to a resolution of about 50kHz in the frequency domain.

We intentionally chose a conspicuous number of points and a very high resolution for the FFT in order to have the best precision as possible in the calculation, in accordance to the capability and the speed of the calculator (i.e. Matlab). The squared absolute value of the difference of the FFTs is:

$$\Delta FFT = |\mathbf{FFT}_0 - \mathbf{FFT}_1|^2 \quad (2.15)$$

The plot of ΔFFT (Fig. 2.12), calculated for different values of MF, provides some useful information regarding the leakage distribution of the circuit. First, there is a flatten bandwidth in the energy deviation plot, equal to about -120dB. This

value is constant irrespective of the amount of output unbalance. Then, beyond the frequency $f_0 \approx 1MHz$, the plots increase and are not more superimposed, and some lobes at frequencies multiple of 1GHz are visible.

This frequency is related to $\delta = 500ps$, as it will be analytically demonstrated in next paragraph, and at this frequency the maximum amount of leakage due to the capacitive mismatch is concentrated: higher the output unbalance, higher the transient effects on the current peaks, higher the lobes of the plot, as visible in Fig. 2.12.

In order to remove the transient leakage due to the capacitive unbalance, the on-chip filter must be able of low pass filtering all the frequencies beyond the cutoff frequency f_0 of the TEL circuit. With the same setup of Fig. 2.9, where we assume a fixed input resistance R_S equal to 100Ω , the minimum value for the on-chip capacitance can be estimated as:

$$C_F^{opt} = \frac{1}{2\pi R_S f_0} \approx 1.6nF \quad (2.16)$$

which is in accordance with the value of 1nF found with transient simulations (Fig. 2.10): if a capacitance lower than C_F^{opt} is used, the on-chip filter is not able to completely remove the high frequencies components due to the mismatch, and some relevant samples fall outside δ . Thus, we claim that the PAAs resistance of the circuit must be adequately assessed also in the frequency domain in order to understand if the timing enclosing property in the time domain is really satisfied.

In order to have a fair comparison, the same set of simulations and the same analysis have been executed for a SABL inverter; the plot of the frequency deviation for different MFs is depicted in Fig. 2.13.

Unlike the case of TEL, in a SABL circuit, which is based on a purely synchronous evaluation, there is no possibility to identify a cutoff frequency. The energy deviation strongly depends on the capacitive unbalance also at low frequencies, and there is no possibility to remove the data-dependent leakage by low passing the traces. Note that already for a moderate mismatch (MF = 2, red curve) ΔFFT is in the order of -80dBA at low frequencies, which is 80db higher than the balanced case and 40dB higher than the leakage of TEL considering the same MF.

The metric in Eq. 2.15 can be generalized to a more general case of N input vectors. We define the *Frequency Energy Distribution (FED)* as the one-dimensional vector of the variances of the frequency samples at the discrete frequency f of the FFTs of all the possible current traces N .

$$\mathbf{FED} = [\sigma_1 \ \sigma_2 \ \dots \ \sigma_F] \quad (2.17)$$

$$\sigma_f = \left[\frac{1}{N} \sum_{i=1}^N \sqrt{[\overline{FFT}[f]^2 - FFT[f]_i^2]} \right]^2 \quad (2.18)$$

with $f = 1, 2 \dots F$. The one dimensional vector $\overline{FFT} = [\overline{FFT}[1] \ \overline{FFT}[2] \ \dots \ \overline{FFT}[F]]$ contains the averages of the points of the FFT; a sample $\overline{FFT}[f]$ is calculated as:

$$\overline{FFT}[f] = \frac{1}{N} \sum_{j=1}^N FFT_j[f] \quad (2.19)$$

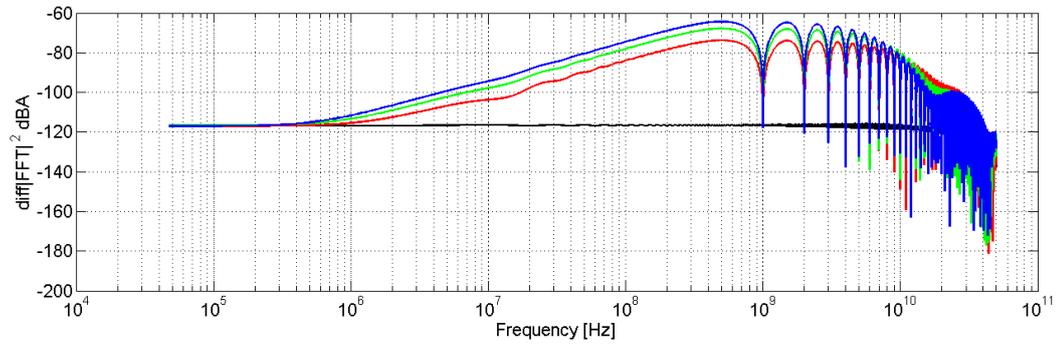


Figure 2.12. ΔFFT vector for the TEL inverter for different value of the mismatch factor ($\delta = 500ps$): MF = 1 (black), 2 (red), 3 (green) and 4 (blue).

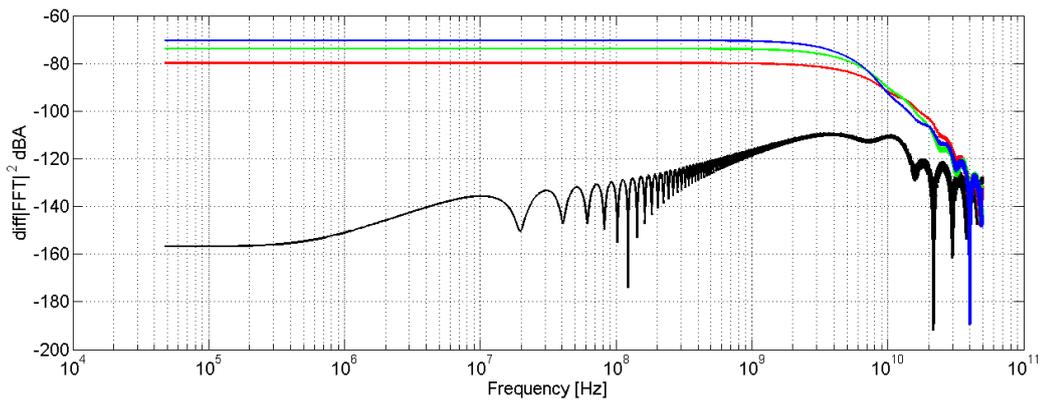


Figure 2.13. ΔFFT vector for the SABL inverter for different value of the mismatch factor: MF = 1 (black), 2 (red), 3 (green) and 4 (blue).

2.5.3 Relation between δ and f_0 in a TEL gate

In this section we calculate the relation between δ and f_0 in a TEL gate. We make use of the Fourier Transform and its properties, and results can be extended considering the FFT of the signals.

The preliminary assumption is that the current peaks at the evaluation and the postcharge phases when traces are not filtered (i.e. the current peaks in Fig. 2.10 in red) can be modeled as the sum of two Gaussian pulses $s_0(t)$ and $s_1(t)$, centered at 0 and δ respectively:

$$s_0(t) = I_0 \cdot e^{-\frac{t^2}{2\sigma_0^2}} + I_1 \cdot e^{-\frac{(t-\delta)^2}{2\sigma_1^2}} \quad (2.20)$$

$$s_1(t) = I_1 \cdot e^{-\frac{t^2}{2\sigma_1^2}} + I_0 \cdot e^{-\frac{(t-\delta)^2}{2\sigma_0^2}} \quad (2.21)$$

I_0 , I_1 , σ_0 and σ_1 are physical parameters which depend on the value of C_{L1} and C_{L2} and the drive strength of the logic cell: higher the output capacitance and lower the drive strength, higher the amplitude and the standard deviation of the pulse. Applying the linearity and the translation property, and considering that the Fourier Transform of a Gaussian pulse is again a Gaussian pulse with an inverse standard deviation, we obtain:

$$S_0(f) = I_0\sqrt{2\pi} \sigma_0 e^{-(\pi\sqrt{2}\sigma_0 f)^2} + I_1\sqrt{2\pi} \sigma_1 e^{-(\pi\sqrt{2}\sigma_1 f)^2} \cdot e^{-j2\pi\delta f} \quad (2.22)$$

$$S_1(f) = I_1\sqrt{2\pi} \sigma_1 e^{-(\pi\sqrt{2}\sigma_1 f)^2} + I_0\sqrt{2\pi} \sigma_0 e^{-(\pi\sqrt{2}\sigma_0 f)^2} \cdot e^{-j2\pi\delta f} \quad (2.23)$$

In order to extrapolate a relation between δ and f_0 , we calculate the absolute value of the difference of the Fourier Transforms; after some calculations:

$$|\Delta S(f)| = |S_1(f) - S_0(f)| = 2\sqrt{2\pi} |\sin(\pi\delta f)| \cdot \left| I_1\sigma_1 e^{-(\pi\sqrt{2}\sigma_1 f)^2} - I_0\sigma_0 e^{-(\pi\sqrt{2}\sigma_0 f)^2} \right| \quad (2.24)$$

The last factor in Eq. 2.24 represents the difference of the Gaussian pulses when there is no delay; in the ideal case of $MF = 1$ and symmetric cell, we have $I_0 = I_1$ and $\sigma_0 = \sigma_1$, thus $|\Delta S(f)| = 0$ at each frequency, independently from δ . However as previously discussed, in submicron technologies it is hard to guarantee a perfect balance between C_{L1} and C_{L2} , therefore we consider the realistic case of $MF \neq 1$. From Eq. 2.24, it can be noted that the dependence of $|\Delta S(f)|$ on δ is sinusoidal, and there is an infinite number of local minima and maxima, which is just deducible from the lobes in Fig. 2.12. If we consider $\delta \neq 0$, $I_0 \neq I_1$ and $\sigma_0 \neq \sigma_1$, we have:

$$\max |\Delta S(f)| = 2\sqrt{2\pi} \left| I_1\sigma_1 e^{-(\pi\sqrt{2}\sigma_1 f)^2} - I_0\sigma_0 e^{-(\pi\sqrt{2}\sigma_0 f)^2} \right| \iff \sin(\pi\delta f) = 1 \quad (2.25)$$

$$f_m^{\max} = \frac{1 + 2m}{2\delta}, \quad m \in \mathbf{Z} \quad (2.26)$$

$$\min |\Delta S(f)| = 0 \iff \sin(\pi\delta f) = 0 \quad (2.27)$$

$$f_m^{\min} = \frac{m}{\delta}, \quad m \in \mathbf{Z} \quad (2.28)$$

The frequency pattern of $|\Delta S(f)|$ shifts towards the right (left) part of the frequency axis if δ decreases (increases). Fixed $m = m'$, $f_m'^{(\min)}$ and $f_m'^{(\max)}$ have an

inverse relation with δ . The first minimum and the first maximum can be found for $m = 0$ at the frequencies $f_0^{(min)} = 0$ and $f_0^{(max)} = \frac{1}{2\delta}$. For the case $\delta = 500ps$, the values are in accordance to the plot in Fig. 2.12. The cutoff frequency f_0 is located in the frequency range bounded by $f_0^{(min)}$ and $f_0^{(max)}$ where the function is monotonically decreasing, thus also f_0 has an inverse dependence with δ . Relaxing the condition in Eq. 2.28, we obtain:

$$\min |\Delta S(f_0)| \approx 0 \iff \sin(\pi\delta f_0) \approx 0 \iff f_0 \ll \frac{1}{\pi\delta} \quad (2.29)$$

as expected. This relation is experimentally confirmed by repeating the simulations of previous section with different values of δ . The plot of f_0 as a function of δ is reported in Fig. 2.14 for values of δ in the range $100ps \div 5ns$.

As shown in Fig. 2.15 and Fig. 2.16 for the cases of $\delta = 100ps$ and $\delta = 5ns$ respectively the diagram of ΔFFT shifts in the frequency axis. The domain of the curve in Fig. 2.14 is given by the minimum (i.e. δ_{MIN} defined in Eq. 2.14) and the maximum value of δ (i.e. $\delta_{MAX} = \frac{T_{CK}}{2}$) in a given technology and for a certain clock frequency. If δ goes to δ_{MAX} , the TEL gate works similarly as the SABL gate because the postcharge phase elapses, and the cutoff frequency f_0 goes to 0, invalidating the benefits of the time-enclosed encoding.

We point out that modeling a current peak as a Gaussian pulse neglects the tail lobes in the spectrum, which instead must be considered for example in the case of many logic gates switching at the same clock cycle. In this case, the current trace is composed of several peaks and the pattern has not a Gaussian shape (see Fig. 2.20 in next section). Furthermore, in each current trace the static power consumption is also superimposed to the dynamic peaks. In a symmetric gate as a TEL inverter, the static consumption is balanced for both the transitions; the residual leakage in the plot of ΔFFT (in the order of -120dB) is probably due to the numeric error done by the simulator.

2.6 A cryptographic case study

2.6.1 The SERPENT-block

In this and in the next section we perform the security evaluation of our countermeasure, considering a specific cryptographic case study for the simulations, that will be called in the remainder of this work as SERPENT-block.

The SERPENT-block is a circuit which implements a cryptographic primitive of the Serpent block cipher. The SERPENT-block has been designed using an implementation of the TEL family of circuits, the iDDPL style, which will be described more specifically in the next chapter. A SABL implementation of the same data-path has been also built in the SERPENT-block in order to have a term of comparison for the simulation. In this paragraph we describe the architecture of the crypto-circuit under analysis, as well as the power consumption model of the pipeline which has been used to implement CPA attacks on the circuit. Furthermore, in this paragraph we describe how the above described design methodology could be applied to enhance the level of security of the circuit.

The SERPENT-block has been designed and instantiated as a macro-block in a

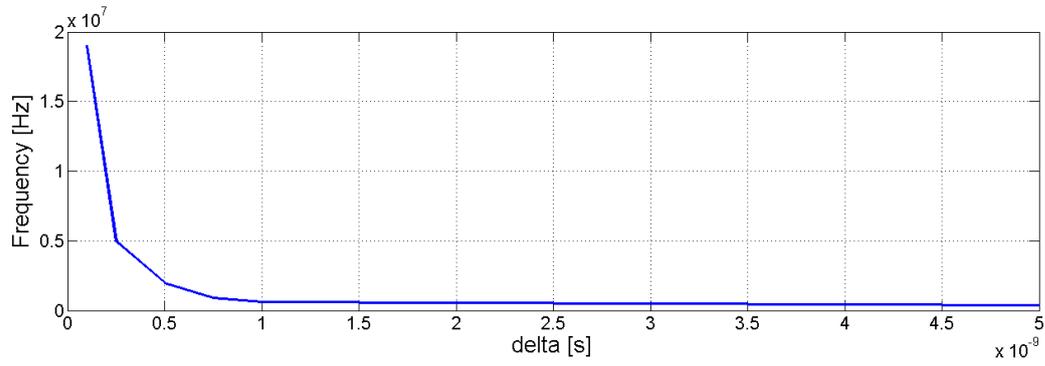


Figure 2.14. Plot of the frequency f_0 as a function of δ .

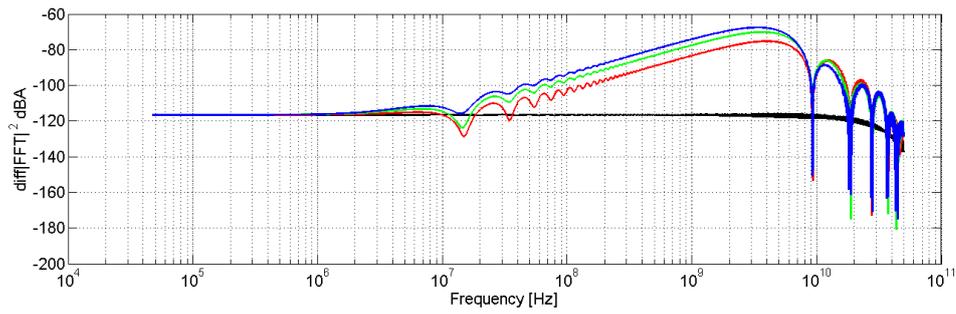


Figure 2.15. ΔFFT vector for the TEL inverter for $\delta = 100ps$

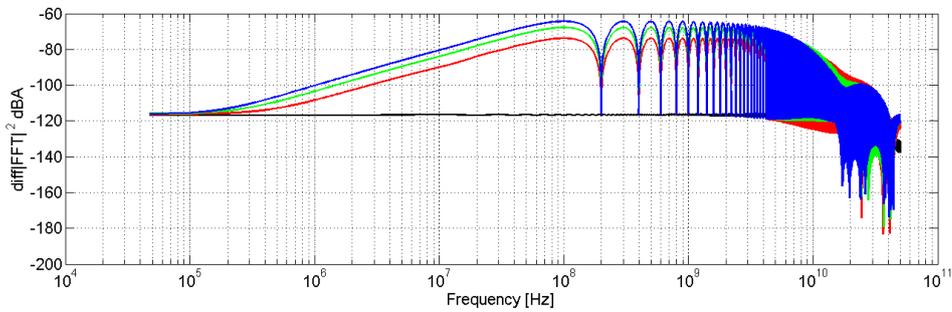


Figure 2.16. ΔFFT vector for the TEL inverter for $\delta = 5ns$

semi-custom digital design flow of a cryptographic ASIC, the *SERPAES* prototype chip. The implementation details of the SERPENT-block, as well as the detailed digital design flow of the SERPAES chip, will be described in Chapter 4.

2.6.2 Description of the architecture of the circuit

The SERPENT-block is a 4-bit cryptographic circuit, which implements a 4-bit slice unit of the *Serpent* block cipher, and has been designed in two versions: one using TEL cells and one using SABL cells. *Serpent* is one of the finalists of the AES contest [8], and it is based on 4x4 S-Boxes, which are widely adopted in recent cryptographic applications thanks to their compactness and efficiency.

Several block ciphers are based on S-Boxes implemented through Look Up Tables (LUTs), in particular for microcontroller or FPGA applications. Anyway LUT-based architectures can be highly inefficient, if we consider for example a 32-bit or 64-bit processor splitting data words into nibbles for LUT-based 4x4 S-Boxes, as those used by *Serpent*, which may require additional special instructions. For this reason we have opted for a fully combinational design of the circuit, which allows to exploit the efficiency of the 4x4 S-Boxes and is more suitable for a hardware implementation.

In Fig. 2.17 a possible generic bit slice implementation of the *Serpent* encryption unit is shown. Each inner round of the algorithm is represented as a two-level pipeline: a nibble of a datum is stored in a 4-bit register, then it is XORed with a nibble of the round key, processed by the S-Box block and finally stored in a 4-bit register. After two clock cycles the datum is ready to be processed by the *LinearTransformation* block. The S-Boxes are built according to their logic function, by using combinational logic gates instead of 4-bit LUTs. The architecture is iterative after the eighth *LinearTransformation* of each inner round.

In the proposed architecture the 32 nibbles of a 128-bit data word are processed in parallel at each clock cycle. Moreover, the architecture is compact and the area occupation is expected to be reasonably restrained. However, we have chosen a single unit of the processor because a full design verification of the entire 128 bit processor would have required a very long time for simulating all possible input vectors at SPICE level. The data path of the circuit is reported in Fig. 2.18.

The choice of reducing the span of the attack to 4-bit words is compatible with the bit slice structure of *Serpent*: if we consider for instance the first round of the encryption, the power consumption of the logic is the sum of the power consumption of 32 identical parallel bit slice units [7], which switch at the same time. Therefore power analysis simulations can be simplified by analyzing the resistance of one of these bit slice units, and considering the other switching circuits as on-chip noise. Then, by exploiting the leakage of the target bit-slice it is possible to recover 4 bits of the key word, and replying the same attacks for the other bit slice units to recover the whole key word, as in a *divide and conquer* strategy.

The circuit in Fig. 2.18 processes a nibble of the 128-bit data word in a two stage pipeline. In the pipeline stages, a 4-bit data word is first converted and stored in a register, then it is XORed with a nibble of the round key, processed by the 4x4 S-Box S0 block and finally stored in an output register. The hardware description of the S-Box S0 was done using the Synopsys Design Compiler, which generated a netlist of combinational gates, and exported into Cadence environment. The data-path

in Fig. 2.18 has been implemented using TEL data encoding, with a relevant time $\delta = 1ns$.

The TEL circuit has been implemented using iDDPL gates. The circuit template has been already presented in Fig. 2.6(a). In next chapter we describe more in detail the iDDPL library adopted for designing the circuit, whereas the implementation details of the circuit are provided in Chapter 4.

2.6.3 Direct analysis of the power model of the pipeline

In this section a fine-grain analysis of the power model of the circuit is executed, in order to build a suitable template of the power consumption. The template relies on the fact that an attacker can characterize a device by building a very accurate model. Ideally, in an unbounded profiling phase (i.e. in the situation in which an attacker can have an infinite number of traces and the electronic noise is removed), it is sufficient to consider all the data transitions which are directly related to a specific key to recover. Cadence simulations are intrinsically noise free, and therefore they are suitable to build a precise power template of the circuit, as discussed in Sec.2.2.1.

With reference to Fig. 2.18, the selected data word at the output of the SBox S_0 , $w = f(IN, KEY)$, represents the target function of the attacks. In power analysis attacks we suppose that the attacker knows the plaintexts and wants to recover the secret key by measuring the power traces of the circuit and at the same time predicting the leakage. The timing diagram of in/out signals and the latency of the processed words at each clock cycle are reported in Fig. 2.19.

The 4-bit input data word which enters in the pipeline at the clock cycle k is indicated with the notation $IN(k)$. The circuit is a two stages micropipeline, in which the word $IN(k)$ is first sampled and converted into the DPL domain $(D, \bar{D})(k)$ at the rising edge of the clock cycle k ; then the datum is sampled at the rising edge of the clock cycle $k + 1$ by the first flip-flop, and during the evaluation phase it is processed by the cryptographic combinational logics, obtaining the target function w ; finally the latter is sampled at the rising edge of the clock cycle $k + 2$ by the output flip-flop and re-converted into the single-rail domain, obtaining the encrypted single-rail datum $OUT(k)$.

The round key is generated by the key scheduler and stored in a register at the same clock cycle k , and XORed to the word in the evaluation phase of the clock cycle $k + 1$. The latency of the pipeline is equal to three clock cycles and this information provides the time span for a single power analysis acquisition. The target function w is the combination between the input word $IN(k)$, processed by the circuit starting from the clock cycle k , and the $KEY(k)$ stored in the register at the same clock cycle:

$$w = f(IN(k), KEY(k)) = S_0(IN(k) \oplus KEY(k)) \quad (2.30)$$

In Table 2.6 the model of the power consumption for each clock cycle is calculated. With reference to the clock cycle k , w_k^j represents a function of the input word $IN(k)$ processed by the logic cell j , whereas d_k^j is the logic state of the nodes of the cell, and the leakage trace $s_k^j = [s_{k1}^j \ s_{k2}^j \ \dots \ s_{kT}^j]$ is the correspondent current trace with

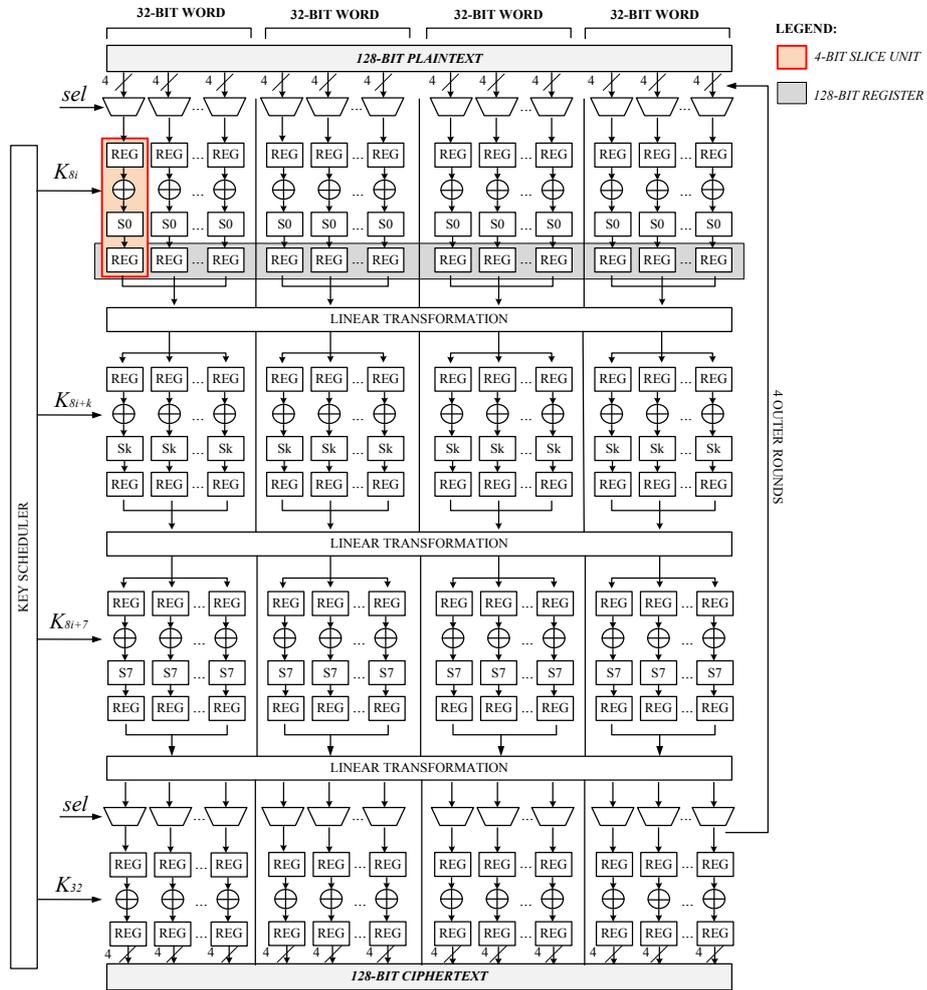


Figure 2.17. A generic bit slice hardware implementation of the Serpent encryption processor.

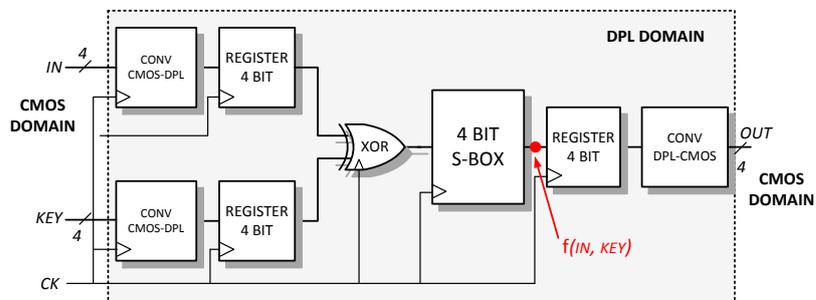


Figure 2.18. Data path of the SERPENT-block using a DPL implementation.

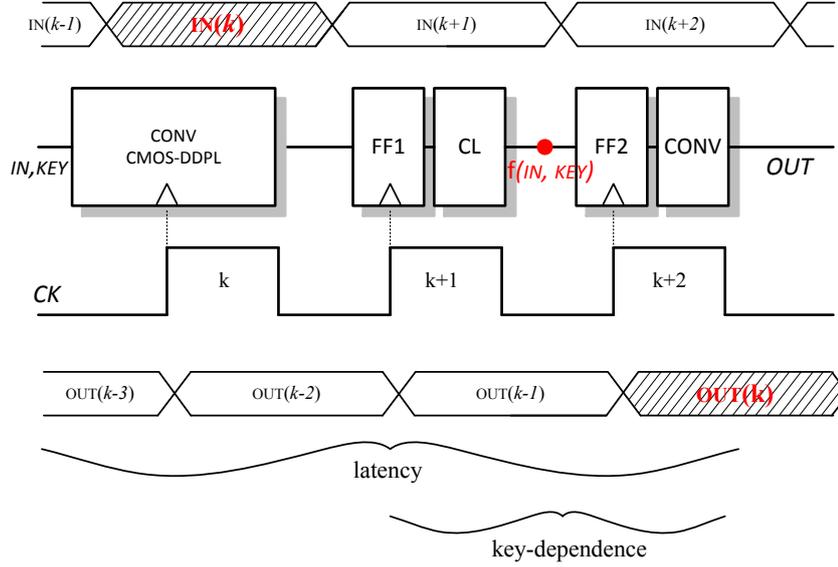


Figure 2.19. Timing diagram of in/out signals and latency of the processed words at each clock cycle of the pipeline.

T time samples. Note that in the pipeline, at the same clock cycle k also the data words $w_{k-1}^{j'}$ and $w_{k-2}^{j''}$, which depend on the input word at the previous clock cycles $IN(k-1)$ and $IN(k-2)$, are processed by the logic cell j' and j'' , respectively, and contribute as sources of noise to the overall leakage s_k during the clock cycle k .

As visible in table, the correlation between the current trace and the target function w is high only when the circuit is processing $KEY(k)$, that is during the second and the third clock cycles. According to the power model in table, the strategy of the attacker is to build a template of the circuit by collecting the noise free traces correspondent to $N = 4^3 = 256$ data commutations, which involve the combinational logic at the clock cycle $k+1$ and the output flip-flop at the clock cycle $k+2$. Therefore, the purpose of a DPL circuit is to reduce the leakage by mitigating the key-dependence detectable in the clock cycles $k+1$ and $k+2$, where the leakage is potentially visible (high correlation).

With reference to the TEL implementation of the circuit, at the falling edge of the clock all internal capacitances of the logic gates are precharged to zero; at the rising edge of the clock, the asserted signals are evaluated and the internal capacitances of the correspondent differential half circuit are discharged, whereas the non-asserted signals are evaluated after δ and the internal capacitances of the other differential half circuits are precharged. Thus, the adversary knows exactly at which time moment any change in the current pattern due to the electrical mismatches can occur, that is after the rising clock edges at the cycles $k+1$ and $k+2$.

Table 2.6. Power model at each clock cycle: the key dependence of the instantaneous power consumption can be detected in the 2nd and the 3rd clock cycles.

Cycle k	Processed words	Leakage s_k	Corr[KEY(k), s_k]
$d_{k-1}^{CONV} \rightarrow d_k^{CONV}$	$w_k^{CONV} = \mathbf{IN}(k)$	s_k^{CONV}	<i>low</i>
$d_{k-2}^{FF1} \rightarrow d_{k-1}^{FF1}$	$w_{k-1}^{FF1} = \mathbf{IN}(k-1)$	s_k^{FF1}	<i>low</i>
$d_{k-2}^{CL} \rightarrow d_{k-1}^{CL}$	$w_{k-1}^{CL} = S_0[\mathbf{IN}(k-1) \oplus \mathbf{KEY}(k-1)]$	s_k^{CL}	<i>low</i>
$d_{k-3}^{FF2} \rightarrow d_{k-2}^{FF2}$	$w_{k-2}^{FF2} = S_0[\mathbf{IN}(k-2) \oplus \mathbf{KEY}(k-2)]$	s_k^{FF2}	<i>low</i>
Cycle $k+1$	Processed words	Leakage s_{k+1}	Corr[KEY(k), s_{k+1}]
$d_k^{CONV} \rightarrow d_{k+1}^{CONV}$	$w_{k+1}^{CONV} = \mathbf{IN}(k+1)$	s_{k+1}^{CONV}	<i>low</i>
$d_{k-1}^{FF1} \rightarrow d_k^{FF1}$	$w_k^{FF1} = \mathbf{IN}(k)$	s_{k+1}^{FF1}	<i>low</i>
$d_{k-1}^{CL} \rightarrow d_k^{CL}$	$w_k^{CL} = S_0[\mathbf{IN}(k) \oplus \mathbf{KEY}(k)]$	s_{k+1}^{CL}	high
$d_{k-2}^{FF2} \rightarrow d_{k-1}^{FF2}$	$w_{k-1}^{FF2} = S_0[\mathbf{IN}(k-1) \oplus \mathbf{KEY}(k-1)]$	s_{k+1}^{FF2}	<i>low</i>
Cycle $k+2$	Processed words	Leakage s_{k+2}	Corr[KEY(k), s_{k+2}]
$d_{k+1}^{CONV} \rightarrow d_{k+2}^{CONV}$	$w_{k+2}^{CONV} = \mathbf{IN}(k+2)$	s_{k+2}^{CONV}	<i>low</i>
$d_k^{FF1} \rightarrow d_{k+1}^{FF1}$	$w_{k+1}^{FF1} = \mathbf{IN}(k+1)$	s_{k+2}^{FF1}	<i>low</i>
$d_{k+1}^{CL} \rightarrow d_{k+2}^{CL}$	$w_{k+1}^{CL} = S_0[\mathbf{IN}(k+1) \oplus \mathbf{KEY}(k+1)]$	s_{k+2}^{CL}	high
$d_{k-1}^{FF2} \rightarrow d_k^{FF2}$	$w_k^{FF2} = S_0[\mathbf{IN}(k) \oplus \mathbf{KEY}(k)]$	s_{k+2}^{FF2}	high

2.6.4 Estimation of the cutoff frequency f_0 of the circuit

The next step is the characterization of the leakage of the circuit by collecting the current traces related to all the possible 256 input combinations. It must be noted that this step must be performed before the layout of the chip, considering different mismatch factors on the differential wires in order to model the effect of the interconnect wires. The clock frequency is chosen to be equal to 10MHz which is typical for smart-cards applications, the V_{DD} voltage to 1V and the time window δ of the TEL circuit to 1ns. We have inserted at the output of each logic gate two capacitances to simulate the interconnect unbalance. We have collected the current traces from the case of negligible unbalance ($MF \approx 1$) to the case of high unbalance ($MF = 3$). As an example, in Fig. 2.20 the superimposed current traces when $MF = 3$ in the evaluation/postcharge phases are shown; as expected, several peaks can be identified in the instantaneous trace, due to the presence of several logic gates switching at the same time. Moreover, also a certain amount of static power.

We have repeated the frequency analysis previously presented for the TEL inverter gate. In order to take into account all possible inputs, we have used the *FED* metric defined in Eq. 2.17 and Eq. 2.18 to determine the amount of bandwidth required to design the on-chip filter. In this set of simulations, the PSN is modeled as an ideal voltage source and the current drawn by the circuit is sampled with a time resolution of 20ps. This simulation setup is equivalent of gathering measurements on the actual circuit with a sampling frequency equal to 50GSample/sec, which poses a constraint on the maximum bandwidth (equal to 25GHz for the Nyquist's limit). The number of points of the FFT is around 200k, which corresponds to a resolution of about 400kHz. Higher values are outside of the memory of Matlab and cannot be processed. The FED is plotted in Fig. 2.21 for the two cases of low unbalance $MF \approx 1$ and $MF = 3$. Even in this case we decided to use the largest number of points as possible for the FFT.

In Fig. 2.21, at low frequencies the FED is in the order of -80dB, which indicates a higher variation of the static power consumption with respect to the case of a single inverter. The main lobe of the FED is at 500MHz, in agreement with Eq. 2.26,

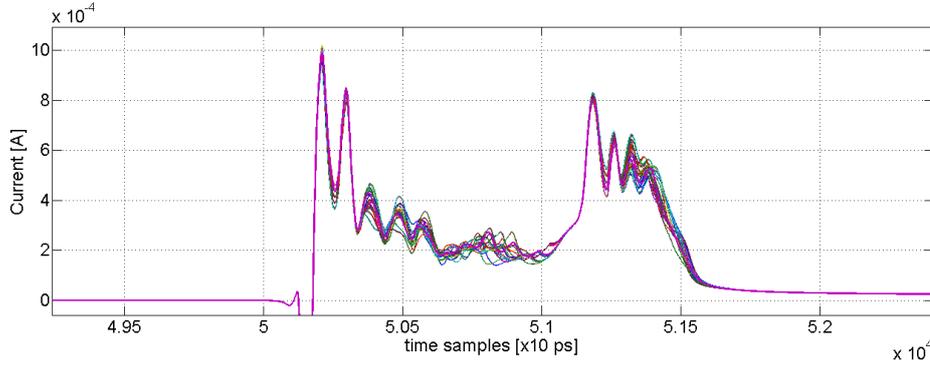


Figure 2.20. Current traces for each of the 256 input combinations of the TEL circuit in the evaluation and postcharge phases of the third clock cycle.

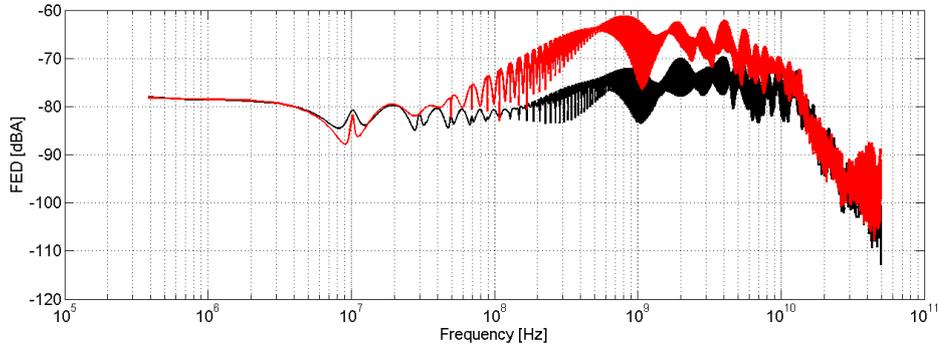


Figure 2.21. FED vector for TEL circuit with low unbalance on the interconnect wires (black curve) and with a maximum unbalance ($MF = 3$) (red curve).

whereas the frequency f_0 is around 30MHz by visual inspection.

Apart from a constant term due to the static consumption and several tail lobes in the FFTs of the traces, the Gaussian model described in Sec. 2.5 still holds and the inverse relation between f_0 and δ depicted in Fig. 2.14 is also confirmed. The static consumption cannot be eliminated by the PSN filter and represents a resilient leakage which does not depend on the dynamic power model and is uncorrelated to the key; thus, f_0 represents just the cutoff frequency of the filter, which must be designed in order to eliminate all the lobes at higher frequencies that correspond to the transient leakage.

2.7 Security evaluation of the TEL circuit

In this section we perform PAAs against a TEL implementation of the previously described architecture. The Pearson's correlation coefficients vector $v = [\rho_1 \rho_2 \dots \rho_T]$ used in standard Correlation Power Analysis (CPA) attacks [17] reveals important information regarding the time instants in which the correlation between current samples and intermediate values is high. For this reason, after having supposed that the adversary knows exactly the relevant time interval, the correlation coefficient vector is used as statistical distinguisher to discriminate the correct key guess.

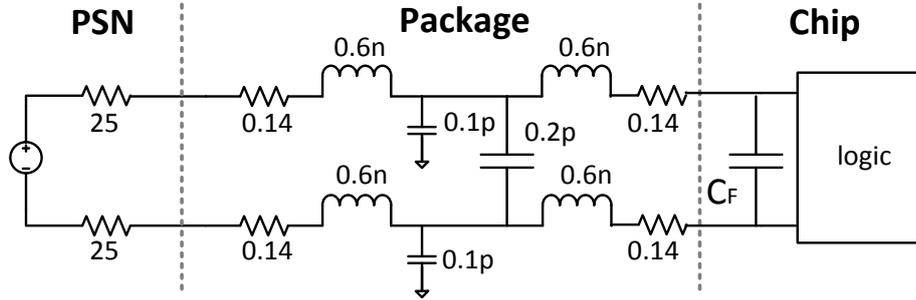


Figure 2.22. Equivalent circuit model for the testbench in Cadence simulations [59]

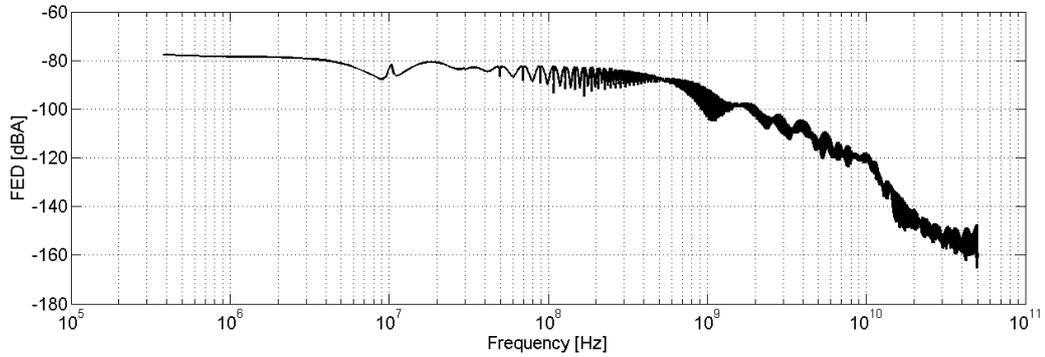


Figure 2.23. FED vector for the TEL circuit calculated after having filtered the current traces ($f_0 = 30MHz$).

2.7.1 Design of the on-chip filter considering chip peripherals

As discussed by authors in [59], in order to have realistic SPICE simulations, a good model for the chip peripherals must be taken into account. Thus, in accordance to the model defined in [59], for the simulation testbench we use the same equivalent circuit which includes the package impedances of the chip as the only sources of impedance that must be included and cannot be removed by an adversary in a non-invasive attack. The effects of the external environment (e.g. socket, cable, etc.) are included in the model of the PSN, which is represented as a generic voltage source with a series resistor $R_S = 50\Omega$. We collected the 256 current traces of the circuit after post layout simulations, using the simulation parameters described in previous section.

According to the pattern of Fig. 2.21 and the impedance model of Fig. 2.22, the capacitance C_F must be at least equal to 100pF in order to obtain a cutoff frequency of about 30MHz. With this value, we have repeated post-layout simulations of the circuit and calculated again the residual FED vector (Fig. 2.23).

The lobes are almost completely removed and the FED is nearly flattened; a residual variation at the multiple of the clock frequency is still visible, but it is below the value at low frequencies.

2.7.2 Area estimation of the countermeasure

As it will be described in next chapters, the Time Enclosed Logic gates have been abutted using a rail-to-rail place methodology, and they have been routed using the Automatic Routing Tool of Virtuoso. The design occupies an active area of about $2.100\mu\text{m}^2$. Through a parasitic extraction we verified that $MF < 3$ after the automatic routing procedure for all the differential interconnect wires, according to the assumption done during the design steps to obtain the plot in Fig. 2.21.

In the simulation model of Fig. 2.22 we have considered the on-chip capacitance as a discrete component. In real cases, it is implemented using the decoupling capacitance cells of the technology library during the design back-end flow. Part of the capacitance can be also implemented by inserting CMOS polysilicon capacitors directly on the V_{DD} global metal wires in the layout. In this case, in order to have a total capacitance equal to 100pF, if we consider a capacitance per area unit of $13\text{fF}/\mu\text{m}^2$, in accordance to the specifications of the 65nm technology we have chosen, an area of about $7.700\mu\text{m}^2$ is required. The overall area estimation of the TEL chip with the polysilicon capacitances is of about $10.000\mu\text{m}^2$.

We point out that this result is specific for our applications and depends on the technology and on the design. Using more scaled technologies it is possible to obtain lower values for the on-chip capacitance and reduce the area overhead: for instance, with reference to Eq. 2.12 and to Table 2.4, if we consider the same circuit implementation, with a critical path of 8 stages ($N_{MAX} = 8$), using a 28nm technology δ_{MIN} can be estimated equal to about 150ps; thus, using a reasonable value $\delta = 200\text{ps}$ which is five times lower than δ used in the 65nm implementation, and according to the circuit components in Fig. 2.22 the cutoff frequency f'_0 would be equal to $5 \cdot f_0 = 150\text{MHz}$, which can be obtained with $C_F = 20\text{pF}$ and a strong reduction of area penalty.

2.7.3 Evaluation of the leakage of the noise-free current traces

Before mounting CPA attacks against the cryptographic circuit, in this section we investigate the distribution of the leakage of the noise-free current traces of the TEL circuit simulated in Cadence and sampled with an unbounded resolution. As a fair comparison, we have designed a SABL implementation of the same circuit.

As first evaluation criterion, we have calculated the correlation between the Hamming weight of the word w , calculated with the above described power model, and the noise free traces, sampled with a resolution of 10ps, which corresponds to an unrealistic situation of attack setup with a remarkable time resolution of 100GSample/sec. The correlation coefficient plots are depicted in Fig. 2.24 and Fig. 2.25 for TEL and SABL circuit, respectively. The correct key is reported in bold line.

The main difference between the figures is that in the TEL implementation the correlation coefficient of the correct key is high only during a brief time interval (δ) soon after the rising edge of the third clock cycle, and this is an expected result because depends on how we have built this logic style. Instead, in the SABL implementation the correlation coefficient is high during the whole second and third cycle of elaboration, and this was also an expected result because the relevant time

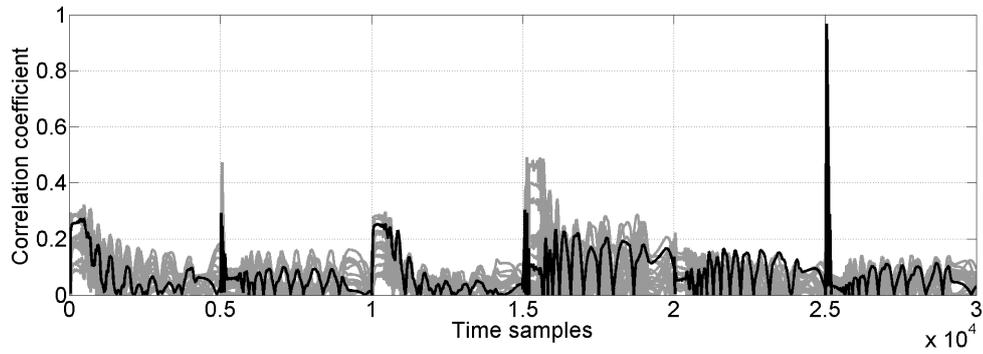


Figure 2.24. Correlation coefficients plot of the 256 simulated traces of the TEL circuits as a function of time; correct key is indicated in bold black line.

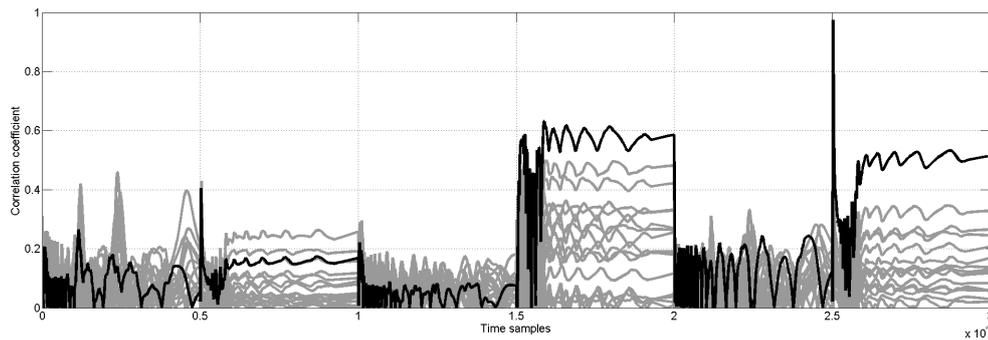


Figure 2.25. Correlation coefficients plot of the 256 simulated traces of the SABL circuits as a function of time; correct key is indicated in bold black line.

of the SABL circuits is just equal to $\frac{T_{CK}}{2}$. These results are in accordance to the predicted power model described in previous paragraph.

In accordance to the plot of Fig. 2.13, the insertion of a low pass filter in the SABL implementation does not help to break the correlation between the instantaneous current and the key because there is a resilient leakage already at low frequencies. In other words, the weakness of SABL circuits which has been detected in the frequency domain causes the extension of the information leakage for the entire relevant time $\frac{T_{CK}}{2}$. On the contrary, the TEL circuits, which are based on a dynamic data encoding in a short relevant time, efficiently hide the information visibility in the time domain thanks to the timing enclosing property, forcing the attacker to use more costly measurement setups in order to detect any leakage.

As a second evaluation criterion, we tested the DPA-resistance of the circuit by investigating the *Point Biserial Correlation (PBC)* coefficient [95], which estimates the correlation between two sets of values when the elements of one group are dichotomous variables (i.e. each bit of the output of the S-Box):

$$PBC = \frac{(M_0 - M_1)}{S_n} \cdot \sqrt{\frac{n_0 n_1}{n^2}} \quad (2.31)$$

The PBC is calculated for each of the four bits at the output of the S-Box for each possible key, and has been used in previous works to assess the DPA resistance of cryptographic implementations [50]. The traces have been subset into two groups according to the value of the selected bit. M_0 (M_1) is the mean value of the leakages when bit is 0 (1), n_0 (n_1) is the number of leakages of the group associated to bit 0 (1), S_n is the standard deviation of the leakages distribution. PBC estimates the level of correlation between the input and the output of a n-bit logic, and is strongly correlated to the circuit architecture: higher is the correlation between a bit and the power consumption, higher is the information leaked by a specific bit-slice.

The plots are reported in Fig. 2.26 and 2.27. We observe that the main part of the leakage is concentrated in the bit-slice of bit-1 and bit-2, where the correlation coefficient is higher in correspondence to the correct key. More important, the correct key can be detected in the SABL implementation for any points in the second and the third evaluation clock cycle, whereas TEL circuits can be hypothetically attacked only during the interval δ , as previously seen.

As expected, a first evaluation of the leakage properties of the TEL circuit with two statistical distinguishers revealed that a higher correlation is detected only during the relevant time δ , where the effect of the capacitive unbalances is very strong. The high frequencies components of the transient leakage which generate relevant current points outside δ have been removed so that the current pattern in the time domain is completely de-correlated from the intermediate value outside the relevant time δ .

2.7.4 Correlation Power Analysis attacks with Gaussian noise

As a final step, we perform the CPA attack procedure on the noisy traces of the SERPENT-block, with a fixed number of input plaintexts.

According to the Gaussian template model, a normally distributed noise has been considered superimposed to the Cadence noise-free traces. In order to perform attacks

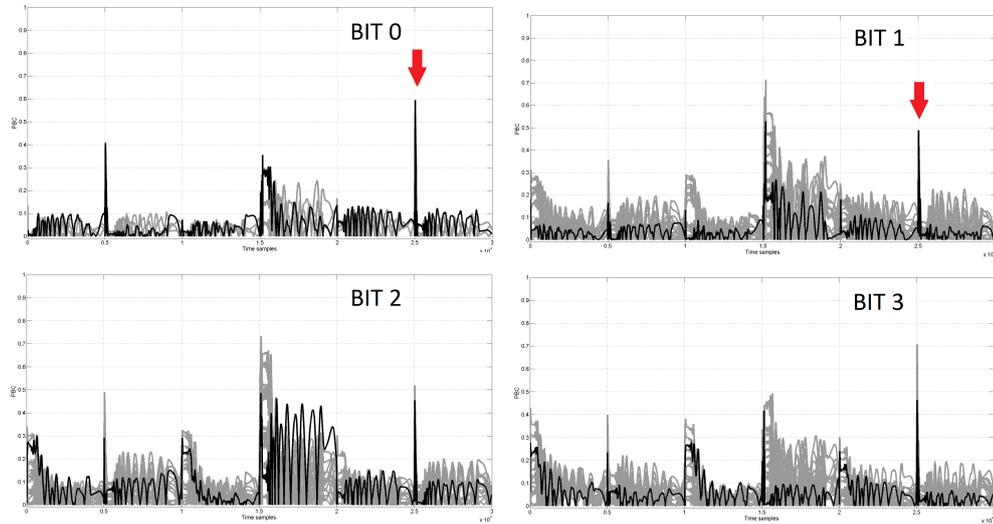


Figure 2.26. PBC of the TEL circuit for the bits of the word at the output of the S-Box (correct key in bold).

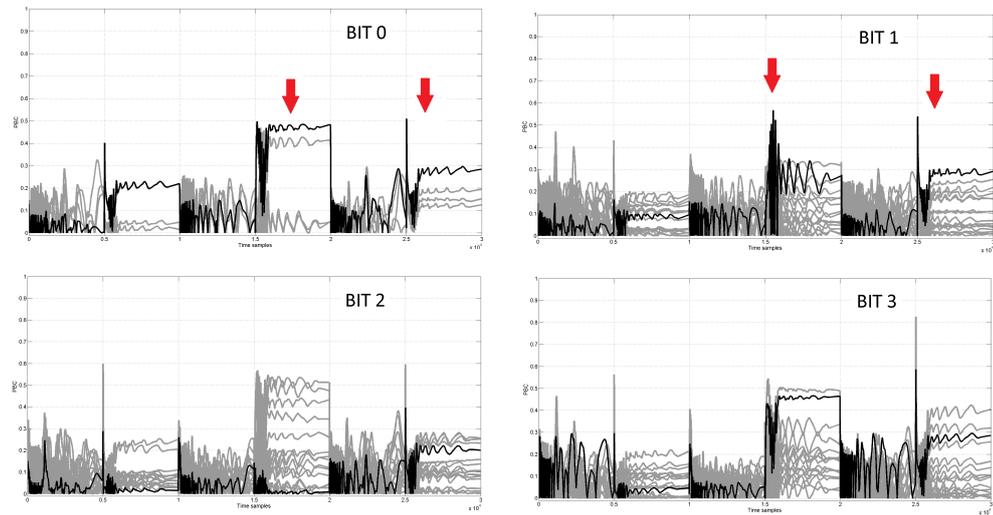


Figure 2.27. PBC of the SABL circuit for the bits of the word at the output of the S-Box (correct key in bold).

in a reasonable time (in the order of some hours), we have neglected the quantization noise due to the AD conversion, which otherwise would have dramatically increased the time of elaboration.

At this point the noise-free current traces have been sampled with a more realistic sampling period of 1ns, in order to emulate the sampling of a basic oscilloscope with a limited time resolution of 1GSample/sec and a bandwidth in the order of few hundreds of MHz, in accordance to the assumptions on the attack model done at the beginning of this chapter. At the same time, the sampling period has been considered to be not constant because of the random sampling imprecision of a real oscilloscope. The sampling time instants are not strictly multiple of 1ns for the presence of a uniformly distributed random jitter in the acquisition, due to the thermal noise, flicker noise, and shot noise contributions inside the oscilloscope. We have considered a peak-to-peak total jitter of about 100ps. These post-processing phases have been implemented using a Matlab script that we have specifically developed for this testbench and that could be also rearranged for other applications. The duration of the attack is in the order of a couple of hours with 1M input plaintexts. After the elaboration, the number of points is equal to 100 for each clock cycle (i.e. 300 for a three cycles elaboration).

For a fixed level of noise, CPA attacks have been mounted with an increasing number of traces, up to 1M. Then, we have calculated the *minimum number of Measurements to Disclose the Key (MTD)* as the cross over point in the correlation coefficient plot. According to the definition of MTD given in [122], MTD is the minimal number of traces needed before the correct key is clearly distinguishable. Attack have been executed on TEL and SABL implementations, increasing the number of input traces step by step.

As done for any other countermeasure implementations tested in simulation, a critical value of Gaussian noise σ_{noise}^{CR} can be determined. It is defined as the maximum value of Gaussian noise beyond which an attacker cannot discriminate the correct key with fixed sources in terms of memory and time. Obviously, lower is σ_{noise}^{CR} , higher is the PAAs resistance of the circuit implementation. The noise is given by the sum of electronic and switching noise, and at simulation level it is summed to the noise-free traces. PAAs have been repeated using different values of σ_{noise} , and at each step the MTD is calculated as a function of σ_{noise} .

In Fig. 2.28 the correlation coefficient plot as a function of the number of input plaintexts is depicted for all the possible keys for the TEL implementation, in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4} > \sigma_{noise}^{CR}$. The correlation coefficient plot as a function of the time samples is showed in Fig. 2.29. From these figures, it is evident that the attack is not successful with the adopted PAAs setup, and the correlation peak detected in the ideal scenario is not more visible.

The same level of noise has been used to mount PAAs against the SABL circuit in order to have a fair comparison between the two implementations. The correlation coefficient plot as a function of the number of input plaintexts and the correlation coefficient plot as a function of the time samples are showed in Fig. 2.30 and 2.31, respectively. From these figures, it is clear that the SABL circuit can be attacked with less than 100k input traces, confirming to have a low resistance to PAAs.

According to the definition of critical noise, we calculate this value for both the TEL and the SABL implementations by repeating PAAs reducing the value of the

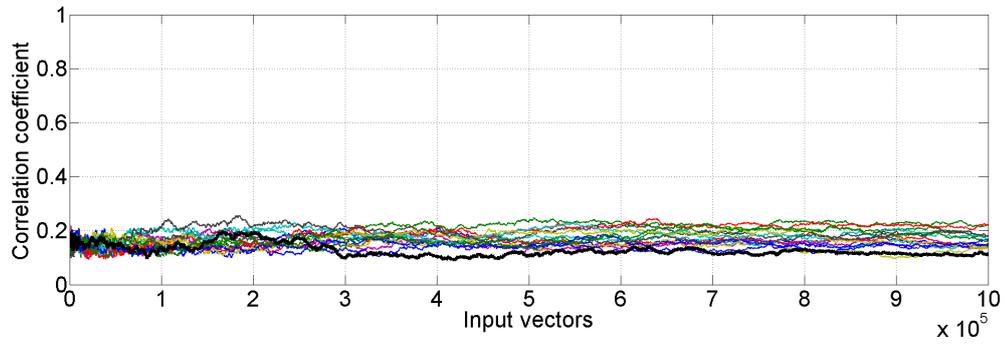


Figure 2.28. Correlation coefficients plot as a function of the number of input plaintexts for the TEL circuit, in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.

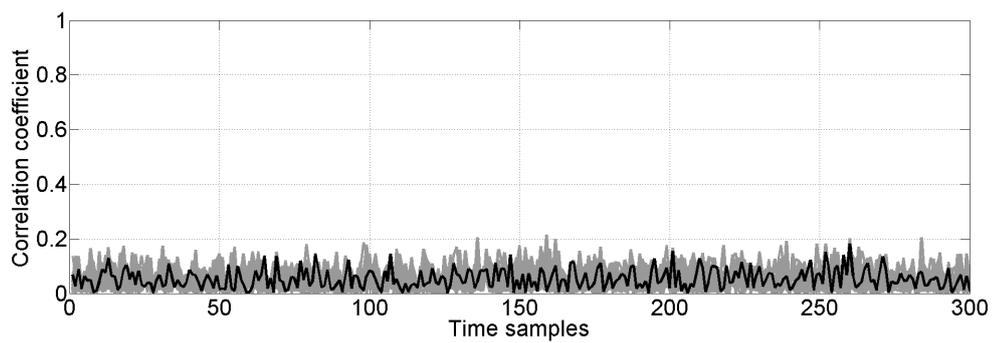


Figure 2.29. Correlation coefficients plot as a function of time for the TEL circuit in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.

Gaussian noise step by step. In Fig. 2.32 the plot of the MTD as a function of noise is reported; the critical noise σ_{noise}^{CR} represents just the value in the x-axis which corresponds to $MTD = 1M$ (i.e., the maximum value of noise beyond which it is not possible to distinguish the correct key with the maximum number of traces).

The most important thing that can be deduced from Fig. 2.32 is that the MTD of the TEL implementation is about one order of magnitude higher than the correspondent SABL implementation. In practical applications, the PAAs resistance of TEL can be also much more greater than SABL, as described in this chapter.

With the aim of recovering the correct key of the TEL circuit, the number of input plaintexts must be much higher than 1M input vectors; the critical noise is in the order of about $2 \cdot 10^{-4}$, which represents a relatively low value of noise if compared for example to the values found in simulations for other logic styles [71].

In order to have a better idea of the level of the noise compared to the intensity of the exploitable signal, in our application the critical noise corresponds to a SNR equal to about 10^{-2} . In practical cases, noise can be even more relevant, therefore the SNR is typically lower than this value.

Obviously, by reducing the level of noise (or alternatively increasing the number of traces), also the TEL implementation could be successfully attacked, as any other countermeasure. This is shown in Fig. 2.33 and 2.34, where the correlation coefficient plot as a function of the number of input plaintexts and the correlation coefficient plot as a function of the time samples in the case of $\sigma_{noise} \approx 10^{-3} < \sigma_{noise}^{CR}$ are showed. In this case, MTD is equal to about 120k traces.

The simulation results presented in this section show unequivocally that the TEL principle, combined to the layout optimization and methodology presented in this chapter, can help to mitigate the electrical mismatches in submicron circuits, enhancing the robustness of the implementation in terms of number of traces for disclosing the key (more than one million) in a PAAs scenario, where the power template of the circuit is perfectly known by the adversary and the correlation coefficient is adopted as statistical distinguisher. If compared to other state of the arts logic styles, like RTZ families which are widely adopted in the context of PAAs, with the same level of noise and number of traces as attack parameters, and under the assumptions that the value of δ is chosen in order to be smaller than the resolution of the attacker, the security level can be increased at least of an order of magnitude in the real case of mismatched design.

2.8 Conclusions

From our point of view, the relevance of the work proposed in this chapter is double: first, a bi-dimensional hardware countermeasure against PAAs is proposed; then a new design methodology, based on the analysis of the frequency distribution of the leakage of the current traces, is presented.

The first important result is that TEL circuits overcome standard synchronous DPLs, as for example RTZ logics, thanks to their hybrid logic data encoding, which makes this logic family inherently tolerant to the electrical mismatches, always present in submicron circuits, and consequently more resistant against PAAs. A back-end optimization is required in order to remove the high frequencies components

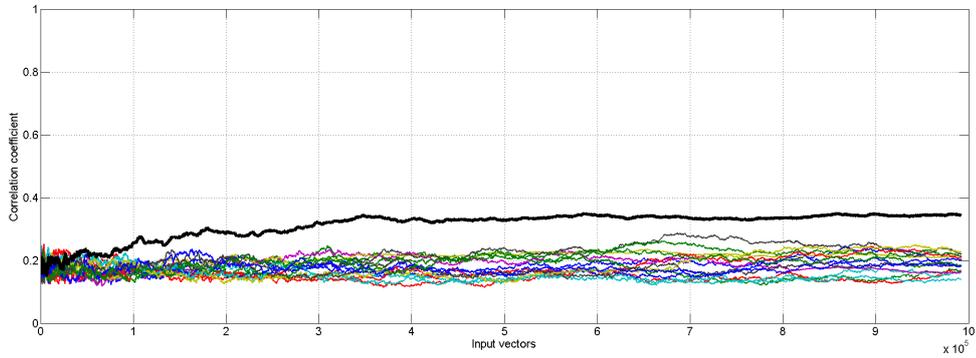


Figure 2.30. Correlation coefficients plot as a function of time for the SABL circuit in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.

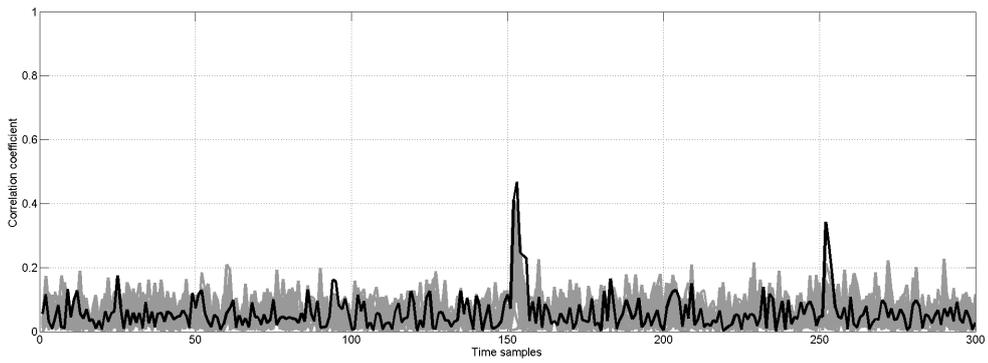


Figure 2.31. Correlation coefficients plot as a function of time for the SABL circuit in the case of $\sigma_{noise} \approx 2 \cdot 10^{-4}$; correct key is indicated in bold black line.

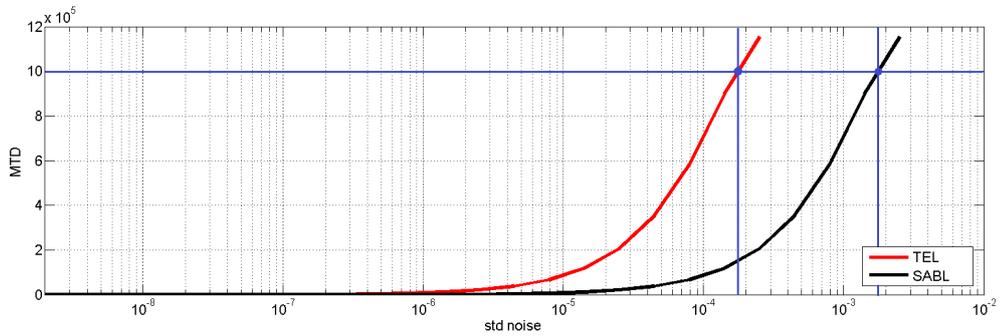


Figure 2.32. MTD as a function of the noise standard deviation.

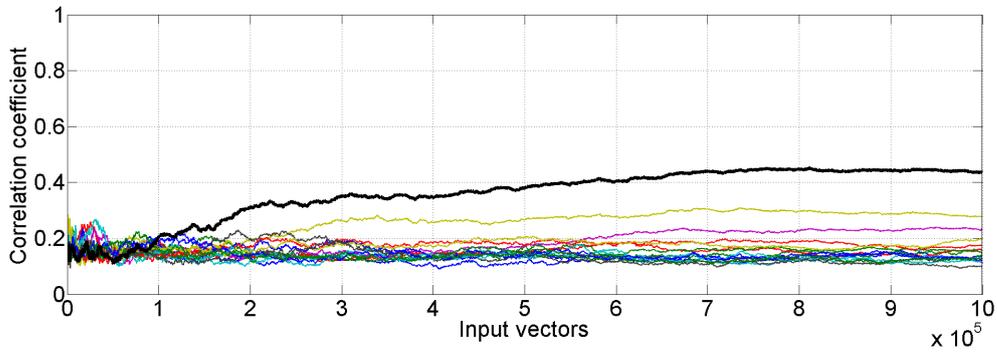


Figure 2.33. Correlation coefficients plot as a function of the number of input plaintexts for the TEL circuit, in the case of $\sigma_{noise} \approx 10^{-3} < \sigma_{noise}^{CR}$; correct key is indicated in bold black line.

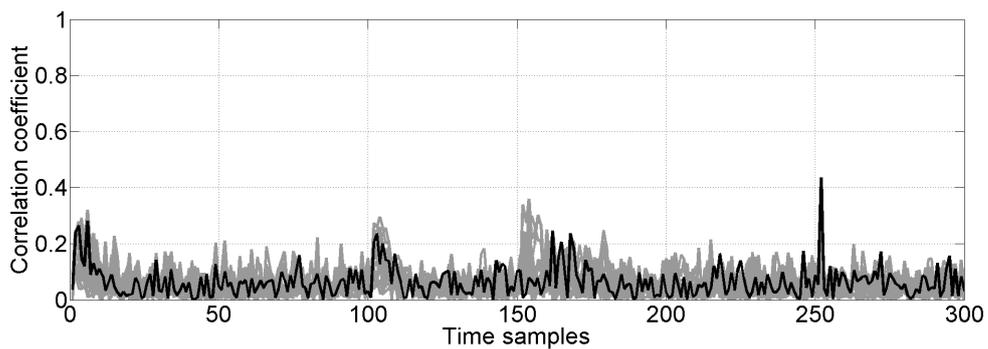


Figure 2.34. Correlation coefficients plot as a function of time for the TEL circuit in the case of $\sigma_{noise} \approx 10^{-3} < \sigma_{noise}^{CR}$; correct key is indicated in bold black line.

directly at layout level, but it can be easily done by the EDA tool during the digital design flow, without requiring a submicron precision as required for other techniques [47] [122] [124] or other additional efforts.

Anyway, TEL circuits are perfectly compatible to be implemented also together with one of these techniques for very high secure processors, at the expenses of the design complexity. Furthermore, this work proves that a frequency leakage analysis is fundamental already during the design steps, considering that novel SCAs, like *Electromagnetic Analysis Attacks (EMAs)* rely on the data-dependence in the frequency spectrum and are particularly critical also for DPLs.

Future research must be addressed towards the design of robust circuit templates to implement the TEL data encoding, both for ASIC design and FPGA applications, and towards the proposal of more precise power models which take into account time and frequency leakage at the same time, even adopting information theoretic metrics [71] [116]. Furthermore, as a future improvement in the evaluation of the security of TELs, other side-channel attacks known in literature, e.g. EMA attacks, can be mounted against a real implementation.

In next chapter, we describe more in detail one specific circuit implementation of TEL, the *improved Delay-based Dual-rail Precharge Logic (iDDPL)* style, which has been used for implementing the SERPENT-block evaluated in this chapter from a security point of view.

Chapter 3

An ASIC-oriented implementation of TEL circuits

3.1 Introduction

In the previous chapter we have described the Time Enclosed Logic circuits as a countermeasure against PAAs. The effectiveness of TEL circuits relies on the combination of two methodologies which aim at balancing the instantaneous power consumption at different abstraction levels: the first countermeasure acts at logic level and is based on the adoption of a novel data encoding protocol, which hides the data-dependence of the current in the time domain and overcomes the effect of unbalanced capacitive load; the second countermeasure is based on the insertion of decoupling capacitances in the layout of the circuit, which eliminate directly at physical level the residual leakage due to the timing mismatches. The effectiveness of TEL circuits has been proved on a case study cryptographic circuit, using a specific circuit implementation, the *improved Delay-based Dual-rail Precharge Logic (iDDPL)*, which requires a full custom design.

In this chapter we provide a complete description of the iDDPL style. For this purpose, we will present some combinational and sequential circuits which satisfy the requirements of TEL circuits. Furthermore, we will describe a specific circuit implementation which is based on the generation of the TEL data encoding in different points of the pipeline shown in Fig. 2.7 and allows to efficiently handle the TEL data flow along the combinational paths.

The objective of this chapter is to build a standard-cell library, composed of a number of iDDPL gates, adoptable to implement digital cryptographic circuits without any specific constraint on the place and route of the digital back-end flow. At the end of this chapter we will present a prototype iDDPL standard-cell library, the *DDPL065* library, which has been implemented using devices from the CMOS065 technology library. The DDPL065 library is composed of a minimum number of balanced iDDPL gates, sufficient to design cryptographic blocks (e.g. S-Box and registers), and characterized at layout level in terms of area, power and timing. The library has been used to design the SERPENT block evaluated in previous chapter, and can be improved and extended for further research in order to develop more complex logic circuits. Further details about the design flow of the SERPENT block

will be provided in next chapter.

All the electrical schemes described in this chapter have been designed in Cadence Virtuoso and simulated with Spectre in the Analogue Design Environment. The devices are the LPSVT BSIM4 transistors from the technology library CMOS065, already described in Chapter 1. If not differently indicated in the text, the simulations have been executed with an operating frequency of 10MHz, typical for standard cryptographic operations (e.g. smart-cards), a voltage supply of 1V, and rise and fall time of the signals of 10ps. The nominal delay δ of the TEL signals is set to 500ps. Finally, the layout of the logic cells has been done using Cadence Layout Editor Suite, with Calibre processor to perform the physical design rule check.

3.2 The Improved Delay-based Dual-rail Precharge Logic family

3.2.1 A full custom circuit implementation of TEL

In accordance to the discussion done in Chapter 2, and in particular with reference to Fig. 2.4, a TEL circuit can be implemented both using standard-cell logic gates and full custom cells. For this purpose, we have chosen this second option, which allows to better exploit the performances of the TEL circuit because relies on a transistor-level optimization. Furthermore, a full custom solution can be directly tested using SPICE simulations, in order to make the logic style prone for being implemented on a cryptographic ASIC, which represents the final scope of our work.

The first step in the design of a TEL implementation has been the study of the digital logic styles presented in literature with the aim of counteracting PAAs. For this purpose, we have identified two full custom logic styles as reference from which this work is started: the *Three-phase Dual-rail Precharge Logic (TDPL)* [21] and the *Delay-based Dual-rail Precharge Logic (DDPL)* [20], which have been both implemented in our department in last years.

Basically, TDPL is an improvement of the SABL style, enhanced with the presence of an additional output transistor in the circuit scheme of the logic cells, which introduces a third state in the logic data encoding, the *postcharge*, with the purpose of balancing the energy of the gate in presence of capacitance mismatches. TDPL circuit requires the routing of a second clock signal, which represents the control signal *ctrl* in Fig. 2.7. However, this poses a constraint on the layout of the circuit, because routing an additional dynamic signal requires to enlarge the design in order to avoid the effect of cross-coupling which may introduce crosstalk and impact the functionality of the circuit; moreover, the relevant interval of the data encoding is still too large, due to the presence of the sense amplifier.

On the contrary, the DDPL style is an improvement of TDPL, and is based on the following observation: to obtain the same result of the TDPL, that is balancing the energy of the circuit even in presence of unbalanced capacitances, it is possible to reduce the redundancy of the time interval length: instead of encoding a bit in a relevant interval fixed by the clock period, it is possible to generate a differential signal pair where the relevant time is reduced down to Δ , which represents a fraction of the clock period. This way, it is possible on a side to improve the energy balancing

activity of the circuit also if two differential output capacitances are mismatched by reducing Δ as much as possible, and at the same time this can be done irrespective of the clock frequency.

3.2.2 Limitations of the Delay-based Dual-rail Precharge Logic (DDPL) style

The iDDPL is conceived as an improvement of the DDPL style, which cannot be directly used as template for TEL circuits. Indeed, one of the most important drawbacks of the DDPL style is that even if the energy on a clock cycle is balanced, as proved by authors in [20], the instantaneous power consumption may still depend on the processed data. More specifically, the dynamic peaks in the current traces may have a certain time length, and this depends on the electrical mismatches, causing the leakage to be visible also outside the interval Δ , both in the evaluation and in the precharge phase. In other words, the DDPL style does not meet the requirement on timing enclosing defined for TEL circuits in Sec. 2.4.6, and the schemes presented by authors in [20] cannot be adopted to build PAA-resistant circuits implemented in a TEL fashion.

The new logic family we will describe is an improvement version of the DDPL style, and is called for this reason improved DDPL. The iDDPL style is proposed with the aim of balancing not only the energy in a clock cycle, but also the instantaneous power consumption. Unlike DDPL, the iDDPL style meets the two main requirements of TEL, described in Sec. 2.4.6 and reported in the following:

1. The average current on a clock cycle is balanced (property of *energy balancing*).
2. The instantaneous current exhibits a limited time interval in which the leakage could be potentially detectable (property of *timing enclosing*).

Similarly to DDPL, also the iDDPL circuit template is conceived for full-custom applications. The combinational and the sequential iDDPL circuits presented in this chapter are specifically designed and tested in order to meet these requirements; the subset of iDDPL logic gates which will be described represents the DDPL065 library used to design the SERPENT-block.

3.2.3 Cell template of an iDDPL gate

The iDDPL style is a differential dynamic logic, inspired by its predecessor DDPL [21]. The basic iDDPL BUFF/INV gate is shown in Fig. 2.6, and reported for simplicity in Fig. 3.1. We refer as n-type (p-type) to a dynamic circuit topology in which the evaluation network is the pull-down (pull-up). The cell library we will describe in this chapter is composed of n-type iDDPL gates. Even if DDPL gates were built in a p-type structure, this choice helps to reduce the area overhead of the circuit, thanks to the fact that nMOS transistors have a higher charge carrier mobility than pMOS and require less active area.

The working principle of the gate is described in the following. In accordance to the timing diagram of Fig. 2.3, when *clk* is low, the cell is in a frozen state because

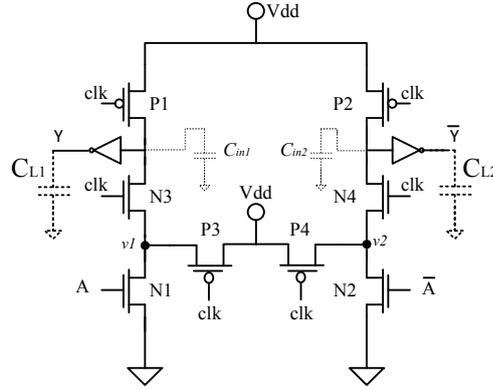


Figure 3.1. Cell template of an iDDPL BUFF/INV gate for the TEL circuits.

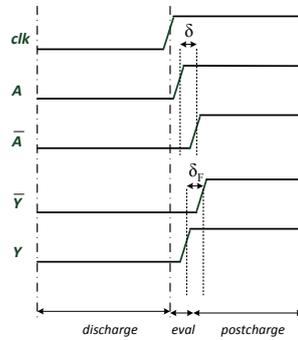


Figure 3.2. Timing diagram of the BUFF/INV cell in correspondence of signal (1,0) (logic-1 in CMOS domain)

the keeper transistors P1-P2 are simultaneously activated and force the internal nodes to V_{DD} . The outputs are then set to 0 through a couple of CMOS inverters which drive the input capacitances of the following gate, similarly to any Domino logic. This phase is the discharge (see Table 2.2).

The evaluation phase begins at the clock rising edge, when the keeper transistors are open. According to the data formalism of TEL encoding introduced in previous chapter, the first signal of the differential pair (A, \bar{A}) to exhibit a rising edge, that is the *asserted* signal, discharges all the internal capacitances of one half circuit. When the input capacitance of the output inverter ($v1$ or $v2$) is discharged through the evaluation network, the output capacitance is charged to V_{DD} .

After a time interval equal to δ , also the *delayed* or *non-asserted* signal goes to V_{DD} , and the transition is complete. This phase is called postcharge because also the other output capacitance is charged to V_{DD} . From this time instant the gate remains in a frozen state until the beginning of the next clock period. Finally, at the falling edge of the clock cycle, all the internal capacitances are again charged to V_{DD} , the output capacitances are discharged and another discharge phase begins. As an example, in Fig. 3.2 the timing diagram of the signals processed by the BUFF/INV gate when the input signal is (1,0) (i.e. a logic-1 in the CMOS domain) is reported.

As in standard dynamic logics, the functionality of the gate does not depend on

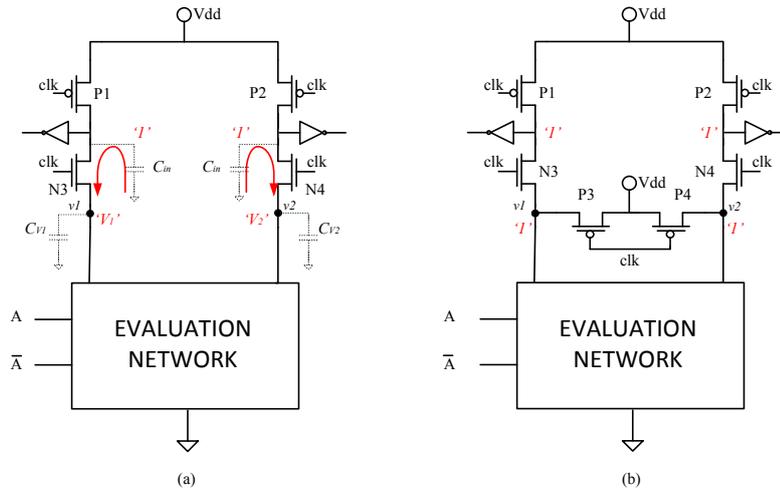


Figure 3.3. iDDPL gate suffering on the memory effect (a) and iDDPL gate without memory effect with the presence of internal keeper transistors (b)

the sizes of the transistors, that indeed can be chosen in order to have the minimum area; in the gate of Fig. 3.1 the nMOS transistors are sized with $W_n = 135nm$ and $L_n = 65nm$, whereas pMOS have $W_p = 200nm$ and $L_p = 65nm$; the CMOS inverters are taken from the standard-cell library and have the minimum fanout ($W_n = 200nm$, $W_p = 270nm$, $L_n = L_p = 65nm$). The fanout of the gate can be increased by enhancing the size of the CMOS inverters, and possibly by scaling the sizes of the transistors of the cell if the evaluation network cannot drive the capacitances C_{in} of the output inverters.

The reason for the presence of keeper transistors P3-P4, which is an improvement with respect to the DDPL template, depends on the phenomenon of *charge sharing*. In fact the internal capacitances C_{v1} and C_{v2} of a combinational cell are different, in accordance to the number of transistors of the evaluation network needed to implement the logic function. With reference to the signal waveforms, at the end of the clock cycle, during the postcharge phase, all the internal capacitances of the cell in Fig. 3.3(a) are discharged to zero; in the next clock cycle, at the beginning of the discharge phase, when clk goes to zero, the nodes $v1$ and $v2$ would be floating, and the charge on the capacitances C_{in} flows into C_{v1} and C_{v2} ; this leads to an increase of the voltage at nodes $v1$ and $v2$ from zero to the voltages V_1 and V_2 respectively, with in general $V_1 \neq V_2$.

The phenomenon of charge sharing has been detected in other dynamic logic styles and usually is not critical. However, the drawback in DPLs is that at the beginning of the evaluation and the postcharge phases the dynamic peaks of the current trace depend in a case on the value V_1 and in the other on V_2 ; therefore, when the evaluation network is activated and the capacitances are discharged, the current trace exhibits a pattern which depends on the data combination. This leads to the so called *memory effect* on the internal nodes, which has been detected in other previously published DPL styles (e.g. [70]). A simple solution to eliminate the memory effect is precharging the critical nodes at V_{DD} . Thus the insertion

of transistors P3-P4 (Fig. 3.3(b)) has the purpose of keeping the internal nodes $v1$ and $v2$ fixed to V_{DD} , so that during the precharge phase the charge sharing cannot occur, and the amount of charge needed to discharge the capacitances C_{in} in the evaluation path is always constant. This solution creates an unavoidable amount of additional area, which is anyway small and has the purpose of eliminating any residual data-dependence in the current trace, and has been also adopted in other DPLs to counteract this effect, like in [70].

Another important property of the iDDPL cell template is that the circuit can be seen as two independent half circuits. The evaluation and the postcharge phases are reciprocally asynchronous, therefore the differential signals propagate independently along the two halves. On the contrary, in the SABL circuit template (see Fig. 2.6), when one half circuit evaluates discharging the input capacitance of the correspondent output inverter, then a feedback loop composed of two cross coupled inverters forces the logic state on the other half circuit.

As it will be shown in next sections, this topological property of the iDDPL gates offers the advantage of simplifying the layout, preventing cross talk effects that may occur by the presence of feedback loops and floating nodes which are very sensitive on the switching activity of near wires.

The architecture of iDDPL cells is fully compatible with the TEL principle. The energy per cycle of the cell is expected to be balanced, according to the analysis done in Sec.2.4.2. Note that the current adsorption is independent from the order of arrival of the signals, because each capacitance in the circuit is always charged and discharged once in the clock cycle. Moreover the output inverters exhibit no data dependence because in each clock cycle they perform the same transitions (i.e. $0 \rightarrow 1$ and $1 \rightarrow 0$ on complementary outputs). The static consumption of the logic gate is ideally zero, because the presence of N3-N4 assures that no direct current path may exist between V_{DD} and the ground.

In accordance to the cell template of Fig. 3.1, the scheme of an iDDPL AND/NAND gate is presented in Fig. 3.4. The pulldown network is composed of 4 transistors, which represents the minimum number to implement the AND and the NAND functions in a differential logic.

The instantaneous current traces of the BUFF/INV and the AND/NAND gates driving another similar cell are simulated in Cadence and depicted in Fig. 3.5; in the figures, the traces which correspond to all possible data transitions are superimposed. The plots have been generated with a frequency of 100MHz and $\delta = 100ps$.

3.2.4 Conversion of the signal from the CMOS domain

An important issue to be analyzed is how to convert the signal from the CMOS single-rail (SR) domain into the TEL domain, in order to be fed as input of a iDDPL circuit. For this purpose, the circuit shown in Fig. 3.6 is proposed.

The circuit elaborates the datum IN and its negated not_IN . The gate takes as input the clock clk and a δ -delayed version of the clock clk_d . When clk is low, transistors P1 and P2 force the internal capacitances of the output inverter to V_{DD} , and the differential output (D, \bar{D}) are set to zero (discharge). When clk goes to V_{DD} ,

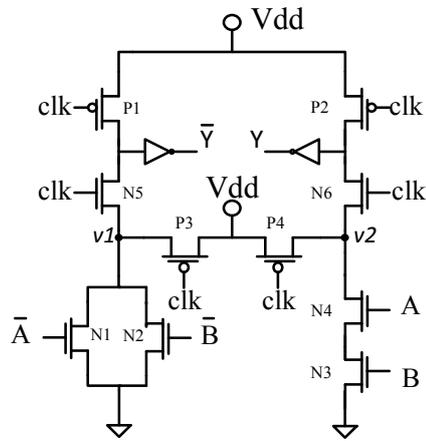


Figure 3.4. Cell template of an iDDPL AND/NAND logic gate.

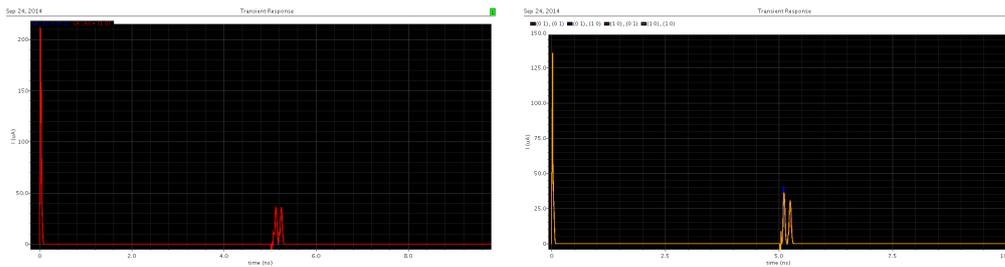


Figure 3.5. Current traces of an iDDPL BUFF/INV (left) and an AND/NAND gate (right)

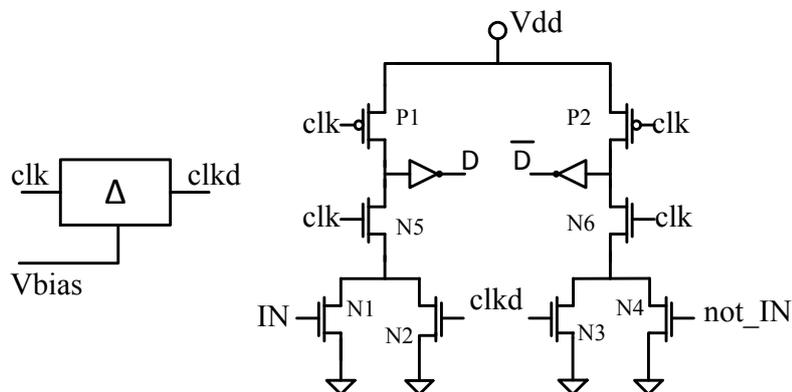


Figure 3.6. Circuit for the conversion of a CMOS signal into the TEL domain for the iDDPL style.

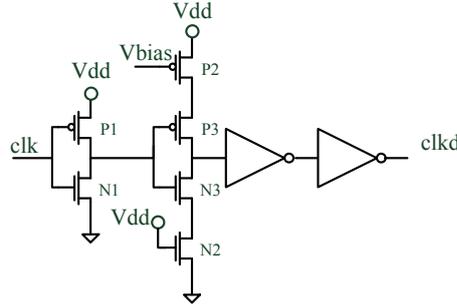


Figure 3.7. Implementation of a delay element line based on a current starved inverter for the DDPL flip-flop.

$N5$ and $N6$ are closed and connect the inverters to the pull-down network; when IN is a logic 1, not_IN is 0 and viceversa, therefore only one half circuit can discharge the input capacitance of the inverter, setting the output wire to V_{DD} (evaluation). The other half circuit is activated when clk_d goes to V_{DD} , after a time delay δ ; from this point, all the output signals are at V_{DD} (postcharge).

The main part of the logic gate is the block Δ in Fig. 3.6, which represents a delay line and has the purpose of generating clk_d . There are several solutions to implement a delay line in digital circuits. A straightforward method for doing this is using a chain of inverter stages, but this solution is not area efficient even for generating a δ in the order of few hundreds of picoseconds. For instance, in order to obtain a δ equal to 500ps, about 20 inverter stages would be required for a single converter in the CMOS065 technology, therefore this cannot be considered as an optimized and low area solution.

An analysis of different circuit solutions to implement a constant delay at circuit level using CMOS technology is reported in [57]. Among the reported circuits, we selected the topology shown in Fig. 3.7 which is very simple and area efficient.

The circuit shown in Fig. 3.7 is composed of four inverting stages, thus the output waveform has the same phase of the input waveform: the first stage is a standard CMOS inverter, whereas the second stage is a current starved inverter. A current starved inverter [83] is a circuit which inverts the signal similarly as a standard inverter, but unlike the latter it allows to control the propagation times of the rising and the falling edge by changing the pullup and the pulldown resistances. This can be done by applying an external voltage on the gate of two additional transistors, a nMOS and a pMOS, which are inserted in the pulldown and the pullup path respectively. For our application, only one edge must be delayed, therefore a control voltage V_{bias} is applied on the gate of the pMOS transistor P2 in the pullup path, whereas transistor N2 is connected to V_{DD} in order to leave unchanged the path resistance of the pull-down network. Anyway the presence of N2 is redundant and could be removed. Note that this way the signal clk_d at the output has a different duty cycle with respect to clk .

At the output of the starved inverter, there are two cascaded CMOS inverters which buffer the clk_d signal on the load, balancing the rising and falling times of the signal and regenerating the waveform. The advantage of this topology with respect

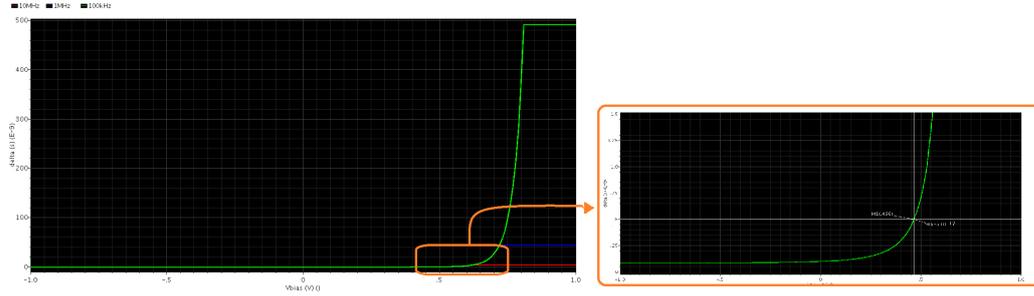


Figure 3.8. Dependence of the delay δ with the control voltage V_{bias}

to other solutions (e.g. the insertion of CMOS varactors [69]) is that a stable and controllable delay can be obtained by changing the polarization of P2; furthermore, alternating a current starved inverter with basic inverters lowers the sensitivity of the delay to temperature and supply voltage variations, as reported in [58].

The block Δ is described through the tuning range, that is the curve of the variation of δ with the control voltage V_{bias} . The tuning range curve is reported for different operating frequencies in Fig. 3.8, considering a range of $[-1V +1V]$ for the control voltage. The supply voltage is equal to 1V and the rise and fall times of the input signal are of 10ps. With this setup, to obtain $\delta = 500ps$, V_{bias} must be chosen equal to 486mV. We have inserted a load capacitance of 1fF in order to simulate the gate capacitance of transistor N2 in Fig. 3.7, whereas the parasitic capacitances of the converter are not taken into account in this simulation. Note that the variation of δ with V_{bias} is independent from the frequency, provided that $\frac{T_{CK}}{2} > \delta$. The operating frequency sets a limit for the choice of δ , and therefore $V_{biasMAX}$ represents the maximum allowable value for V_{bias} before that $\delta = \frac{T_{CK}}{2}$. In other words, the operating frequency reduces the domain of δ , and thus of V_{bias} . A further analysis of this circuit will be executed in next chapter.

3.2.5 Design-time metrics and simulation parameters

In the remaining part of this chapter, we will adopt some metrics to assess the performances of the circuit schemes in terms of functionality and security.

The first metric refers to the ability of a logic gate to balance the average current on a clock cycle, that is directly linked to the energy of the cell. For this purpose we use the *Normalized Energy Deviation (NED)*, already recalled in the previous chapter, and the *Normalized Standard Deviation (NSD)*. Energy per cycle, NED and NSD are so calculated [121] [123]:

$$E_{AV} = V_{DD} \int_0^T I_{DD}(t) dt \quad (3.1)$$

$$NED = \frac{\max[E_{AV}] - \min[E_{AV}]}{\max[E_{AV}]} \quad (3.2)$$

$$NSD = \frac{\sigma_E}{\mu_E} \quad (3.3)$$

where I_{DD} is the current drawn by the circuit in presence of one data combination among a number of possible N configurations, and E_{AV} is the correspondent average energy in a clock cycle; μ_E and σ_E are the average and the standard deviation of the distribution of the energies per cycle for all the N configurations. The energy per cycle is calculated measuring the adsorbed current $I_{DD}(t)$ on the V_{DD} pin and integrating it in a clock cycle. Smaller are NED and NSD, better is the energy balancing ability of the gate.

NED and NSD are linked to the average current adsorbed in a clock cycle, but they do not give information on how much each time sample of the instantaneous current trace is scattered according to the data transitions. More specifically, even in presence of a low energy deviation, some current samples could exhibit a data-dependence which could be exploited by PAAs for extracting information on processed data.

As described in previous chapter, TEL circuits suffer on the early evaluation errors due to the fact that a combinational network changes the value of the dynamic delay δ . As it will be describe in next paragraphs, the *fluctuation effect* is a direct consequence of early evaluation. Fluctuation is a transistor level effect due to the increase of the value of the dynamic delay δ_F at the output of a combinational circuit with respect to the input delay, which may affect the timing specifications of a TEL gate. In order to assess this issue, we use the the variation of δ , indicated as $\Delta\delta$:

$$\Delta\delta = \delta - \delta_F \quad (3.4)$$

Note that if the parameter $\Delta\delta$ is greater than zero, it coincides with the the propagation time of the gate defined in Chapter 2. Thus, in order to avoid fluctuation and guarantee a positive propagation time of a gate, this parameter must be positive. A negative variation $\Delta\delta$ for any input combination must be avoided both for security and timing issues of the entire TEL circuit.

The above mentioned energy balancing and timing enclosing properties can be assessed by the following conditions, respectively:

1. $NED = 0$
2. $\Delta\delta < 0$ for any input data

The most critical aspect is represented by the mismatches on the interconnect wires, therefore NED and $\Delta\delta$ must be calculated in an extreme condition (e.g. the maximum expected mismatch).

However the above conditions are ideal. In practical applications where is is actually impossible to control each source of mismatch, they can be relaxed by:

1. $NED \approx 0$ ($NED\% \leq 1\%$)
2. $\frac{|\Delta\delta|}{\delta} \approx 0$ if $\Delta\delta < 0$

These metrics are very useful to assess the balance of combinational TEL circuits in presence of mismatch. However, as it will be discussed in next sections, the design of sequential circuits must be executed with more caution. Registers in DPL styles represent an important information leakage source, and in a power-balanced

circuit the number of leaking samples must be reduced as much as possible. For this purpose a good statistical gauge to assess the dependence of the instantaneous adsorbed current on the input pattern is the coefficient of variation CV [18], which represents a normalized measure of the dispersion of the set of current samples as a function of time. CV is calculated as the ratio between the standard deviation and the mean of the set of the current samples of the traces at each time instant:

$$CV(t) = \frac{\sigma_I(t)}{\mu_I(t)} \quad (3.5)$$

Unlike NSD, which gives an estimation of the normalized dispersion of the average current (i.e. the energy) on a clock cycle for every data configuration, CV calculates the dispersion on each time sample of the current traces. Therefore, it provides information about the leakage content of each current samples in the time domain [50].

In the case under consideration, a large coefficient of variation for the current samples measured at a certain time instant indicates that at that time instant the current traces exhibit a strong variability with the input data. Stated that some leakage is expected, a well balanced sequential element should exhibit low NED and NSD (i.e. low energy deviation), and at the same time the shortest possible time interval of high CV (i.e. only the current samples falling in the relevant interval may exhibit a high deviation).

3.3 Design and characterization of iDDPL combinational gates

3.3.1 Fluctuation effect in iDDPL gates

In this section we present the circuit schemes of a basic set of iDDPL combinational gates, which have been designed following the principles of the schemes depicted in Fig. 2.4 and using the cell template of the inverter in Fig. 3.1. In order to be adoptable to the templates reported in Fig. 2.4, the circuits must have the fundamental property to have balanced propagation times on the differential signals so to avoid timing mismatches and satisfy the timing enclosing property. The templates of Fig. 2.4 are built using a redundant implementation of the logic function in order to obtain balanced networks; for example, the circuit scheme of the minimum AND/NAND gate reported in Fig. 3.4 does not satisfy this design rule.

The objective of this section is to perform an in-depth analysis of some balanced combinational logic in iDDPL. More specifically we focus on the study of some Boolean functions and their dual function which are optimized on the timing mismatches: NOT, NAND, NOR, XOR. These logic gates represent a complete set of basic cells adoptable to build more complex logic functions with balanced propagation times. The proposed gates are at minimum fanin and minimum fanout, optimized to reduce the area overhead, but the analysis could be extended in the future for more complex cells. With respect to the templates of Fig. 2.4, the iDDPL is a dynamic Domino style and allows to reduce the transistor number by avoiding

the complementary network of standard CMOS gates.

In Chapter 2, we have introduced the so called *fluctuation effect of δ* in TEL circuits, which has been defined as the random variations of δ along a combinational path which results in an increase of the output delay δ_F with respect to the nominal value δ . This phenomenon must be avoided in order to satisfy the timing enclosing property and meet the timing constraints of the pipeline as well as fix the level of security of the circuit to the desired level.

In next paragraphs we will study this effect at gate level by verifying that the condition $\Delta\delta < 0$ is satisfied for any input data. More specifically, it will be shown that possible causes of fluctuation in TEL combinational networks are:

1. the early evaluation of the data;
2. the different values of the capacitances at the output nodes;
3. the mismatch variations of the adjacent devices.

3.3.2 Analysis of the early evaluation effect

The *early evaluation* is a transistor-level effect which causes a logic gate to evaluate before all inputs are valid. This effect is directly linked to the logic function and is mapped into the physical implementation. It is very critical for a DPL combinational gate because it produces a dependence of the adsorbed current on the arrival times of the input signals, resulting in a data dependent power consumption even for DPA-resistant circuits implemented with perfectly balanced internal and output capacitances.

There is a number of papers in which authors describe this vulnerability in the DPA-resistant logic families, both theoretically [64] [107] and experimentally [12]. In many cases the early propagation must be eliminated through the design of more complicated logics, with the drawback of an additional hardware overhead, both for solution based on existing standard cell (FPGA) and for full-custom logic styles (ASIC). Usually, the right solution is adding a minimum amount of redundant transistors in order to avoid early evaluation and minimize the area overhead.

In general, using Boolean functions synthesized with the minimum number of operators does not allow to mitigate this effect; thus adding some redundancy in the input pattern dependence is usually convenient. In next paragraphs we propose a theoretical model which describes the variation of δ in some basic combinational gates. The purpose of this analysis is to verify to what extent the early evaluation effect may impact the value of δ_F , and possibly to furnish a light circuit level solution by re-designing a cell in order to guarantee that $\delta_F < \delta_{inA}, \delta_{inB}$ for each input data combination by adding a minimum amount of redundancy.

In order to provide an accurate model to analyze the early evaluation in TEL combinational gates, the synchronization of the differential signals at the input of the logic must be adequately defined. With reference to Fig. 2.8, the differential signal of a combinational path propagate with different propagation times T_1 and T_2 . In order to guarantee that $\delta_{CL} < \delta$, it must be ensured that $T_2 < T_1$. If we consider a generic gate in the path, without loss of generality we assume that the input pair (A, \bar{A}) has a delay $\delta_A = \delta$, and the second input pair (B, \bar{B}) is delayed with respect

to (A, \bar{A}) so that the rising edges of the asserted signals are separated of t_1 and the rising edges of the delayed signals of t_2 , with $t_2 < t_1$. This assumption is compatible with the fact that the input signals (A, \bar{A}) and (B, \bar{B}) arrive from gates which do not suffer on fluctuation, that is $\delta_A = \delta$ and $\delta_B = \delta_A + t_2 - t_1 \leq \delta$. This is consistent with the assumption that the delay between complementary input signals cannot be greater than δ . Moreover, it must be noted that t_1 must be always less than δ_A at the input of a logic gate in order to guarantee that the gate correctly processes the input data.

In the circuit schemes analyzed in the following paragraphs the sizes of the transistors are optimized for power and area requirements. We use a minimum length equal to $L_{min} = 65nm$ and an aspect ratio $\frac{W}{L}$ equal to 2 and 3 for nMOS and pMOS transistors, respectively, which are compatible with the values found in the CMOS065 tech library. In this analysis we suppose that the load is balanced; the effect of unbalanced capacitances will be analyzed in the following paragraphs.

Analysis of δ in the minimum iDDPL AND/NAND gate

The iDDPL BUFF/INV cell of Fig. 3.1 is a symmetric gate, therefore signals have identical propagation times and no fluctuation of δ is found in presence of balance load. On the contrary, the iDDPL AND/NAND gate shown in Fig. 3.4, and reported in Fig. 3.9, has an asymmetric evaluation network. In Fig. 3.9 also the equivalent circuits PD1 and PD2 of the two differential pull-down networks during the evaluation phase are shown. In the following, we use the RC model with resistive switches, which is the simpler methodology to analyze first order digital networks.

The evaluation network of the cell is composed of four transistors, that is the minimum number required to implement the dual logic function AND/NAND. The physical design of a gate is minimized by exploiting the fact that, for some input combinations, a gate can propagate its logical output early without having to wait for all of the logical inputs, but as we have already mentioned this represents a drawback in an anti-DPA logic style because it generates a power consumption dependent on the arrival times of signals. This can be observed in Fig. 3.9: the circuit PD1 is activated when N1 or N2 is closed, whereas PD2 is activated when both N3 and N4 are closed. This means that the capacitance at the node $v2$ is discharged only after that both the rising edges of A and B are arrived; on the contrary, the capacitance at the node $v1$ is discharged only at the arrival of the first signal without waiting for the second one. This creates a timing mismatch between PD1 and PD2 dependent on the input data.

In the following, we neglect the asymmetry of the internal parasitic capacitances. This is consistent with the fact that the overall impedance is dominated by the input capacitance of the output inverter and the pull down resistances. Therefore, let us assume in the calculation $C_{v1} = C_{v2} = C$. Actually the overall capacitance at the node $v2$ is slightly smaller than the capacitance at $v1$, but given that the pass transistors P3 and P4 charge both C_{v1} and C_{v2} at V_{DD} during the discharge phase in order to eliminate the memory effect, the impact of the asymmetry between C_{v1} and C_{v2} on the time constants during the evaluation/postcharge is slight.

The propagation time of the evaluation network, and therefore the circuit model

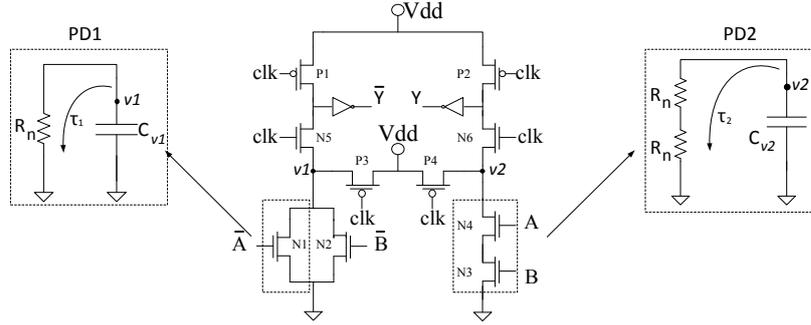


Figure 3.9. A basic iDDPL AND/NAND gate with the equivalent circuits of the evaluation network.

PD1 or PD2, depends on how many transistors are simultaneously activated. If we model the pull-down resistance of each transistor with a resistor R_n , then during the evaluation phase the capacitances C_{v1} and C_{v2} have a different time of discharge because the number of simultaneously activated transistors is different. The time constants satisfy relation Eq. 3.6.

$$\tau_1 = R_n C < \tau_2 = 2R_n C \quad (3.6)$$

C_{v2} discharges more slowly than C_{v1} , and Y has a propagation time greater than \bar{Y} . We name $\Delta\tau_{and} = \tau_2 - \tau_1$ the delay associated to the difference between the two pull-down paths, with $\Delta\tau_{and} > 0$. The analysis of the variation of δ for different data inputs is reported in equations Eq. 3.7- 3.10, which refer to the time diagram in Figure 3.10.

$$\delta_F^{1,0} = \delta + t_2 - t_1 + \Delta\tau_{and} \quad (3.7)$$

$$\delta_F^{0,0} = \delta + t_2 + \Delta\tau_{and} \quad (3.8)$$

$$\delta_F^{1,1} = \delta - t_1 - \Delta\tau_{and} \quad (3.9)$$

$$\delta_F^{0,1} = \delta + \Delta\tau_{and} \quad (3.10)$$

$\delta_F^{A,B}$ indicates the output delay for the inputs $A = (A, \bar{A})$ and $B = (B, \bar{B})$, where 1 stands for (1, 0) and 0 stands for (0, 1). Being $t_1 > t_2$ for hypothesis and $\Delta\tau_{and} > 0$, equations 3.8 and 3.10 clearly indicate that $\delta_F < \delta$, and the timing enclosing condition, $\Delta\delta > 0$, is not satisfied.

Design of an improved iDDPL AND/NAND gate and analysis of δ

The fluctuation effect detected in the AND/NAND gate depends on the asymmetry of the evaluation network, which is characterized by different propagation times along the differential halves. In [126] authors present a design methodology to create fully connected differential pull-down networks so to balance the propagation delays for any input combination. Some dummy transistors are inserted with the aim to equalize the resistive path during the evaluation/postcharge phase. This methodology has been generalized in the circuit template of Fig. 2.4, where the propagation times have been balanced by inserting the same number of NAND gates

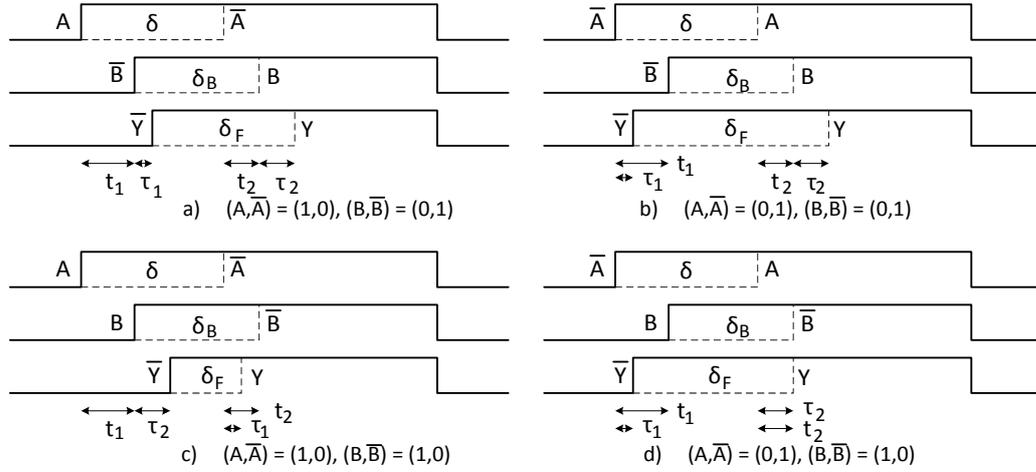


Figure 3.10. Time diagram of the evaluated signals at the output of a basic DDPL AND/NAND cell for all possible inputs.

in the paths and relying on the fact that TEL signals have always a $0 \rightarrow 1$ transition in the evaluation/postcharge. We use this methodology to design an optimized version of the iDDPL AND/NAND gate, using two nMOS transistors to implement each possible logic path (Fig. 3.11). Such an evaluation network allows to balance the propagation times for each possible input data.

As visible in Fig. 3.11 the pull-down sub-circuit on the left is represented by two series resistances R_n , which must be simultaneously activated; on the contrary, the sub-circuit on the right part of figure can be modeled with a network composed of a variable number of resistances, depending on the number of simultaneously activated evaluation paths. The time constants of the networks PD1 and PD2 satisfy Eq. 3.11:

$$\tau_1 = 2R_n C > \tau_2 = \frac{2}{3}R_n C \quad (3.11)$$

According to the possible data combinations, the sub-circuit on the right behaves like PD1 or PD2, thus the time constant associated to the discharge of the capacitance C_{v2} can be τ_1 or τ_2 according to the number of simultaneously activated transistors, whereas for C_{v1} the time constant is always τ_1 .

Similarly to the case of the minimum AND/NAND gate, for the equivalent circuits we have neglected the asymmetry of the internal parasitic capacitances and considered $C_{v1} = C_{v2} = C$. We name $\Delta\tau_{and} = \tau_1 - \tau_2$ the delay associated to the difference between the propagation times of the two pull-down paths, with $\Delta\tau_{and} > 0$. At this point, two different cases can be distinguished, according to the value of t_2 . If the input pairs are delayed so that the rising edges of the not asserted signals are separated by $t_2 > \tau_1$, the sub-circuit networks behave always as PD1 (Fig. 3.12), whereas if $t_2 < \tau_1$ the sub-circuit network on the right can behave like PD1 or PD2 according to the input data (Fig. 3.13).

The case $t_2 > \tau_1$ is described by Eq. 3.12- 3.15:

$$\delta_F^{1,0} = \delta + t_2 - t_1 \quad (3.12)$$

$$\delta_F^{0,0} = \delta + t_2 - t_1 \quad (3.13)$$

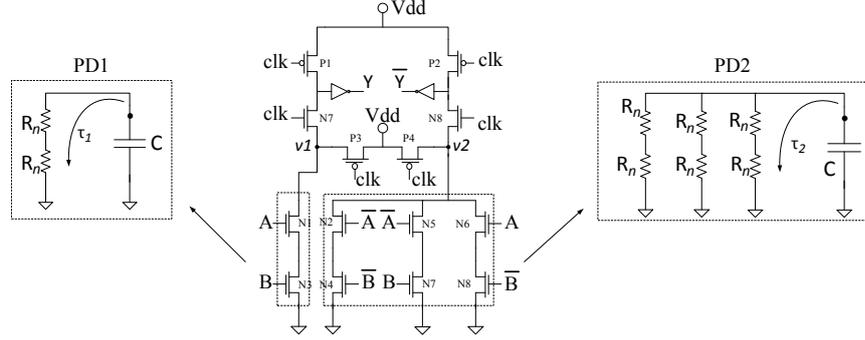


Figure 3.11. A basic iDDPL AND/NAND gate with the equivalent circuits of the evaluation network.

$$\delta_F^{1,1} = \delta - t_1 \quad (3.14)$$

$$\delta_F^{0,1} = \delta - t_1 \quad (3.15)$$

In the case $t_2 < \tau_1$, the output delayed signal propagates according to the time constant of PD2 when all transistors are simultaneously activated. The output delay is calculated in Eq. 3.16- 3.19:

$$\delta_F^{1,0} = \delta + t_2 - t_1 \quad (3.16)$$

$$\delta_F^{0,0} = \delta + t_2 - t_1 \quad (3.17)$$

$$\delta_F^{1,1} = \delta + t_2 - t_1 - \Delta\tau_{and} \quad (3.18)$$

$$\delta_F^{0,1} = \delta - t_1 \quad (3.19)$$

Being $t_1 > t_2$ for hypothesis and $\Delta\tau_{and} > 0$, from equations 3.12- 3.15 and 3.16- 3.19, the condition $\delta_F < \delta$ is verified for every input combination: even if the asymmetry of the pull-down network of the AND/NAND gate generates a non-constant output delay δ_F , the value of δ does not fluctuate at the output of the gate.

According to equations, it is straightforward to note that for the AND/NAND gate the propagation time depends on the input data combination. The maximum propagation time depends on the delay between the asserted signals at the input of the gate:

$$\Delta\delta_{MAX} = t_1 \quad (3.20)$$

It will be shown that this value represents the sum of the constant times of the gate which come before in the pipeline.

Design of a XOR/NXOR gate and analysis of δ

A similar analysis is carried out for the iDDPL XOR/NXOR gate. In Fig. 3.14 a n-type iDDPL XOR/NXOR gate is shown. Unlike the case of the AND function, the symmetry of the XOR function can be easily mapped into a symmetric circuit topology, therefore the pull-down network can be represented by two equal differential

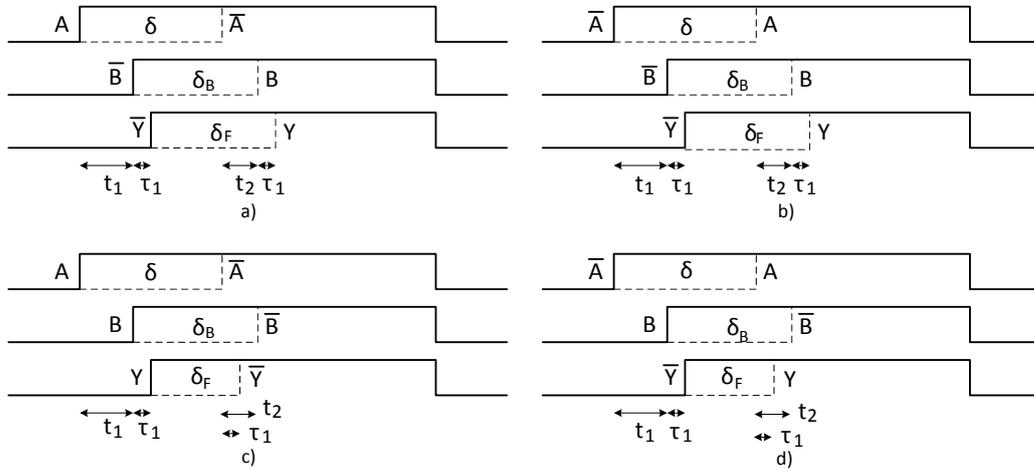


Figure 3.12. Time diagram of the evaluated signals at the output of the improved iDDPL AND/NAND cell for all possible inputs, when $t_2 > \tau_1$.

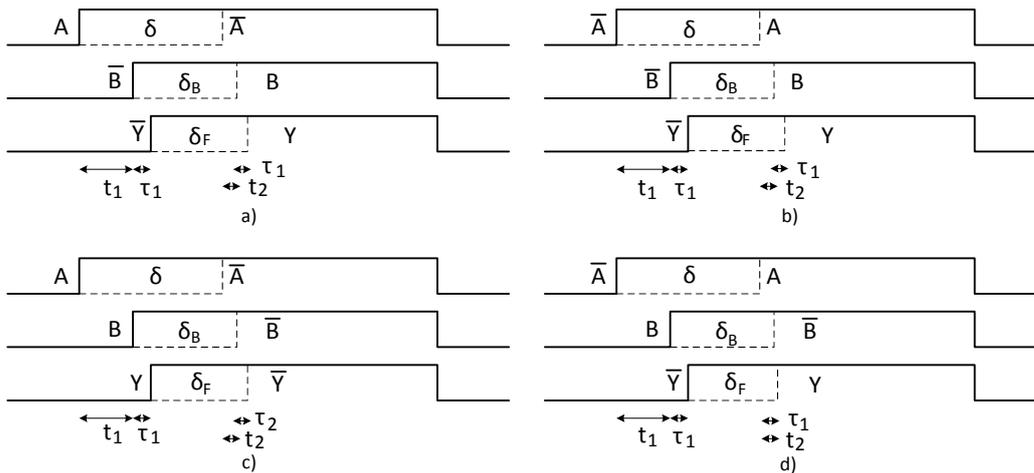


Figure 3.13. Time diagram of the evaluated signals at the output of the improved iDDPL AND/NAND cell for all possible inputs, when $t_2 < \tau_1$.

sub-circuits composed of two parallel branches with two series transistors. As shown in Fig. 3.14, in each differential sub-circuits one branch (PD1) or two branches (PD2) conduct, according to the number of simultaneously activated transistors. Again, we consider $C_{v1} = C_{v2} = C$. The time constants of PD1 and PD2 satisfy Eq. 3.14.

$$\tau_1 = 2R_n C > \tau_2 = R_n C \quad (3.21)$$

We name $\Delta\tau_{xor} = \tau_1 - \tau_2$ the delay associated with the difference between the propagation times of the two pull-down paths, with $\Delta\tau_{xor} > 0$. Even in the case of the XOR/NXOR gate, two different cases can be distinguished.

If the input pairs are delayed so that the rising edges of the non-asserted signals are separated by $t_2 > \tau_1$ (see Fig. 3.15), the sub-circuit networks behave always as PD1 because of the symmetry of the gate. This leads to a constant value for the actual delay δ_F irrespective of the input data configuration (Eq. 3.22):

$$\delta_F = \delta + \tau_1 - (t_1 + \tau_1) = \delta - t_1 \leq \delta \quad (3.22)$$

Thus the output delay δ_F is independent from $\Delta\tau_{xor}$ and, more important, is always less than δ for all possible data combinations.

The second case occurs when the delay of the non-asserted signals of (B, \bar{B}) from (A, \bar{A}) is negligible, that is if $t_2 < \tau_1$ (see Fig. 3.16). In this case the output non-asserted signal propagates according to the time constant of the circuit model PD2 where all transistors are simultaneously activated. The propagation time reduces from τ_1 to τ_2 because the pull-down resistance path is lower (see Eq. 3.23), and in this case the output delay depends on $\Delta\tau_{xor}$:

$$\delta_F = \delta + t_2 + \tau_2 - (t_1 + \tau_1) = \delta + t_2 - t_1 - \Delta\tau_{xor} \leq \delta \quad (3.23)$$

In any case, no increase of δ is expected in the iDDPL XOR/NXOR gate. The symmetric topology of the gate and the balanced evaluation network help to avoid fluctuation. Similarly to the case of the AND/NAND gate, the maximum propagation time of the XOR/NXOR is:

$$\Delta\delta_{MAX} = t_1 \quad (3.24)$$

The previously presented analysis allows to conclude that by carefully designing the evaluation network, the early propagation effect in the iDDPL combinational gates can be controlled. The guideline is to guarantee a good balance of the resistive paths of the evaluation network by inserting dummy transistors. This way the propagation times of the asserted and the not asserted signals, which in turns depend on the time constants associated to the resulting resistive path, are constant irrespective of the input data combination and their arrival times.

In Fig. 3.17 reduced implementations of early evaluation free AND/NAND, OR/NOR and XOR/NXOR gates are reported. The OR/NOR gate has the same topology of the AND/NAND gate, but with swapped signals according to the De Morgan law. A similarly handmade analysis can be conducted also for these gates, obtaining again the result $\delta_F < \delta$ for every input combination. The pull down network of these cells requires only 6 transistors instead of 8 transistors, which represents the minimum number of transistors to implement these Boolean functions without fluctuation.

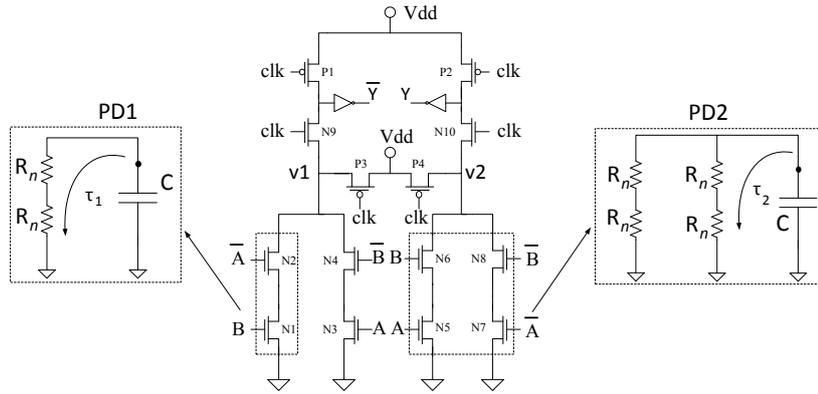


Figure 3.14. A basic DDPL XOR/NXOR gate with the equivalent circuits of the evaluation network.

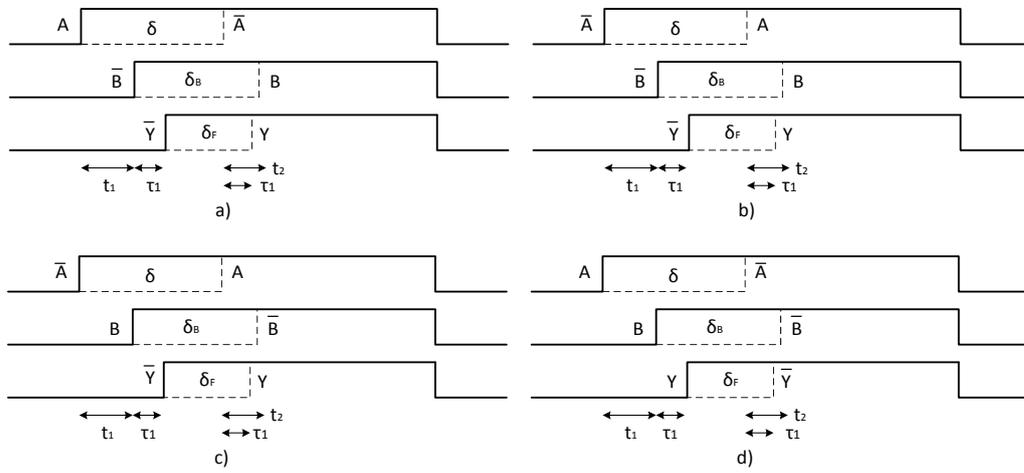


Figure 3.15. Time diagram of the evaluated signals at the output of a basic DDPL XOR/NXOR cell for each input, for the case $t_2 > \tau_1$.

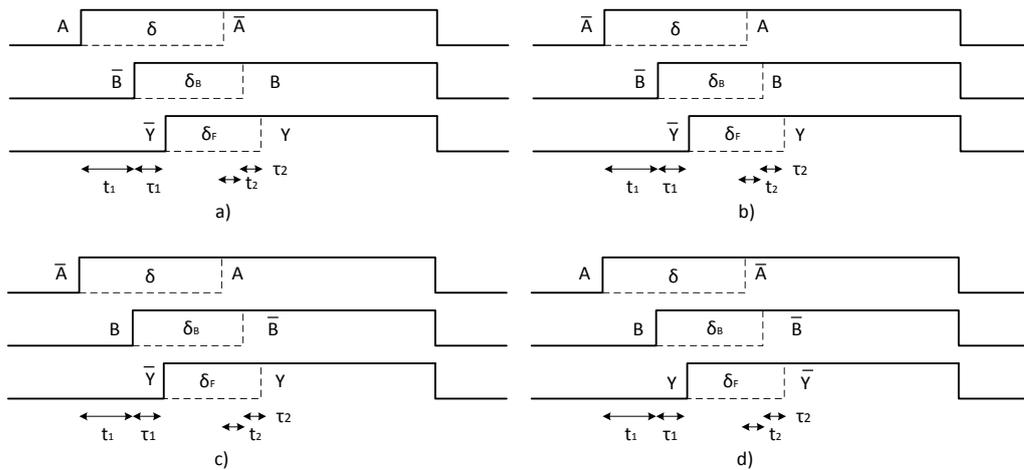


Figure 3.16. Time diagram of the evaluated signals at the output of a basic DDPL XOR/NXOR cell for each input, for the case $t_2 < \tau_1$.

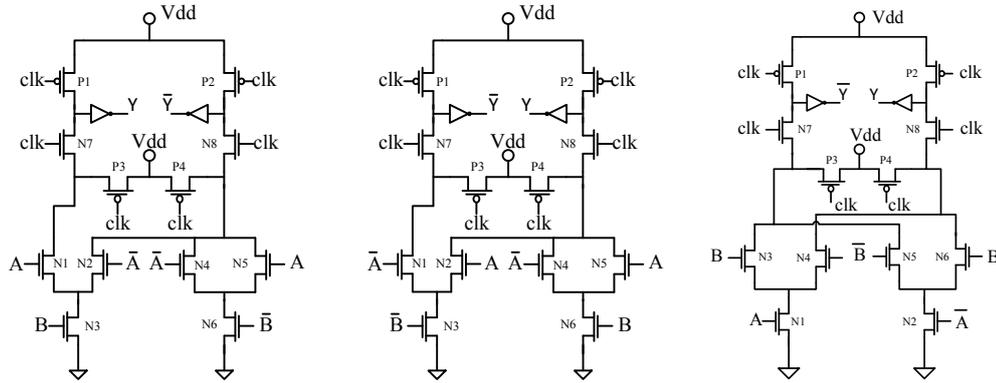


Figure 3.17. Early evaluation free iDDPL AND/NAND (left), OR/NOR (middle) and XOR/NXOR gate (right) with minimum number of transistors.

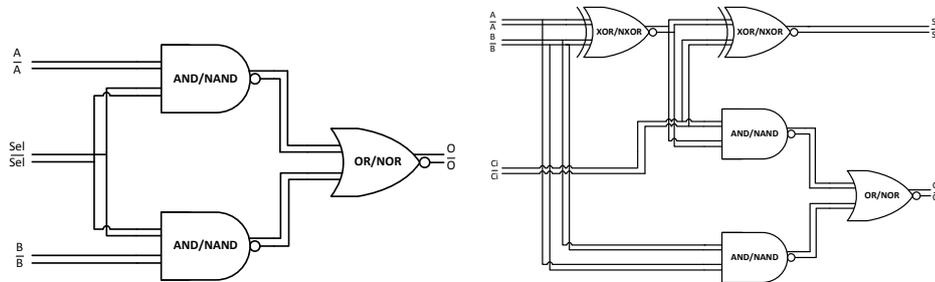


Figure 3.18. Logic gates implementation of balanced iDDPL multiplexer (left) and full adder (right).

This subset of Boolean operators can be adopted to build any logic function, relying on the fact that the propagation times of the differential signals are balanced in order to avoid fluctuation and the differential wires are insensitive on the capacitive mismatch thanks to the TEL encoding, whereas the interconnect resistance can be neglected provided that the wires are drawn short enough. As an example, in Fig. 3.18 the implementation of a multiplexer and a full-adder is shown. These gates are balanced on early evaluation being designed by interconnecting the above analyzed Boolean gates.

3.3.3 Combinational gates with capacitive load imbalance

In the model described in previous section no assumptions on the output capacitances were made. In this section we assess the impact of the mismatch of the differential output capacitances on the delay δ .

The load capacitance of a logic gate is composed of three main contributions: the drain capacitance of the gate itself C_d , the gate capacitance of the next stage C_g , and the capacitance due to the interconnect wire C_w . In submicron technologies, the main contribution is given by C_w [101], which is expected to increase with the technology scaling. Moreover each gate is characterized by the same input and output capacitance on each differential pair, thus the main source of imbalance on

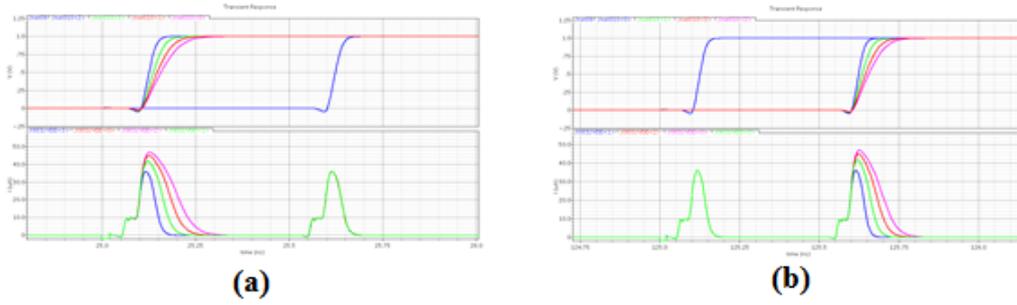


Figure 3.19. Waveforms of the differential signals at the output of a BUFF/INV gate for different MF (up) and current peaks in evaluation/postcharge (down).

the output capacitance can be considered as the interconnect capacitance.

In literature a number of DPL styles implementing the RTZ principle have been presented with a Domino cell topology (e.g. SABL, TDPL, DDPL, etc.): after the precharge phase, the input capacitances of the output inverters are both charged and the output signals are precharged to 0; during the evaluation, the evaluation network discharges only one of these capacitances to 0, forcing a $0 \rightarrow 1$ commutation on the output capacitance, whereas the other one keeps its state. This means that the dynamic consumption of the cell is dominated by the power consumption of the output inverters, which drive the output capacitances. If the differential capacitances are mismatched, the dynamic consumption is data-dependent. In the case of TEL circuits, the output inverters execute always the same transition, and the mismatch does not affect the energy balance of the circuit, at least in a first order analysis. However the mismatch of the load can affect the propagation times of the signals, modifying the value of the output delay δ_F and invalidating the model described in previous section.

In order to assess the impact of the output imbalance on the value of δ , an unbalanced iDDPL BUFF/INV has been simulated for different values of MF and for different values of fanout. In simulations, capacitances $C_{L1} = 1fF$ and $C_{L2} = MF \cdot 1fF$ have been connected on the Y and the \bar{Y} wires, respectively, in order to model the interconnect capacitances. The waveforms of the output signals as well as the current peaks at the evaluation and postcharge phases are plotted in Fig. 3.19. The figures refer to the two possible data commutations. From the figures, it is clear that the presence of unbalanced capacitances increases the value of δ when the non-asserted signal propagates along \bar{Y} . In the other case, δ_F is less than δ . This phenomenon increases for higher MF, as shown in Table 3.1.

The fluctuation effect due to the output imbalance can impact both symmetric and asymmetric gates, as confirmed by repeating the same simulations for an AND/NAND, an OR/NOR, and a XOR/NXOR gate. Results are summarized in Tables 3.2, 3.3, and 3.4.

In order to mitigate this effect, the most simple solution is enhancing the fanout of the gate. For this purpose we have repeated the simulations on the BUFF/INV gate by enhancing the drive strength of the output inverters. Results are reported in Table 3.5, where the value of $\Delta\delta$ is reported for different MF and different values

Table 3.1. Delta as a function of the input data in presence of different MF for a BUFF/INV gate.

MF	BUFF/INV	
	(0, 1)	(1, 0)
1	499.912	499.912
2	495.950	503.950
3	492.220	507.520
4	488.950	510.950

Table 3.2. Delta as a function of the input data in presence of different MF for a AND/NAND gate.

MF	AND/NAND			
	[(0, 1), (0, 1)]	[(0, 1), (1, 0)]	[(1, 0), (0, 1)]	[(1, 0), (1, 0)]
1	486.220	488.287	487.831	482.330
2	473.002	475.597	474.625	493.661
3	461.260	464.254	462.821	504.220
4	450.229	453.540	451.837	514.437

Table 3.3. Delta as a function of the input data in presence of different MF for a OR/NOR gate.

MF	OR/NOR			
	[(0, 1), (0, 1)]	[(0, 1), (1, 0)]	[(1, 0), (0, 1)]	[(1, 0), (1, 0)]
1	482.200	487.751	488.133	488.447
2	493.524	474.518	475.384	475.258
3	504.091	462.712	464.094	463.482
4	514.359	451.705	453.351	452.429

Table 3.4. Delta as a function of the input data in presence of different MF for a XOR/NXOR gate.

MF	XOR/NXOR			
	[(0, 1), (0, 1)]	[(0, 1), (1, 0)]	[(1, 0), (0, 1)]	[(1, 0), (1, 0)]
1	477.871	477.964	477.934	477.899
2	489.362	465.502	465.381	489.392
3	500.040	454.261	454.149	500.054
4	510.340	443.577	443.475	510.330

Table 3.5. Variation of the delay δ with the driving strength for different mismatch factors.

MF	$\Delta\delta[ps]$			
	INVX1	INVX2	INVX4	INVX8
1	0.1	0.1	0.1	0.1
2	-11	-6	-3.5	-2
3	-21.5	-11.6	-6.5	-3.5
4	-31.5	-17	-9.75	-5

of fanout.

As expected, when there is no mismatch (ideal case) δ is almost constant and $\Delta\delta \approx 0$. For moderate mismatches ($MF = 2$), a negative variation of 11ps is already visible for the minimum fanout cells. This value represents the 2.2% of the original δ (500ps). For higher values of MF , $\Delta\delta$ is negative and its absolute value increases, thus in order to guarantee at least that $\frac{|\Delta\delta|}{\delta} \approx 0$, the fluctuation of δ can be controlled using higher fanout cells with sufficiently big output inverters to ensure that $|\Delta\delta| \approx 0.01 \cdot \delta$. For example, in the case $MF = 2$ the INVX2 cell would be sufficient ($\Delta\delta = 6ps \approx 0.01 \cdot \delta$). In any case, the residual variation . A similar analysis can be performed for the other logic gates.

In general, the automatic place and route procedure is iterative, and the cells are automatically selected by the EDA processor from the standard-cell library in accordance to the value of the load capacitances which must be driven, in order to satisfy the timing constraints of the design. However, a residual fluctuation can occur after the routing, and must be controlled during the design phases in order to avoid that it adds up along the combinational path for any input data configuration. For this purpose, a back annotated analysis is useful to verify that the timing constraints are met; on the contrary, the nominal value of δ should be slightly changed.

3.3.4 Presence of mismatch variations

The third possible source of fluctuation in the design of iDDPL circuits is represented by the mismatch variations. The mismatch variations are the change of the parameters of the transistors located close to each other (e.g. length, widths, oxide thickness, etc.) due to the process spreading, which in the nanometer region can strongly impact the symmetry of a logic cell. In iDDPL combinational circuits, the variations of the parameters of the transistors in the evaluation network are a source of asymmetry of the logic cell, impacting the propagation times of the differential signals. This makes the previously presented delay model imprecise, and, more important, affects the timing specifications of the circuit.

With the notation $\Delta_{\mathbf{F}}$, we consider the random variable representing the actual delay at the output of a combinational gate, whereas δ_F is one possible realization. If δ is the nominal delay of the circuit, the timing requirement of an iDDPL circuit is met if :

$$Pr[\Delta_{\mathbf{F}} \leq \delta] = 1 \quad (3.25)$$

for all the internal differential nets of the circuit. The probability function of the variable $\Delta_{\mathbf{F}}$ can be described as a Gaussian curve, with mean $\mu = \delta_F$ and standard deviation σ .

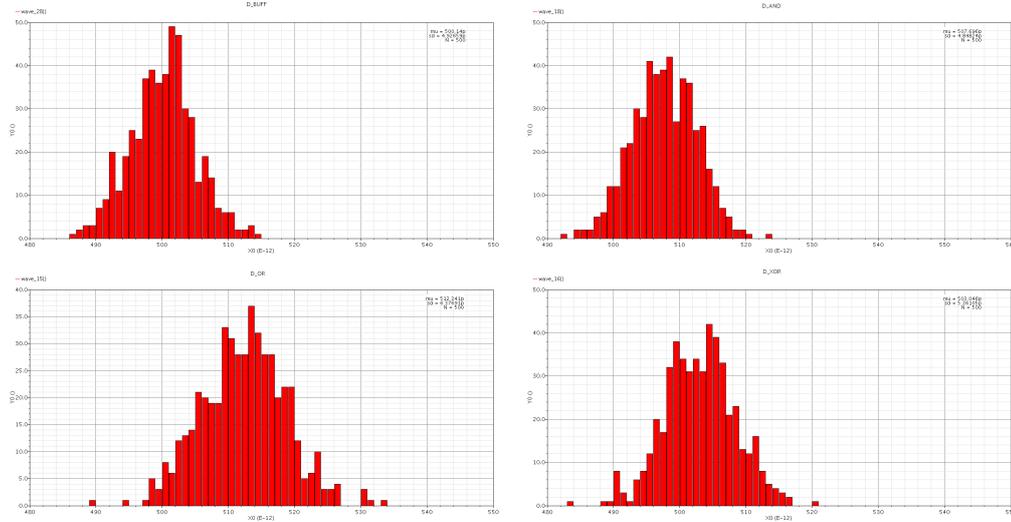


Figure 3.20. Distribution of the output delay δ for the combinational gates after 500 iterations of Monte Carlo simulations.

In order to calculate the impact of the mismatch variations of the transistors for each combinational gate, Monte Carlo simulations have been executed. Results are plotted in Fig. 3.20 for 500 iterations using a nominal δ equal to 500ps.

According to the results in figures, the average value is approximately 500ps, as expected, whereas σ is almost equal to 1% of δ ($\approx 5ps$), which is a very slight value. Thus we conclude that the condition $\frac{|\Delta\delta|}{\delta} \approx 0$ is always satisfied when $\Delta\delta < 0$ and fluctuation is very limited in the combinational gates even when transistors are highly mismatched.

As it will be shown in next paragraphs, the mismatch variations must be taken into account in order to preserve the functionality of the chip as well as guarantee a minimum level of security. For 99% of circuits δ is comprised between -3σ and $+3\sigma$, that is approximately between 485ps and 515ps on a single gate using the values found with Monte Carlo simulations. When the combinational gates are connected in a pipelined circuit, the variances add up coherently, impacting the condition on the maximum number of cascaded stages.

3.3.5 Validation of the model in some case study combinational gates

In this section we prove the accuracy of the theoretical model described in previous section by performing simulations on simple combinational case studies. We take as case study the standard AND/NAND gate, because every logic function can be built as a compound of AND/NAND gates. Results can be extended for the case of any other logic function.

Simulations have been performed in the nominal case and at standard temperature ($25^\circ C$), setting $\delta_A = \delta_B = 500ps$, clock frequency $f_{CK} = 100MHz$ and supply voltage $V_{DD} = 1V$. Furthermore the input signals have been delayed each other with $t_1 = t_2 \approx 120ps$. The aim of the simulations is to measure the variations of

Table 3.6. Output delay in the AND/NAND gates (in ps).

AND/NAND		$\delta_F^{0,0}$	$\delta_F^{0,1}$	$\delta_F^{1,0}$	$\delta_F^{1,1}$
No EE	Bal	422	400	420	375
	Unbal	453	354	460	418
With EE	Bal	649	357	526	524
	Unbal	688	315	571	567

the output delay in some combinational case studies and verify if results are in agreement with the model discussed in previous paragraphs.

A single combinational gate

The basic and the optimized AND/NAND gates have been compared in simulation under balanced and unbalanced load conditions. In presence of balanced capacitances of 1fF, the current pattern in the evaluation and postcharge phase for all input combinations and the clock signal are shown in Fig. 3.21

Current peaks are associated to the asserted and the non-asserted signals respectively, and are separated from a time interval equal to δ_F . For the basic cell the random variations of δ_F highlight the dependence of the current trace on the applied inputs. Moreover the fluctuation effect is visible being $\delta_F > \delta$ and therefore $\Delta\delta < 0$, as predicted by the model. On the contrary, for the optimized cell the current peaks are nearly superimposed (a slight deviation of δ_F less than 50ps is visible) and the relation $\delta_F < \delta$ holds for all data combinations.

Simulations were repeated by loading the gates with unbalanced capacitances on the complementary lines in order to understand the impact of a mismatched load on current traces. The output capacitances were chosen to be equal to 1fF and 4fF on the NAND and the AND output respectively, with a mismatch factor of 4, in order to simulate a very worst condition. Results are summarized in Table 3.6 for both cases and for both load conditions, and are in agreement with the model. Simulations were repeated for the OR/NOR and the XOR/NXOR gates, obtaining similar results.

A multi-stage combinational logic

The analysis is now generalized to the case of a combinational multi-level logic made up of five cascaded AND/NAND gates, implemented with the basic and the optimized AND/NAND gate (Fig. 3.22).

According to the analysis of delay model, the critical path (from the viewpoint of early evaluation effect) is associated to the data transition which causes the output delay of the gate i to be greater than its input delay. This particular data configuration just corresponds to the case $(A, \bar{A}) = (B, \bar{B}) = (0, 1)$ when the fluctuation of δ is maximum, as described by Eq. 3.8.

In order to obtain this particular data combination, the pipeline is built so that the AND (NAND) wire of a gate is connected to the asserted (delayed) wire of one input of the following gate, whereas the other input is $(0, 1)$. This way, the signals

processed at each port are always $(0, 1)$.

In Fig. 3.23 the evaluation and the postcharge current peaks are coupled according to the same color as in Fig. 3.22: the peaks in blue represent the evaluation/postcharge of the first gate in Fig. 3.22, whereas the peaks in violet are associated to the last gate.

The plot on the left in Fig. 3.23 refers to the current pattern of the logic suffering on early evaluation. The current peaks of the first gate exhibit a delay δ_F equal to 540ps, whereas the current peaks of the last gate exhibit a delay δ_F equal to 860ps, which is almost 60% greater than the original value of δ . Thus each stage is characterized by an output delay almost 80ps greater than its input delay. The relation in Eq. 3.26 holds for each gate ($\delta_F^0 = \delta$) :

$$\delta_F^i > \delta_F^{(i-1)} \geq \delta \quad (3.26)$$

It is worth noting that the output delay of the last gate is greater than the original delay of a quantity directly proportional to the number N of stages and to $\Delta\tau_{and}$. The maximum can be estimated by Eq. 3.27 and represents the worst case for this specific logic path in terms of fluctuation of δ :

$$\delta_F^{max} = \delta + \tau \cdot N > \delta \quad (3.27)$$

where τ is just the term $t_2 + \Delta\tau_{and}$ of Eq. 3.8.

On the contrary when the optimized AND/NAND gates with no early evaluation are used, the output delay, as depicted in the right plot of Fig. 3.23, gradually decreases as expected in a iDDPL combinatorial path:

$$\delta_F^i < \delta_F^{(i-1)} \leq \delta \quad (3.28)$$

and the maximum value of the output δ is always less than δ :

$$\delta_F^{max} = \delta - \tau \cdot N < \delta \quad (3.29)$$

According to these results, on a side the fluctuation effect degrades the timing margin of the TEL circuits, and on the other side introduces a data-dependence in the current pattern. It must be pointed out that even if the current traces of the TEL circuit are low-pass filtered by decoupling capacitances on the PSN of the chip, the skews due to the early evaluation add up when descending into a combinational logic netlist and the overall fluctuation can become critical, as depicted in Fig. 3.24, where the current traces have been filtered. Even if the delay δ can be designed to be in the range of a few hundreds of picoseconds in current technologies, the fluctuation can increase uncontrollably along a combinational network. There are many cases of apparently protected logic styles, which was successfully attacked exploiting the side-channel due to the early evaluation, also for skews in the order of few nanoseconds [48] [72].

We can conclude that in the AND/NAND implementation which suffers on the early evaluation effect the fluctuation introduced by a single gate actually adds up to the output delay, whereas using optimized gates the skew is equally distributed both on the asserted and the non-asserted lines and the value of the output delay

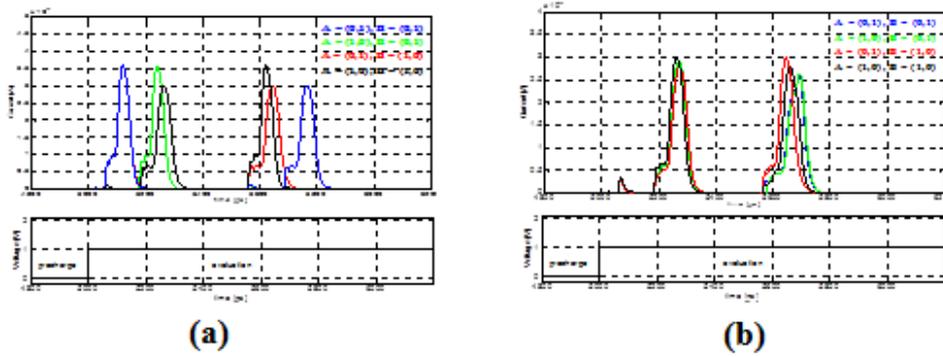


Figure 3.21. Superimposition of current traces for a basic (a) and an optimized (b) AND/NAND gate.

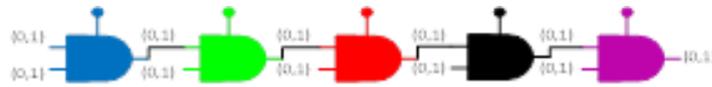


Figure 3.22. A combinational multi-level logic with 5 cascaded AND/NAND gates.

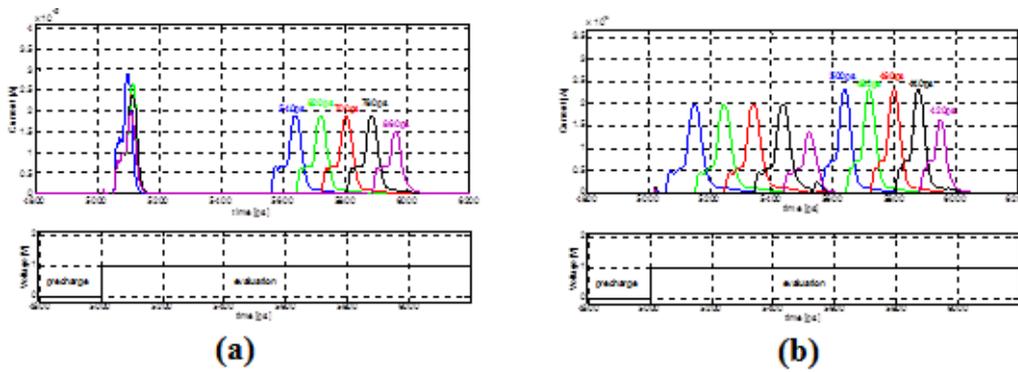


Figure 3.23. Superimposition of current traces for the multi level logic implemented with basic (a) and optimized (b) AND/NAND gates.

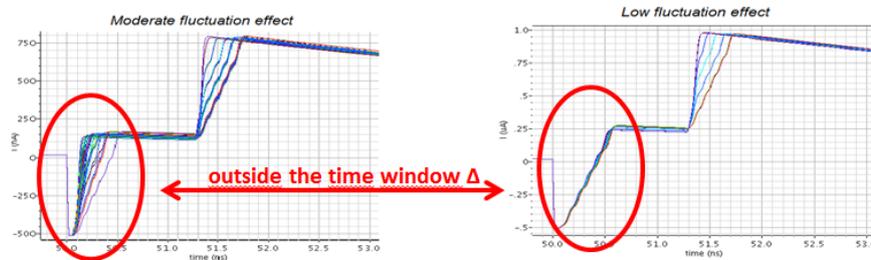


Figure 3.24. Superimposition of filtered current traces for the multi level logic implemented with basic (a) and optimized (b) AND/NAND gates.

δ_F is always lower than the initial value.

In general, the early evaluation must be avoided in order to preserve the timing requirements of the TEL circuit but also to prevent a data-dependent current pattern outside δ . With reference to fig. 3.24, the information leakage is detectable in a longer time period than the original δ , and a standard time-domain power analysis (e.g. CPA) can potentially exploit the time-mismatches to reveal information also outside the time window δ . Furthermore new enhanced power analysis techniques can exploit also the mismatches of the *Power Spectrum Density (PSD)* to detect information.

3.4 Design and characterization of an iDDPL sequential element

3.4.1 Main issues in the design of DPL sequential elements

Designing a balanced sequential element for a specific DPL style is not a trivial issue, because on a side it must be optimized in terms of area, power and delay, and on the other side it must be able to balance the instantaneous current consumption. Registers are maybe the most critical circuit parts with respect to security, because after the sampling phase, a datum is hold in the flip-flop for a time interval, and during this time interval the current consumption of the latch is proportional to logic value of the datum so that it is potentially visible by a side-channel attacker. This is true in particular in ASIC implementations, where the most part of leakage is usually detected in correspondence to the values processed by a register, but it can become critical also on FPGA if several memory cells are used in the design.

The philosophy beyond the design of a flip-flop for a particular DPL style is rather standard and it is based on three basic operations: first a specific differential signal pair is sampled at the clock rising edge; the signals are then held for a clock period; finally they are used to regenerate the original data-encoding in correspondence to the clock rising edge of the following cycle. In many cases the flip-flop is composed of two cascaded latches, where the first latch samples, whereas the second holds the signals and reconverts them into the original domain. A static CMOS Set Reset (SR) latch is very often adopted as slave-latch, thanks to its very simple structure where two NAND gates are cross coupled forming a feedback loop. However, in a CMOS SR-latch the differential NAND gates are unprotected when a capacitive imbalance occurs on the differential wires. For this reason, the flip-flop requires to be protected by a global masking scheme which has the purpose to de-correlate the power consumption of the flip-flop from the datum internally processed when the wires are mismatched.

However, recently it has been discovered that combining masking with DPL circuits does not ensure that the design is protected against PAAs [109]. Masking can improve the PAAs resistance of a dynamic dual-rail circuit, but cannot eliminate the physical leakage due to the differential mismatches; registers, which are critical because the datum is hold for an entire clock cycle, continues to be potentially vulnerable. In other words, sequential circuits must be inherently routing-insensitive

as well as, if not more than, combinational circuits.

As previously asserted, DPL flip-flops are typically arranged in two differential master-slave stages of cascaded latches. In this architecture the clock frequency is doubled in order to achieve the same throughput as conventional CMOS [72]. The drawbacks are an increased power consumption and a complex structure, which are both justified by a higher level of immunity against PAAs. In addition the setup and hold time of the latches must be appropriately evaluated.

In [88], a flip-flop for the WDDL style is presented. The circuit is composed of two differential CMOS flip-flops which work simultaneously. The clock signal is connected only to the sequential gates, precharging and evaluating at the same time. The differential signals propagate along the combinational path, producing a wave. WDDL suffers both from time and capacitive mismatches due to the difficulty of balancing the internal gates. Several improvements have been presented to solve these drawbacks, both at logic [76] and layout level [10] [129].

The first implementation of a SABL flip-flop (SAFF) [74] was a master-slave configuration in which a SABL inverter is the master device and a static CMOS Set-Reset (SR) latch is the slave device. In this implementation the differential capacitances at the output of the latches exhibit different charging and discharging times. More specifically if the SAFF flip-flop stores the same value for two or more consecutive cycles, the latch will not switch and will maintain its state: consequently, there is no dynamic power consumption in the static latch and the flip-flop is vulnerable to power analysis. In [93] the authors proposed a fully dynamic master-slave configuration which solves this problem. Even if the dynamic master-slave implementation of the SABL flip-flop effectively equalizes the power consumption, nevertheless it suffers from the common drawback of SABL gates which exhibit a power consumption dramatically dependent on the capacitive load mismatches on differential pairs. This dependence forces the designer to manually route each differential line in order to guarantee a perfect balance of the capacitances.

A similar implementation of master-slave flip-flop has been proposed for the TDPL style [22]. The slave device is a static SR latch, similarly to the first implementation of SAFF. Thus, also in this case the wires require an adequate balance in order to avoid any dependence on the current trace. Moreover, the flip-flop takes as input two synchronization signal in order to regenerate the three-phases differential signals. Routing two dynamic signals into different parts of a circuit represents an outstanding issue, because to avoid the cross-talk effects on the dynamic wires the interconnections must be adequately separated, leading to a big area overhead.

A similar topology has been used for the DDPL style. In [20], the authors presented a DDPL SR-latch-based flip-flop which has been used to build a cryptographic hardware implementation as a case study. It is composed of two cascaded DDPL latches, which are in turn composed of three stages: the first stage is a converter; the second stage is a static SR latch which executes the sampling of the data; the third stage is a converter which resets the complementary lines into the DDPL domain. Similarly to the SAFF and the TDPL flip-flop, in the DDPL SR-latch-based flip-flop the conversion of a dynamic signal into the static CMOS domain is not the best choice to guarantee a constant power consumption. Moreover this implementation results in a large number of transistors with a high power consumption.

In this section we propose a sequential circuit for the iDDPL style, which is fully

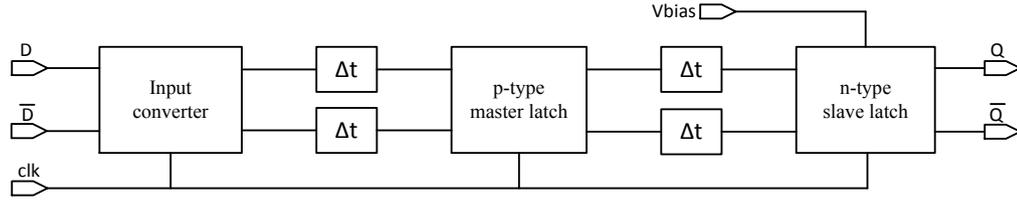


Figure 3.25. Block scheme of the iDDPL master-slave flip-flop.

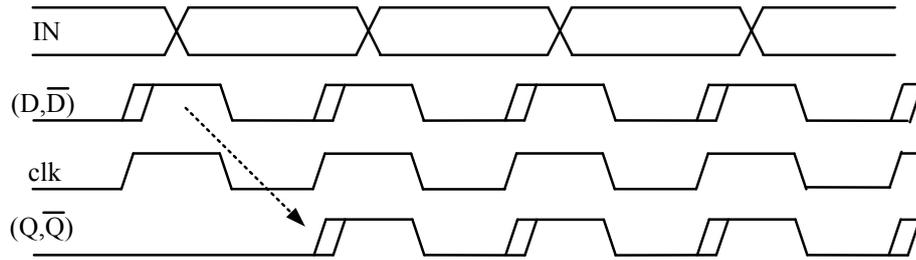


Figure 3.26. Timing diagram of the signal processed in the DDPL master-slave flip-flop.

compatible with the TEL data encoding and can be adopted for the circuit pipeline depicted in Fig. 2.7. The flip-flop will be tested with SPICE level simulations and its ability to balance the instantaneous power consumption will be assessed using the previously introduced metrics (i.e. NED, NSD and CV) in the real case of unbalanced load. The performances of the flip-flop will be compared to those of the aforementioned DPL flip-flops.

3.4.2 A master slave flip-flop for the iDDPL style

As described in previous section, an important property of iDDPL topology is that two independent half circuits can be identified in a gate, which are activated in turn according to the arrival times of the delayed dual-rail signals. The block diagram of the proposed master-slave flip-flop is presented in Fig. 3.25. It is composed of three latches and two inverting delay blocks Δt on both paths.

In Fig. 3.26 the timing diagram of the input/output signals is shown. Signal *IN* represents the input data of the iDDPL circuit encoded as a single-rail signal, which is converted in the TEL dual-rail pair (D, \bar{D}) by a converter at the input of the whole iDDPL circuit (not shown). The signals (D, \bar{D}) are then regenerated after a clock cycle, as in other dynamic flip-flops.

The voltage V_{bias} has the purpose of controlling the delay element in the slave latch. The single blocks of the flip-flop will be described and explained in more detail in the following paragraphs.

The input converter

The first stage is a converter, which is a self-timed pulse clock latch. The circuit is shown in Fig. 3.27. Basically it makes the conversion from the TEL data encoding

to the RTZ domain. Similarly to the iDDPL combinational cell template, the architecture of the gate is differential and it is composed of two independent halves. The working principle of one half circuit is shown in the same figure: when clk is low, P1 is closed, the input capacitance of the inverter at internal node v is precharged to V_{DD} , and the output line is forced at 0.

The input of the cell is a TEL-domain signal pair which is inverted and sampled by the transmission gate N3-P3 controlled by a CMOS XOR gate. The output of the CMOS XOR gate, which is implemented with a balanced differential architecture, is a pulse clock which drives the transmission gate: when $clkb$ is high, which is possible only during the dynamic period δ in which D and \bar{D} are different from each other (evaluation phase), N3-P3 (N4-P4) are closed and the datum is sampled. Only one of the two differential paths is activated through the transmission gates, that is the half circuit connected to the asserted signal, whereas the other stays at zero because the rising edge of the delayed signal comes after the falling edge of the pulse clock $clkb$. The capacitance C is discharged at the arrival of the asserted signal, and the output goes to V_{DD} , whereas the capacitance C keeps its charge for the whole evaluation phase for the delayed signal. No transition occurs on the other half circuit, and the input capacitance of the inverter keeps its charge.

The converter works like a dynamic latch in which the information is sampled by a transmission gate and the charge is hold on a capacitance for an half period. The timing diagram of the signals is shown in Fig. 3.28. It is worth noting that the static XOR gate is symmetric and balanced (i.e. it switches twice in each cycle), and it does not consume static power (Fig. 3.29). Furthermore, the output differential signals are in the RTZ domain for the following elaboration.

The master latch

The cascaded master-slave latches in Fig. 3.25 work similarly as the master-slave SABL flip-flop [123] on the two clock phases, but are implemented using a iDDPL circuit template which allows to avoid the internal feedback loop due to the sense amplifier.

The p-type master latch is shown in Fig. 3.30 together with the working principle. It takes as input the RTZ signals converted by the previous stage; these signals are delayed and negated by the inverting delay line Δt , and sampled at the falling edge of the clock through the pass-transistor P3. This way the charge on the input capacitance C at node v is stored for all the following half period, in which the input converter and the slave latch are in the discharge phase. The output of the latch is the original RTZ signals shifted of an half period, as described in Fig. 3.31 where the time diagram of the processed data is shown.

The slave latch

Actually the n-type slave latch is the iDDPL converter depicted in Fig. 3.32. Also the working principle of the circuit is depicted. The circuit has the aim of re-converting the RTZ signals elaborated by the previous stages into the TEL domain, shifting the signals of another half period.

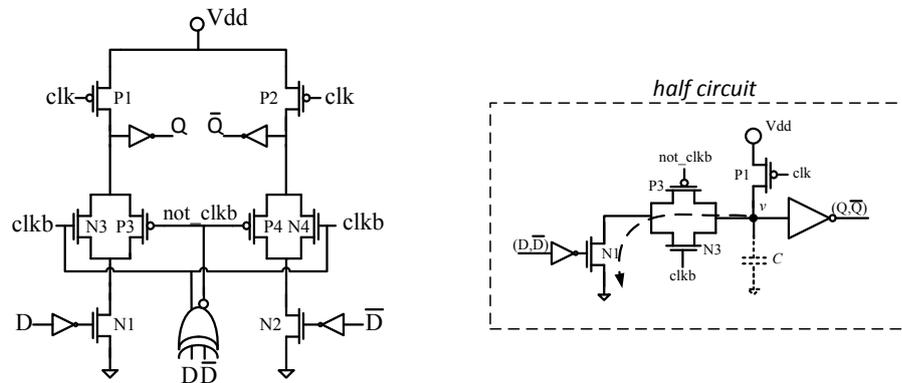


Figure 3.27. Scheme of the input converter (left) and working principle (right).

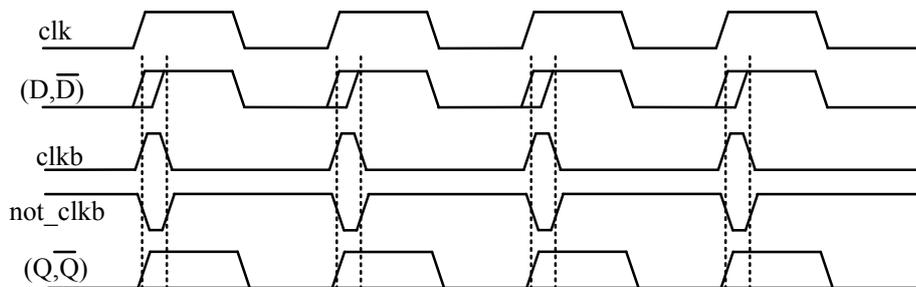


Figure 3.28. Timing diagram of the signals elaborated by the input converter.

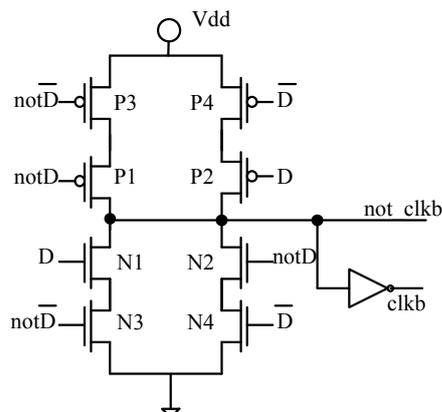


Figure 3.29. Scheme of the differential CMOS XOR gate which controls the transmission gates of the input converter.

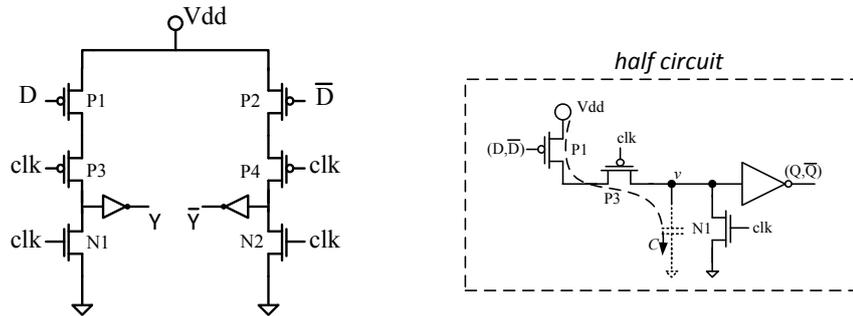


Figure 3.30. A p-type master latch and working principle of a differential half circuit.

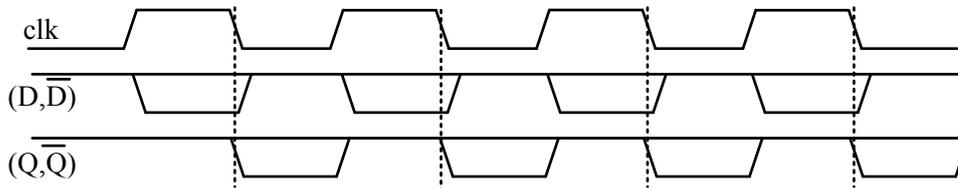


Figure 3.31. Timing diagram of the signals elaborated by the master latch.

The latch takes as input the RTZ data and the clock signal, and internally generates the δ -delayed clock signal clk_d through the Δ -block element. According to the working principle as well as the timing diagram reported in Fig. 3.33, the RTZ signals are first delayed and negated by the Δt block, then they are sampled on the rising edge of the clock signal. Then at this time instant only one half circuit is activated and the capacitance C on this half is discharged, whereas the other half circuit must wait for the rising edge of clk_d (after the original time delay δ).

The task of the Δt blocks in Fig. 3.25 is both inverting the signals and fixing the hold time at the input of the latches by increasing the precharge phase duration [93]. They are implemented with an odd number of stages of cascaded CMOS inverters which aim at inverting the signals and adding a sufficient delay time at the same time. For our design three minimum inverters are sufficient for meeting the requirement on the minimum hold time. Note that for circuits in which the clock skew is critical, the blocks Δt can be also replaced by current starved inverters and the static voltage required for generating clk_d inside the Δ -blocks can be also routed for the blocks Δt . This helps to prevent early evaluation errors due to clock skews effects, which in the dynamic circuits can be critical, with no area overhead, but increases the fanout of the clock distribution network.

Since the output lines of the iDDPL master-slave flip-flop exhibit a delay with respect to each other which is just equal to the original δ , the proposed memory element works like a dynamic flip-flop (with a delay equal to one clock-period) in which the data integrity is regenerated at the output with the same speed penalty as in the other dynamic flip-flop with respect to a static implementation. Furthermore, the latency of a datum in the storage circuitry is of one clock cycle.

An improved architecture of the input converter

The most critical block in the flip-flop is certainly the input converter, which must be as balanced as possible in order to avoid any data-dependence of the instantaneous current trace of the flip-flop. More specifically, the output signals are in the RTZ domain, therefore the interconnection of this gate with the master latch must be adequately controlled in order to guarantee a perfect balance.

Moreover, the input converter fixes the setup time of the flip-flop, which must be as low as possible in order to have a sufficient number of combinational stages. In the circuit shown in Fig. 3.27, the presence of a static gate at the input, that is a differential CMOS XOR gate, is not the best solution. A more balanced circuit topology is depicted in Fig. 3.34.

This circuit is symmetric and fully dynamic, and does not require the usage of static gates which have internal unbalanced parasitic capacitances. Moreover, the number of transistors is also reduced, and this allows to have a lower area occupation and reduce the setup time of the flip-flop.

3.4.3 Simulation and comparison of some DPL 4-bit registers

In this section the performances of the iDDPL flip-flop are tested on a case study circuit. The circuit is a 4-bit register which is used also in the iDDPL implementation of the SERPENT-block.

The circuit has been designed using different logic styles, in order to have a comparison: CMOS, WDDL, SABL, TDPL, DDPL, and iDDPL. The electrical schemes of the flip-flops under test have been found out from the literature. The CMOS register has been taken from the tech library. In order to have a fair comparison, all transistors have been designed with minimum sizes ($L = 65\text{nm}$, $W = 135\text{nm}$) so to guarantee at the same time the lowest occupation of area and the correct working of the logic. All the Domino inverters in the flip-flops and the XOR gate in the iDDPL flip-flops have been sized using the values for the minimum fan-out found in the technology library.

The circuit is simulated in the real case of unbalanced load: the output capacitance C_{L1} on a rail is fixed at 1fF, whereas C_{L2} has been changed in the range 1fF to 5fF with a step size equal to 1fF for the non-asserted lines, so to simulate different levels of mismatch. Again, a 1V supply voltage and a 10 MHz operating frequency are used. In order to simulate the cells in a real operating condition, the falling/rising edges of the clock and the data inputs were set to 20ps, and in the design all signals were driven by buffers from the CMOS tech library. A dynamic delay $\delta_i = 500\text{ps}$ for DDPL input signals has also been adopted, and simulations were executed in the nominal case at standard temperature (25°C). The input data were chosen in order to consider all possible transitions inside the logics.

The results (i.e. the time samples representing the current adsorbed for each clock cycle under unbalanced load conditions) are reported in Fig. 3.35, where the superimposition of the current traces in a clock cycle is depicted for each of the registers under analysis for the typical case of mismatch factor equal to 3 as a representative case. The traces were sampled with a time period equal to 1ps, which for a clock cycle corresponding to 100ns leads to 10^5 time samples.

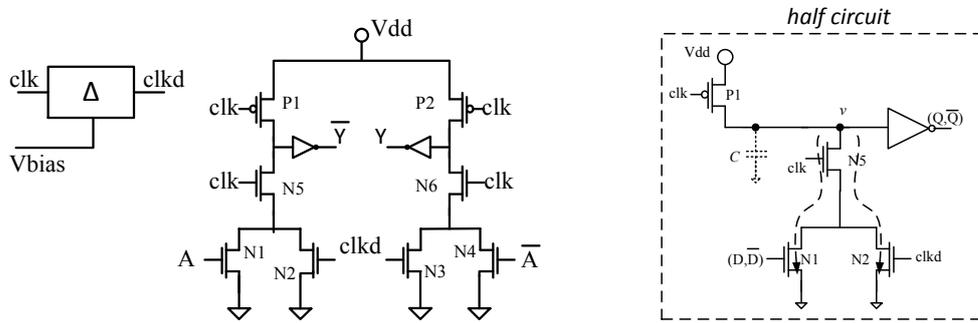


Figure 3.32. A n-type slave latch and working principle of a differential half circuit.

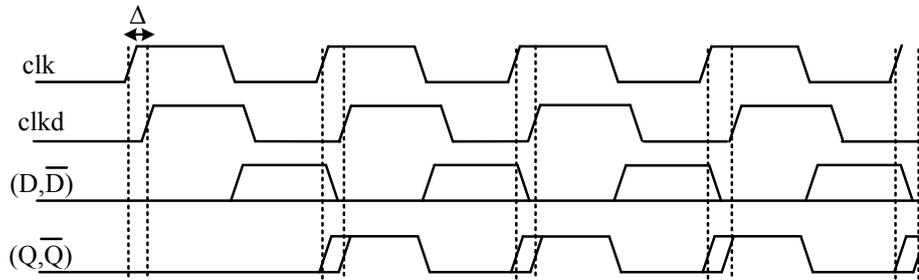


Figure 3.33. Timing diagram of the signals elaborated by the slave latch.

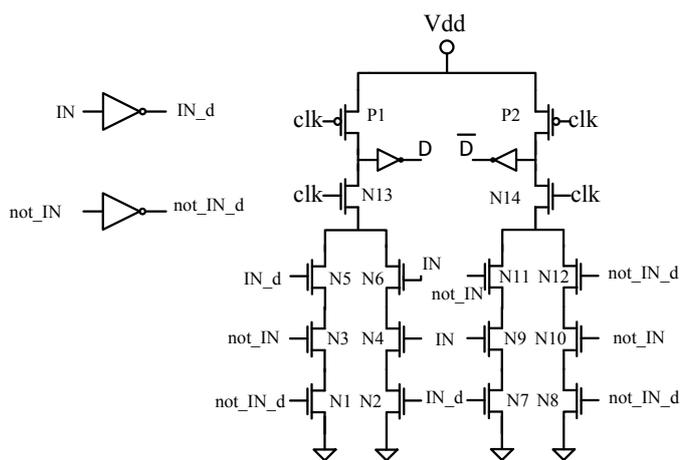


Figure 3.34. Scheme of the improved input converter.

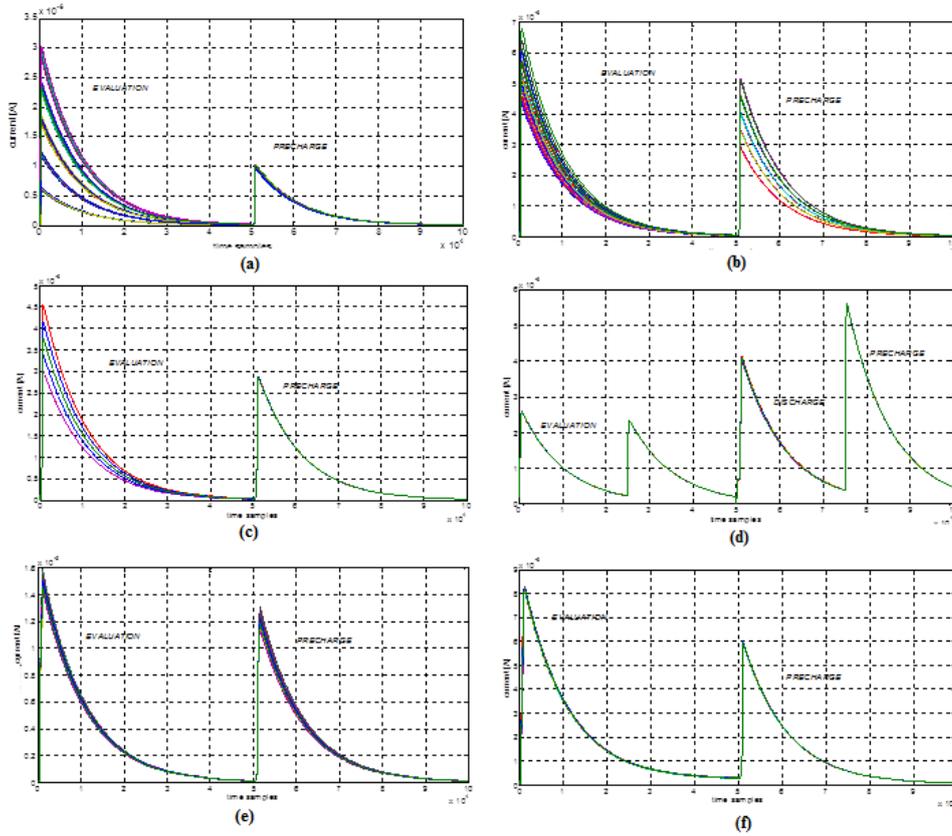


Figure 3.35. Superimposition of current traces for all possible input data in a CMOS (a), WDDL (b), SABL (c), TDPL (d), DDPL (e), and iDDPL (f) 4-bit register (MF = 3).

In Fig. 3.36 the curves of the coefficient of variation as a function of the time samples are shown. The figures capture the normalized variance of the current traces for each time sample. As expected, the standard CMOS register exhibits a high unbalance for the whole evaluation phase. The coefficient of variation grows according to the increase of the MF of the differential capacitances during the evaluation phase for WDDL and SABL flip-flops, amounting to high values already for a low MF, and this cannot be acceptable. For the WDDL register an increase also occurs during the precharge phase.

On the contrary the variance of the current traces is very low in TDPL and DDPL flip-flops for the whole clock cycle, confirming the benefit of the three-phase and delay-based encoding. For these logic styles the CV remains almost constant regardless of variations of the mismatch factor. This confirms that TDPL and DDPL exhibit an exceptional power-balance even in presence of an extreme capacitive mismatch. The peak in the amplitude of CV is a transient effect which depends on the commutation of the clock and is not data-dependent, therefore it can be neglected.

The iDDPL flip-flop exhibits a residual information leakage only in the time interval δ_i , as shown in Fig. 3.37. During this time window a slight data-dependence of the current still remains, and the amplitude of the CV curve depends on the

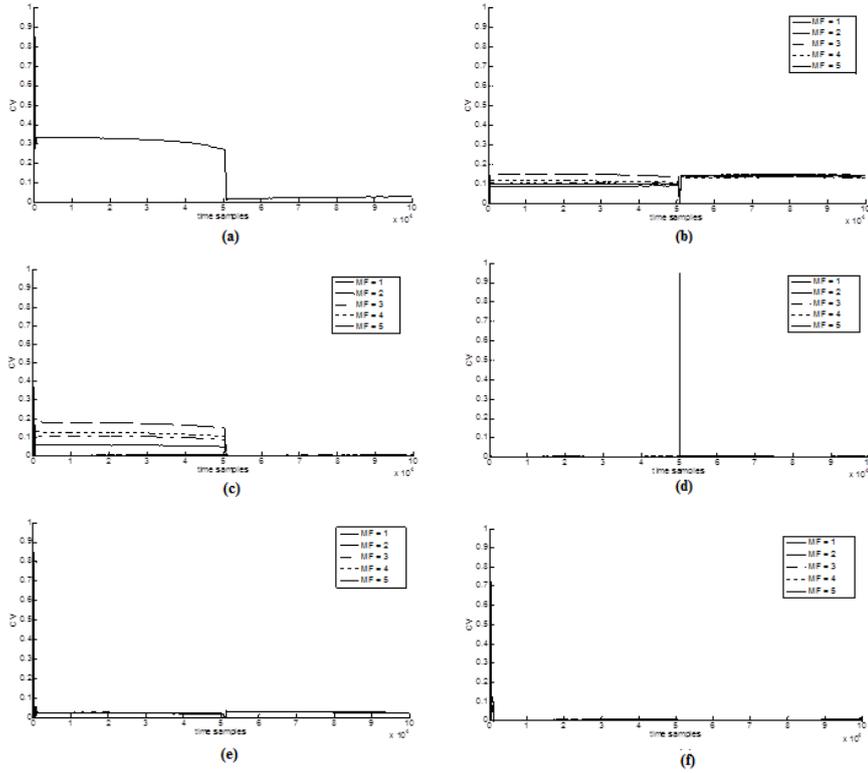


Figure 3.36. Coefficient of variation (CV) for a CMOS (a), WDDL (b), SABL (c), TDPL (d), DDPL (e), and iDDPL (f) 4-bit register as a function of the time samples in a clock cycle under different mismatch factors.

mismatch factor. As stated in previous section, the time period δ_i does not exceed the original δ . In simulations it is equal to 500ps, and in current technologies it is in the order of some hundreds of picoseconds (but always higher than the setup time). On order to capture these leakage points, a resolution higher than 2GSamples/s is required. This sampling rate can be reached using oscilloscopes with a high bandwidth, which are typically very expensive to be adopted in a low cost measurement setup for PAAs [20].

Results demonstrate that in RTZ-based schemes, where the logic state of the flip-flop is kept by a feedback structure (i.e. SR or sense amplifier latch), a strong dependence of the current traces on the processed data can be detected, and this causes the enlargement of the duration of high CV amplitudes. In these circuits the leakage is distributed in a time window comparable to the clock period and easily detectable.

The dynamic working principle of the proposed flip-flop allows to exploit the benefits of the delay-based data encoding, in which data are processed in a time period that is long enough to sample the TEL signals, but at the same time too short to detect the state of the latch. Therefore the side-channel is restricted and hidden from a practical power analysis attack, unlike the RTZ-based circuits in which the data-dependence is extended for a longer period of time.

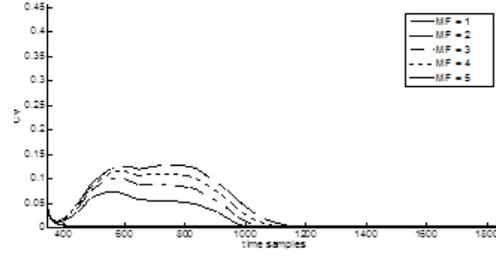


Figure 3.37. Screenshot of the Coefficient of Variation for the master-slave iDDPL 4-bit register around the delay time δ under different mismatch factors.

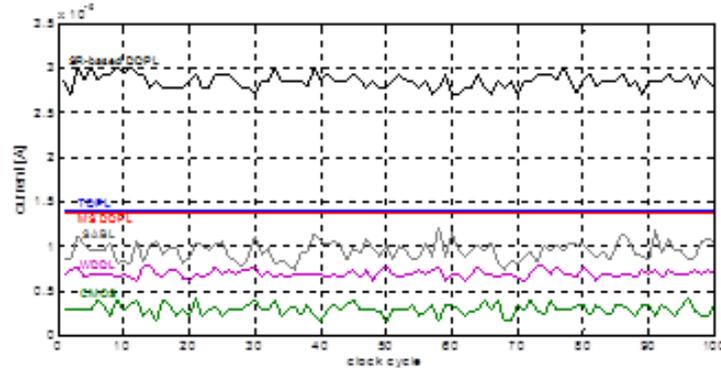


Figure 3.38. A comparison of the distribution of the average current for 100 clock cycles for the DPL flip-flops under test ($MF = 3$).

In Table 3.7 the values of the measured power-balancing metrics for the registers under test are reported for different mismatch factors. The average correlation coefficient CV_{AV} represents the mean of the correlation coefficient during the evaluation phase, where the data-dependence is stronger.

In Fig. 3.38 the distribution of the average currents I_{AV} is depicted for a number of clock cycles (i.e. 100) in the case of mismatch factor equal to 3. The figure shows a very low variance around the mean for the TDPL and iDDPL configurations, which are almost superimposed. Finally, in Table 3.8 the performances of the DPL flip-flops are summarized for the case of high mismatch factor.

The proposed iDDPL flip-flop improves the previously published implementation in terms of area overhead (which has been measured with respect to the area of the CMOS flip-flop used as comparison) and power-balance. The iDDPL flip-flop overcomes also SABL and WDDL in terms of NED, NSD and CV_{AV} , which are extremely slow ($< 1\%$) and comparable to TDPL. Obviously, an area penalty must be taken into account to enhance the power-balance of the circuit.

Moreover, as the unbalances of the internal wires were not considered in simulations, the performances of the proposed flip-flop are underestimated with respect to the TDPL implementation, where the routing of the differential wires is constrained by the presence of the SR slave latch, which may introduce time or capacitive mismatches, and the presence of an additional dynamic signal.

Table 3.7. Power balancing metrics for the registers under test.

MF	NED[%]	NSD[%]	E_{AV}[%]	CV_{AV}[%]
CMOS				
-	60.95	21.98	29.705	0.320
WDDL				
1	39.48	12.10	88.933	0.101
2	37.45	10.93	91.074	0.090
3	39.31	10.66	95.407	0.102
4	34.76	11.03	99.260	0.115
5	42.05	12.27	102.301	0.147
SABL				
1	0.21	0.04	61.436	0.004
2	11.50	3.25	66.220	0.056
3	20.72	6.10	69.722	0.101
4	28.04	7.51	74.105	0.120
5	34.18	10.92	76.799	0.170
TDPL				
1	0.98	0.18	123.428	0.003
2	0.81	0.19	131.586	0.002
3	0.77	0.18	139.438	0.002
4	0.63	0.13	147.405	0.002
5	0.68	0.15	155.309	0.002
DDPL				
1	10.77	2.97	268.461	0.028
2	10.39	2.54	278.126	0.023
3	10.10	2.77	285.391	0.025
4	9.91	2.60	293.290	0.024
5	9.65	2.69	301.288	0.024
iDDPL				
1	0.74	0.15	120.911	0.004
2	0.59	0.13	128.950	0.005
3	0.62	0.13	136.881	0.004
4	0.53	0.12	144.844	0.005
5	0.45	0.11	152.782	0.006

Table 3.8. Performances of the DPLs in terms of occupied area and average energy under high output mismatch (MF = 3).

	# MOS	Area[μm^2]	Area increase	NED[%]	$E_{AV}[fJ]$
WDDL [88]	66	52	x2.7	39.31	95.407
SABL [93]	44	39	x2.1	20.72	69.722
TDPL [22]	58	52	x2.7	0.77	139.438
DDPL [20]	126	113	x5.9	10.10	285.391
iDDPL	66	61	x3.2	0.62	136.881

3.5 The prototype iDDPL library

In this section the iDDPL standard-cell library used to build the SERPENT-block is presented. The timing constraints of the micropipeline iDDPL circuit are discussed, and the condition on the maximum frequency and the maximum number of stages for the critical path are calculated taking into account mismatch variations. The issues regarding the layout of the logic gates which compose the library are also discussed. Furthermore the cells have been characterized in terms of area, energy, and delay. The energy balance have been measured using NED in condition of extremely unbalance load (MF = 4), in order to cover any possible realistic situation.

3.5.1 Architecture of a micropipelined iDDPL circuit

A central point for the design of an iDDPL circuit is to route the static voltage wire V_{bias} to control the delay of the current starved inverter. With reference to Fig 2.7, the control signal $ctrl$ is implemented through the static voltage V_{bias} , which must be routed into each converter in the circuit. The input converters have the purpose to generate the valid data in the TEL domain; similarly, the slave-latch of the flip-flops re-converts the signals from the RTZ domain into the TEL domain by using a converter with the starved inverter.

Routing a static voltage is simpler than routing a dynamic signal, given that the latter may generate additional cross-talk effects on the nearest dynamic wires, therefore no additional area overhead is expected in the layout of the circuit, and a more efficient control is allowed.

Actually the routing of such static voltage signal could be avoided by generating V_{bias} locally (using two diode-connected MOS transistors), or by setting $V_{bias} = 0$ and sizing P2 appropriately (in this case a non-minimum gate length has to be used for P2), but this would require more area and a more complex circuitry to control the variations due to process mismatches.

Mismatch variations may impact the timing constraints on the micropipeline, as it will be discussed in next paragraphs. On the contrary, process variations may impact the relation between δ and V_{bias} , as it will be shown in next chapter on a specific case study. A solution could be inserting a *Delay Locked Loop* outside the circuit where the static voltage V_{bias} is generated, before it is distributed to the circuit, or alternatively inserting an analogue circuitry as for example a bandgap reference; this way it is possible to provide a fixed stable voltage to all sequential gates, reducing the impact of process variations.

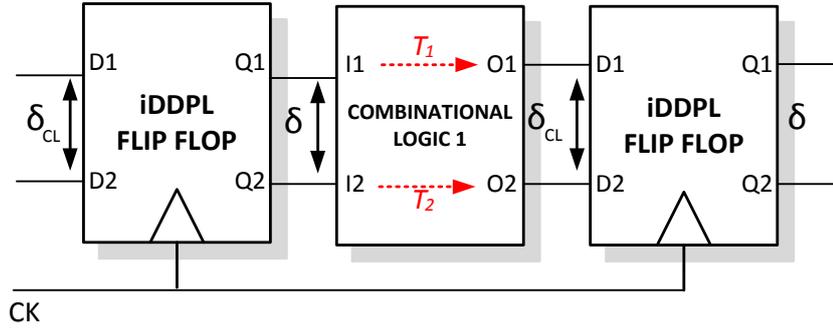


Figure 3.39. Variation of the delay δ along combinational logics.

3.5.2 Analysis of the timing constraints

In this section the timing parameters of an iDDPL circuit are discussed and calculated. According to the timing constraints defined in the previous chapter for TEL circuits, the iDDPL flip-flop must have the property of regenerating the original delay δ at the end of a combinational path, where the final δ_{CL} is less than the original δ if the combinational network is well designed.

Propagation time of a combinational logic

An important parameter to assess the performance of an iDDPL combinational logic circuit is the propagation time of δ , generically defined in Chapter 2 as the difference between the delay δ_{IN} of the differential signals at the input of the logic path and the δ_{OUT} of the differential signals at the output, and indicated as $\Delta\delta$.

According to the equations presented in previous paragraphs, the value of $\Delta\delta$ of a logic gate strongly depends on the delays t_1 and t_2 between the rising edge of the input/output asserted signals and the input/output non-asserted signals, respectively. For example, in the case of an early evaluation free AND/NAND gate, the maximum value of $\Delta\delta$ has been calculated to be equal t_1 . It must be noted that t_1 is the delay between the asserted signals of the input pairs; it represents the sum of the delays introduced by the logic gates which come first along the pipeline, and thus it approximately depends on the time constants τ_1 and τ_2 of these gates.

With reference to Fig. 3.39, let us suppose that the combinational logic 1 is composed of N cascaded logic gates; in figure, I1 indicates the asserted signal at the input of the path, and O1 is the asserted signal at output of the path; similarly, I2 and O2 are the non-asserted input/output signals. In general the overall propagation time T_1 between I1 and O1 is given by the sum of all the $\tau^{(i)}$ encountered along the combinational path:

$$T_1 = \sum_{i=1}^N \tau^{(i)} \quad (3.30)$$

with $i = 1, \dots, N$ and $\tau^{(i)} = \tau_1, \tau_2$, according to the number of active transistors in the evaluation path of a single gate which depends on the value of the input data; a

similar relation can be obtain for the propagation time T_2 between I2 and O2:

$$T_2 = \sum_{j=1}^N \tau^{(j)} \quad (3.31)$$

with $j = 1, \dots, N$ and $\tau^{(j)} = \tau_1, \tau_2$.

According to Fig. 3.39, it is clear that the asserted and the non-asserted signals propagate in the combinational path with separate propagation times. If the logic gates in the pipeline are early evaluation free and $\Delta\delta_F > 0$ for each logic gate of the path, we know that $T_2 < T_1$ and $\Delta\delta_{CL} > 0$. This is due to the fact that the postcharge phase is asynchronous with the clock because occurs after a time interval that may vary from δ_{MIN} to δ_{MAX} with respect to the precharge along the combinational path. Furthermore, in order to preserve the functionality at the input of a combinational logic gate, the cells must be interconnected so that the relative delay between the asserted signals at the input is always less than the value of δ : $T_1 < \delta_{IN}$; if this relation is not satisfied, the logic gate cannot evaluate correctly the data.

We have already shown that the propagation time $\Delta\delta$ impacts the maximum number of combinational stages. However, as it will be shown in next paragraph, the dependence with the working frequency of the circuit is more difficult to determine.

Condition on the maximum frequency of the circuit

In this paragraph we calculate the condition on the clock frequency of the TEL circuit.

According to the tuning range curve presented in Fig. 3.8, the maximum value of δ is bounded by the working frequency of the circuit; thus, under the condition that the value of the δ_F inside the combinational path is always less than δ , it seems that the maximum clock period is equal to the double of δ . However this is a wrong conclusion. In fact, the relation $\delta_{MAX} = \frac{T_{CK}}{2}$ is only theoretical, because it does not take into account the propagation delays T_1 and T_2 described in previous paragraph.

In Fig. 3.40, the waveforms at the input/output of a combinational logic are represented. The value T_{CK-Q} represents the delay between the clock and the asserted signal at the output of the flip-flop, and it is determined by the slave latch of the flip-flop. It can be noted that the condition $\Delta\delta = \delta_{IN} - \delta_{OUT} > 0$ corresponds just to the condition $T_1 > T_2$, as already stated. Furthermore, when the clock period is reduced, the TEL data encoding elapses before $\frac{T_{CK}}{2} = \delta$.

Assuming that the waveforms (O1, O2) in Fig. 3.40 are the signals at the output of the critical path, the maximum frequency can be calculated in this way. When the clock period is reduced, if $T_{CK} < 2 \cdot (T_{CK-Q} + \delta + T_2) = T'_{MAX}$, the discharge falling edge of the clock comes first than the rising edge of the non-asserted signal O2 and the postcharge phase is lost: in this case, the circuit continues working but with a RTZ-like data encoding on the signal (O1, O2) and a degraded margin of security. If the clock period is reduced even more, the other differential signals of the combinational network start to behave like RTZ differential wires. Thus, reducing the clock frequency down to T'_{MAX} , erases the postcharge phase and forces the circuit to work like a RTZ logic, with a reduced level of security. When the clock

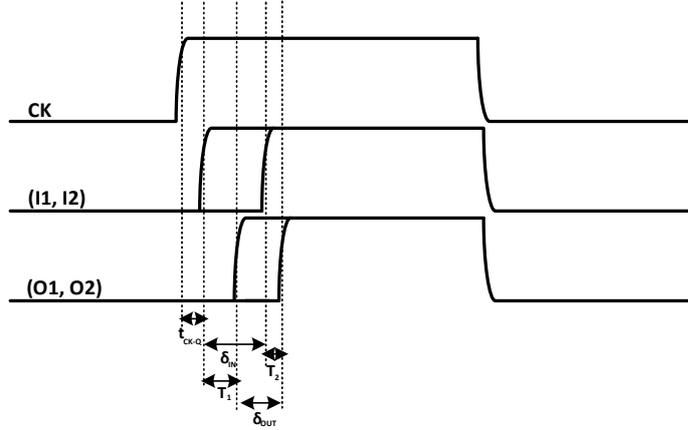


Figure 3.40. Timing of the signals at the input/output of a generic combinational path.

period is decreased to such an extent that also the rising edge of the asserted signal is lost, the evaluation starts to fail, and the functionality of the circuit is compromised. Therefore, the minimum clock period which ensures the correct functionality is given by the relation $T_{CK} < 2 \cdot (T_{CK-Q} + T_1) = T_{MAX}$.

In summary, once the nominal δ is fixed, three different working regions can be distinguished, and the value of the frequency is:

$$f_{CK} \begin{cases} < f'_{MAX} & TEL \text{ working} \\ \in [f'_{MAX}, f_{MAX}] & RTZ \text{ working} \\ > f_{MAX} & no \text{ working} \end{cases} \quad (3.32)$$

where:

$$f'_{MAX} = \frac{1}{2 \cdot (T_{CK-Q} + \delta + T_2)} = \frac{1}{2 \cdot (T_{CK-Q} + T_1 + \delta_{OUT})} \quad (3.33)$$

$$f_{MAX} = \frac{1}{2 \cdot (T_{CK-Q} + T_1)} \quad (3.34)$$

It is clear that the condition on the security margin of the circuit impacts the maximum working frequency of the TEL encoding, but does not impact directly the functionality of the chip. In other words, the presence of δ provides the designer with an additional degree of freedom to enhance the level of security of the chip, without impacting the maximum working frequency with respect to a RTZ implementation. The values t_{CK-Q} and $T_{1,2}$ are technology dependent.

Setup time of the flip flop

According to the analysis done in previous chapter for TEL circuits, the propagation times $\Delta\delta$ reduces along the combinational path. The minimum value allowable for the output delay δ_{CL} is determined by the number of cascaded stages, and depends on the setup time of the flip-flop. The proposed master-slave flip-flop samples a datum during the delay time δ at the rising edge of the clock, and regenerates it at the rising edge of the clock of the next cycle, similarly to other DPL flip-flops,

with a latency of one clock cycle. The delay time δ_{FF} at the output of the flip-flop is just the original delay δ (more precisely it is equal to $\delta - \delta_{CK-Q}$), because of the presence of the delay element in the slave latch which regenerates the original data in the next clock cycle. This is true only if the delay δ_{CL} is greater than t_{su} , otherwise the input converter cannot recognize the data.

The setup time of the flip-flop defines the functionality margin of a iDDPL circuit and represents the minimum delay time δ_{MIN} which allows the flip-flop to correctly sample a TEL differential pair. It must be noted that for a dynamic circuit, defining a hold time does not make sense, because the input datum is synchronized with the clock thanks to the presence of the input converter, thus a flip-flop samples a datum at a certain clock cycle and regenerates it at the following cycle, and during this period the input signals cannot change.

According to the timing analysis done in previous sections, the combinational circuits must be designed in order to guarantee that δ_{CL} is always less than δ . Thus, the value of δ_{CL} for each combinational logic must be within the interval $I = [t_{su}, \delta]$, where $t_{su} = \delta_{MIN}$ is defined as the setup time of the flip-flop, whereas $\delta = \delta_{MAX}$ is obviously the original delay which is chosen to guarantee a certain level of security of the logic.

The setup time of the master-slave flip-flop is calculated with post-layout simulations where all the parasitic elements are taken into account:

$$t_{su} \approx 90ps \quad (3.35)$$

This values poses a limit on the maximum number of stages of a combinational logic before the flip-flop so that the circuit works using a TEL data encoding. This parameter has been used to build the Table 2.4, where a road map for TEL circuits implemented with iDDPL gates has been defined.

The propagation time t_{CK-Q} of the designed flip-flop is about 103ps. The proposed architecture also exhibits a good robustness to clock skews thanks to the internally generated self-timed pulse clock. It must be noted that with the improved circuit topology of Fig. 3.34, the setup time can be also reduced down to 85ps.

Condition on the maximum number of stages in a pipeline

In this paragraph we calculate the condition on the maximum number of cascaded stages that can be considered for a combinational path, by taking into account the mismatch variations of the logic gates.

Consider again the minimum value of δ_{CL} at the output of the critical path; the timing constraints are satisfied if:

$$\delta_{min} > t_{su} \quad (3.36)$$

The minimum value of δ at the output of the combinational path is given by the difference between the nominal δ and the sum of the maximum propagation delays of the combinaional gates:

$$\delta - N \cdot \Delta\delta_{max} > t_{su} \quad (3.37)$$

Let us consider first the case of a single stage path. According to the analysis done in Sec. 3.3.4, the delay Δ_F of the differential data at the output of the gate is a random

variable with a Gaussian distribution, with mean μ equal to δ_F and standard deviation σ in the order of 1% of δ_F , as seen by Montecarlo simulations executed in Sec. 3.3.4. If we consider a range of variation comprised between -3σ and $+3\sigma$, the probability that the variable Δ_F describing the output delay of the mismatched circuits falls within this interval is equal to about 0.99. The minimum value is just equal to $\delta_F - 3\sigma$. If another gate is cascaded, the output delay Δ'_F is again a Gaussian variable, with mean $\delta'_F = \delta_F - \Delta\delta$ and variance given by the sum of the variances of the gates, under the assumption that they are independent. Thus, the standard deviation σ' is given by $\sigma' = \sqrt{2} \cdot \sigma$. If we consider a range of variation comprised between $-3\sigma' = -3\sqrt{2}\sigma$ and $+3\sigma' = +3\sqrt{2}\sigma$, the probability that the variable Δ'_F describing the output delay of the mismatched circuit falls within this interval is equal to about 0.99. The minimum value is just equal to $\delta'_F - 3\sigma = \delta_F - \Delta\delta_{max} - \sqrt{2} \cdot \sigma$. In general, for a number of N cascaded gates, the output delay Δ_{CL} is a Gaussian variable with mean $\delta_{CL} = \delta - N \cdot \Delta\delta$ and standard deviation $\sigma_{CL} = \sqrt{N}\sigma$. If we consider a range of variation comprised between $-3\sigma_{CL} = -3\sqrt{N}\sigma$ and $+3\sigma_{CL} = +3\sqrt{N}\sigma$, the probability that the variable Δ_{CL} describing the output delay of the mismatched circuit falls within this interval is equal to about 0.99. The minimum value is just equal to $\delta_{CL} - 3\sigma_{CL} = \delta - N \cdot \Delta\delta_{max} - 3\sqrt{N} \cdot \sigma$.

The timing constraint is satisfied if the minimum value of δ_{CL} at the output of the combinational path is greater than the setup time of the flip-flop:

$$\delta - 3\sqrt{N}\sigma - N \cdot \Delta\delta_{max} > t_{su} \quad (3.38)$$

which becomes a quadratic inequality in the variable N :

$$N^2 \cdot \Delta\delta^2 - N \cdot [2\Delta\delta_{max} \cdot (\delta - t_{su}) + 9\sigma^2] + (\delta - t_{su})^2 > 0 \quad (3.39)$$

Assuming $\delta > t_{su}$, the associated quadratic equation has two real values:

$$N_{1,2} = \frac{\delta - t_{su}}{\Delta\delta_{max}} + \frac{3\sigma}{2\Delta\delta_{max}^2} \cdot [3\sigma \pm \sqrt{9\sigma^2 + 4\Delta\delta_{max} \cdot (\delta - t_{su})}] \quad (3.40)$$

Only the negative solution can be accepted, thus posing:

$$\bar{N}_{max} = \frac{\delta - t_{su}}{\Delta\delta_{max}} \quad (3.41)$$

which represents the deterministic value, we have:

$$N < N_{MAX} = \bar{N}_{max} + \frac{3\sigma}{2\Delta\delta_{max}^2} \cdot [3\sigma - \sqrt{9\sigma^2 + 4\Delta\delta_{max} \cdot (\delta - t_{su})}] \quad (3.42)$$

We point out that the parameters $\Delta\delta_{max}$, t_{su} , and σ are technology dependent. Equation 3.42 provides the maximum number of stages with a fixed value of δ in a certain technology when the mismatch variability of the logic gates is taken into account. It is given by the sum of two terms: the deterministic value $\bar{N}_{max} = \frac{\delta - t_{su}}{\Delta\delta_{max}}$ and a σ -dependent expression. If $\sigma = 0$, the maximum number of stages is just equal to \bar{N}_{max} . As it will be shown in next chapter, given that σ is a limited fraction of δ ($\sim 1\%$), the second term is usually small and has a small impact on the maximum number of cascaded gates.

3.5.3 Layout of the iDDPL cells

General issues regarding the layout of DPL cells

A final important remark concerns the layout of submicron cryptographic circuits. TELs have been conceived with the purpose of overcoming the data-dependence of the circuit due to the electrical mismatches on the differential wires of submicron cryptographic circuits, under the perspective of being integrated in a semi-custom design flow without any additional constraint on the routing of the standard-cells.

According to the discussion done in Chapter 1, there are three main strategies to overcome this problem: at gate level, masking allows to randomize the power consumption of the circuit in order to de-correlate the instantaneous power consumption from the processed data; at transistor level, the adoption of a novel data encoding in a specific circuit template can help to mitigate the electrical mismatches without additional constraints on the back-end flow; at layout level, the parasitic elements of the differential wires can be balanced with a manual optimization. Advantages and drawbacks of these strategies have been already discussed in Chapter 1.

The adoption of techniques to optimize the routing of the DPL standard cells is in general suboptimal in the design of submicron circuits. Even if these procedures can be automatized by inserting some ad-hoc scripts in the routing flow, it may be really difficult to obtain a very precise balance. Moreover the technology scaling stresses this issue, and the level of balance can result poor in spite of big efforts in terms of time and area overhead. Therefore, these techniques alone are insufficient and cannot be considered as an ultimate solution.

Moreover, a good balanced design starts from the layout of the logic cells: in order to guarantee a good balance of the interconnect wires, the internal nodes of the cells must be also balanced in order to equalize capacitances and propagation times on the differential paths. In other words, cell layouts must be first designed in order to guarantee a preferred routing direction in the 2D plan for the interconnection wires, as preliminary step before equalizing the interconnection wires.

An efficient solution to execute the layout of DPL cells is based on the possibility of locally selecting a low-leakage dual-line with a low sensitiveness on any possible capacitance mismatch between the differential wires, which allows to relax the aforementioned constraints. Indeed, the selected low-leakage dual-rail lines are tolerant on mismatches irrespective of the total amount of parasitics and are completely prone to be automatically routed as wire interconnections. As an example, in Fig. 3.41 two different capacitances are associated to the interconnect wires W and \bar{W} of two symmetric standard cells; thus in order to not reduce the security level of the design, these differential wires are critical and it must be ensured that they are low-leakage.

The identification of low-leakage differential dual-lines offers the possibility to avoid a preferred routing direction for the interconnection of the standard-cells in the 2D plan. The first consequence is that there is no constraint on the layout of the logic cells, which can be designed with a symmetric differential layout exploiting the differential circuit topology of the cells. Having a symmetric differential layout is a great advantage for the design, because allows to have a compact design, reducing the area overhead and avoiding capacitive and timing mismatch on local routing.

This property offers the possibility of freely placing the cells in a compact

grid-based layout and automatically interconnecting them without requiring any constraint on the routing, relying on the fact that the resulting mismatch is unloaded onto the output low-leakage interconnect dual-lines, as depicted in Fig. 3.41. Obviously this solution depends on the specific DPL style and the power leakage model, given that in many cases there is no possibility of selecting a low-leakage dual line, simply because it doesn't exist, as in the case of RTZ logics.

On the contrary, the most important advantage of TEL circuits is that all the differential wires are low-leakage, provided that the circuit implementation is designed in order to satisfy the timing constraints. In other words the balancing property already discussed at schematic level can be mapped at the physical level where the mismatches arise. A first important consequence is that, under the perspective of a side-by-side placement, the layout of a iDDPL gate can be executed following exactly the differential circuit topology of the cell, considering that the interconnect wires of the iDDPL gates are low-leakage and do not have a preferred routing direction, in accordance to the situation of Fig. 3.41.

Following these considerations, in Fig. 3.42 the layout of the basic combinational iDDPL gates, AND/NAND, OR/NOR, and XOR/NXOR, are shown. The layout of the cells has been done according to a rail-to-rail placement in a like-comb design, and are symmetric, compact and very well balanced in order to avoid any unwanted additional mismatch on the internal nodes, as assumed in the models described in this chapter. All the mismatch resulting from the routing are unloaded on the output interconnect wires of the cell, without any additional constraint, and therefore there is no need of manually controlling the balance of the differential interconnect wires, as expected in a standard automatic routing. Three levels of layout have been adopted to design the interconnection wires; clock signal is routed in parallel to the ground rail in order to minimize the parasitic capacitances from other layers; moreover, the static voltage V_{bias} is routed in parallel to the V_{DD} in order to exploit empty areas of the layout.

In Fig. 3.43 a screenshot of the layout of the SERPENT chip is shown after placing the cells; in figure the routing of two differential dual-wires is highlighted. These wires have different lengths of the metal layers and different amount of parasitics ($C + CC$), without impacting the level of security of the circuit. Furthermore, differential interconnect resistances are almost identical because the wires have similar length, thus the resistive mismatch can be neglected.

In conclusion, the circuit template of the iDDPL style allows to design fully differential logic-cells with a differential and balanced layout to support the TEL data encoding; more complex gates can be designed using the same strategy, given that the iDDPL standard-cells offers the possibility of being integrated in a semi-custom design flow where the routing of the cells can be automatically executed by the processor with no additional constraints.

3.5.4 Consideration on the layout of the flip-flop

The layout of secure flip-flops in DPL styles revealed to be a very difficult task for hardware designers. Several DPL styles (e.g. TDPL and SABL) are characterized by two cross-coupled inverters which create an internal feedback loop (i.e. the sense amplifier compound) required for sensing when the evaluated signal discharges the

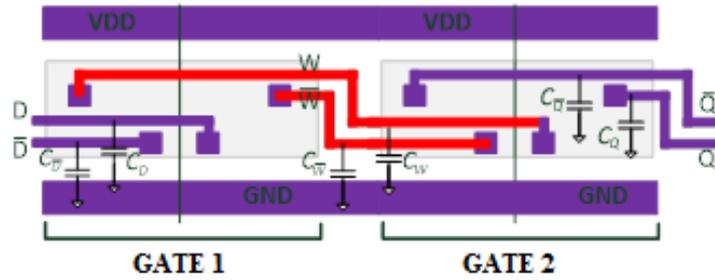


Figure 3.41. Modelization of the routing of two standard cells: the differential wires in red must be low-leakage.

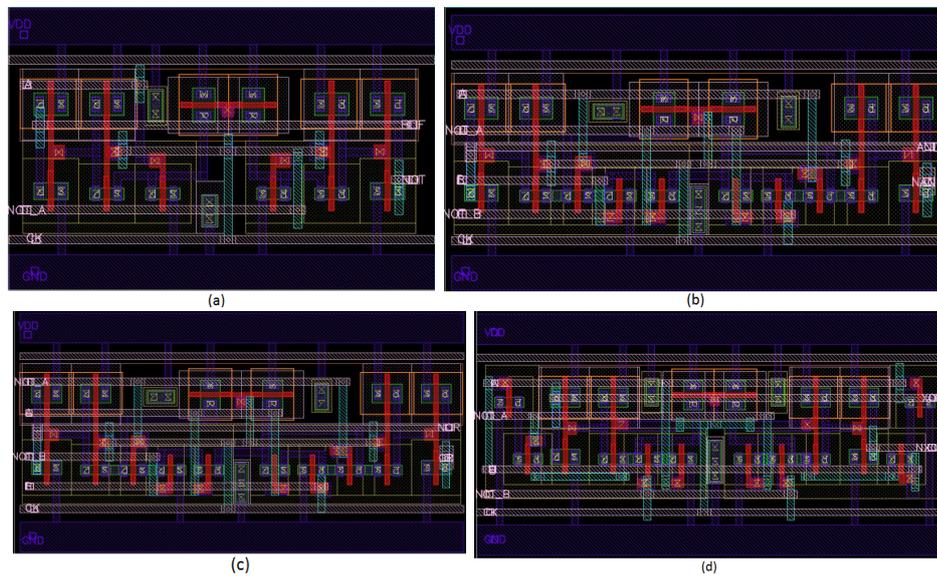


Figure 3.42. Layout of some combinational iDDPL cells: BUFF/INV (a), AND/NAND (b), OR/NOR (c), and XOR/NXOR (d).



Figure 3.43. Screenshot of the SERPENT-block after placement, with highlighted the differential interconnect wires between two iDDPL cells.

output capacitance. These feedback wires are high-leakage, therefore the layout of sense-amplifier based circuits requires to place the cells side-by-side in order to avoid long feedback interconnect wires and reduce the effect of cross-talk on them. In other words, the presence of high-leakage wires in these flip-flops determines the presence of a preferred routing direction, preventing the possibility of executing a symmetric layout.

In order to avoid the data-dependent power consumption resulting from the unbalanced interconnections, a straightforward solution is masking the signal before it is processed by the DPL flip-flop. This principle inspired the design of flip-flops for those DPL styles which make use of an unprotected standard CMOS SR-latch and require, like MDPL and TDPL styles, to combine a datum with a random mask in order to de-correlate the data-dependence from the instantaneous power consumption. However, in submicron technologies it has been demonstrated that combining hiding and masking is not a definitive solution if an adequate balance of the power consumption in presence of mismatches is guaranteed [109]; namely, the best solution is break the link between power and data already at physical level especially in those circuit points which are more critical for the security, as the sequential elements, which enhance the latency of a datum in the pipeline extending the relevant interval. Therefore a well balanced layout is essential in the design of combinational as well as sequential circuits, and, more important, the adoption of CMOS SR-sequential elements as latches should be in general avoided.

The iDDPL flip-flop described in the previous section is a fully differential circuit, designed in order to exploit the TEL data encoding and minimize the extension of the relevant samples during the hold phase. It is characterized by the fact that all the internal cascaded logic blocks (i.e. input converter, master-latch, slave latch) have a circuit topology composed of two independent half circuits that can be designed in a single differential architecture. Indeed, the flip-flop contains high-leakage dual-line pairs which unavoidably result from the conversion from the TEL to the RTZ domain. These potentially vulnerable differential wires are located just between the output of the input converter and the input of the slave latch. However, according to the strategy discussed in previous paragraph, it is still possible to identify low leakage dual-line pairs to connect the cells: these lines are just the input and the output differential wires of the flip-flop; thus, during the layout of the circuit it must be ensured that the high-leakage differential wires are fully equalized to reduce the parasitics and the cross-talk effects and balance the propagation times of the differential RTZ signals.

A differential layout is just in accordance to this: as shown in Fig. 3.44, the design of the flip-flop results to be very compact and perfectly symmetric, and ensures that the parasitic elements as well as the propagation times of the signals on the high-leakage internal differential pairs are well balanced. The half circuits of the blocks have been placed side by side, starting from the input converter (internal) up to the slave latch (external). This way the output complementary lines are the only ones to be mismatched after the routing, and each mismatch can be unloaded onto the output interconnect wires.

A post layout analysis confirms a perfect balance ($MF \approx 1$) of the capacitances on the internal differential wires. The design occupies an active area of about $61\mu m^2$, that can be also reduced by further optimization.

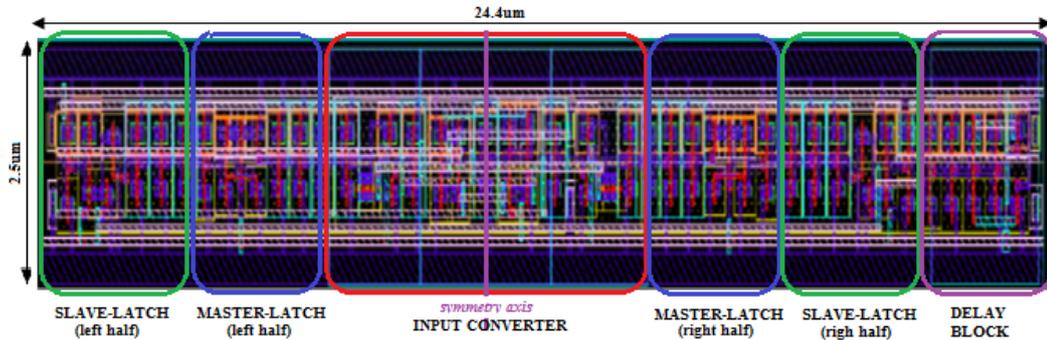


Figure 3.44. Symmetric layout of the iDDPL master-slave flip-flop.

3.5.5 The DDPL065 cell library

Finally, in this section we present the *DDPL065* prototype library (Table 3.9) for the design of lightweight cryptographic primitives, which we have used to design the SERPENT-block evaluated in previous section. The library is composed of iDDPL gates with minimum fanout, designed according to the analysis done in previous paragraphs.

The parameters reported in table are obtained through post-layout SPICE simulations. According to these values, the area overhead is calculated dividing the number of transistors used for the iDDPL gate by the number of transistors used for the correspondent CMOS gate in the tech library. According to the value in table, the iDDPL circuit template impacts the area overhead as well as the power consumption of a factor less than 4 with respect to the CMOS counterpart. The NED is calculated in the case of maximum unbalance ($MF = 4$), which is a considerably high value for small compact designs. For combinational gates, the variation of delta $\Delta\delta$ is calculated on the critical path associated to a specific data combination.

As a final remark, we point out that any residual skew of a differential signal pair along a combinational path, which results after the place and route of the design, can lead to a residual fluctuation which must be adequately controlled by the designer through post-layout simulations. For example, if there are long interconnections in the layout, the propagation times of the differential signals can differ, and the actual delay δ_F can be higher with respect to the value predicted by pre-layout simulations.

In order to assess this residual variation of δ , the distribution of the delay δ_F of the nets for every data combinations must be calculated and, possibly, the netlist must be re-designed according to a back annotation procedure. Alternatively, a designer can mitigate this effect by reducing the nominal value δ . This way, the residual fluctuation of δ due to the routing impacts only the time margin of the critical path, without reducing the level of security of the circuit.

However, for lightweight cryptographic circuits based on small and compact designs, long interconnections are avoided, therefore the mismatch due to the static skews of the differential signal propagating along these path is usually very low. Moreover, the variations due to the thermal gradient as well as the process mismatches

Table 3.9. The DDPL065 cell library, designed using the CMOS065 technology.

DDPL065	# Transistors	Area[μm^2]	max[E _{AV}] [fJ]	max[NED][%]	CK-Q[ps]	max[T1] [ps]	max[T2] [ps]
BUFF/INV	6 nMOS + 6 pMOS	x6.0	9.76	0.20	-	70	70
AND/NAND	10 nMOS + 6 pMOS	x4.0	14.04	1.56	-	115	95
OR/NOR	10 nMOS + 6 pMOS	x4.0	14.04	1.56	-	115	95
XOR/NXOR	12 nMOS + 6 pMOS	x1.5	14.71	1.31	-	112	80
MUX	30 nMOS + 18 pMOS	x4.0	35.07	1.21	-	230	185
FULL ADDER	54 nMOS + 30 pMOS	x2.3	68.67	0.97	-	350	290
CONV CMOS-TEL	11 nMOS + 8 pMOS	-	17.47	10.00	79.2	-	-
CONV TEL-RTZ	16 nMOS + 6 pMOS	-	7.31	29.20	195.2	-	-
CONV RTZ-TEL	12 nMOS + 9 pMOS	-	10.63	0.02	133.2	-	-
CONV TEL-CMOS	19 nMOS + 9 pMOS	-	16.16	39.78	307.7	-	-
FLIP FLOP	36 nMOS + 25 pMOS	x4.3	45.54	1.55	103.2	-	-

between different part of the circuits aren't expected to impact the level of security of the circuit, given the small dimension of modern cryptographic critical cores. Consider that the active area of the iDDPL SERPENT chip is about $2300\mu\text{m}^2$, and, as it will be shown in next section, no residual fluctuation has been detected.

3.6 Conclusions

The purpose of this chapter has been the complete characterization of a prototype standard cell library of iDDPL gates in terms of both functionality and security. The iDDPL style is a full custom logic family which has been used as circuit template for designing TEL cryptographic circuits. Some combinational as well as sequential circuits have been designed and characterized in terms of area, energy, and timing. Moreover, the properties of energy balancing and timing enclosing have been also assessed.

The cell library DDPL065 has been completely characterized with minimum fanout logic using CMOS065 device models. The values are shown in Table 3.9. An extended version of this library, DDPL065_EXT, can be designed using the design constraints defined in this chapter, in order to build higher fanin and fanout gates, as well as more complex combinational functions. The DDPL065 digital library can be adopted for a secure standard automatic place and route similarly to the flow defined in Fig. 1.17 shown Chapter 1, with the fundamental difference that the back-end flow can be performed without any requirement on the routing of the interconnections of the logic cells. The secure semi-custom flow for the iDDPL library can be used to design a TEL circuit with the security performances analyzed in Chapter 2. The digital flow is shown in Fig. 3.45, where unlike the standard flow of DPLs reported in Fig. 1.17, the *optimize.tcl* script has been removed.

The DDPL065 library has been used to design the SERPENT-block described in previous chapter. In the next chapter, the design steps of the SERPENT-block are described in more detail, as well as the synthesis procedure to design the ASIC where the SERPENT-block is placed as macro-block.

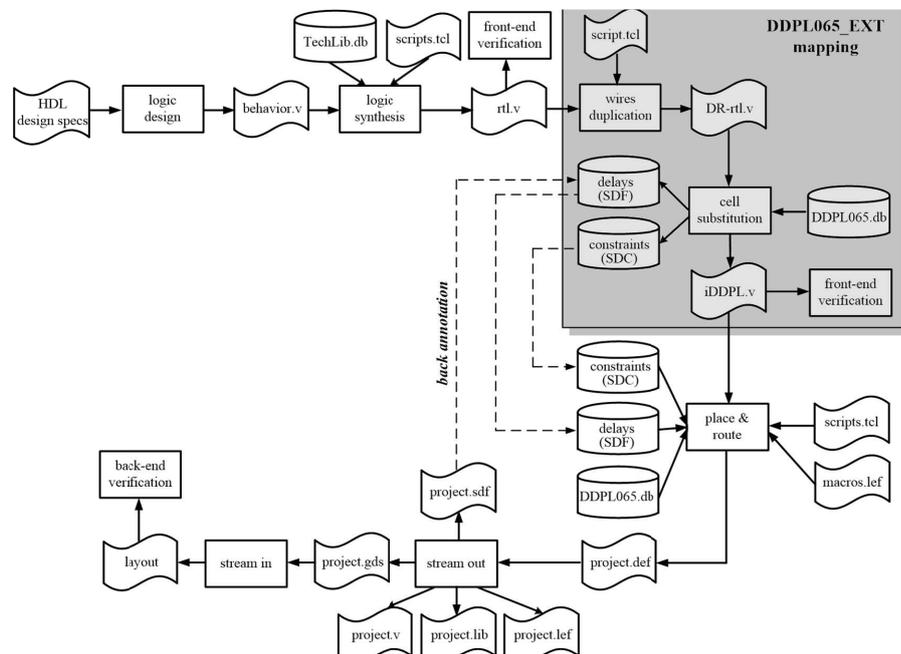


Figure 3.45. A detailed description of a secure semi-custom design flow to implement a TEL circuit using iDDPL gates.

Chapter 4

Design of the SERPAES prototype chip

4.1 Introduction

This chapter is dedicated to the description of the design flow of the cryptographic chip SERPAES, completely developed in our department in the period between the second half of 2013 and the beginning of 2014. SERPAES is a prototype ASIC, designed in CMOS065SVTLP technology. In the chip some cryptographic cores are integrated, which implement fully or partially the AES (or Rijndael) and SERPENT encryption processor units. As already discussed in previous chapters, the design of cryptographic circuits in submicron technologies is characterized by some critical issues that may reduce or even invalidate the effectiveness of a countermeasure which has been defined as secure in simulation (both at RTL or SPICE level). Furthermore new leakage sources can emerge after chip manufacturing. Thus, the purpose of this project is to validate a set of countermeasures against PAAs, developed in our department in last years, through an *in silico* verification.

With reference to Fig. 4.1, the chip is composed of two macro-blocks which have been designed and simulated with different tools: a full-custom cryptographic circuit and a VHDL-described cryptographic block. More specifically, the design of the full-custom cryptographic unit, which has been named *SERPENT-block*, represents one important contribution of this thesis work. It has been designed using two selected DPL styles in Cadence Virtuoso environment, and tested through SPICE-level post-layout simulations. The SERPENT-block is composed of two sub-blocks, one implemented in iDDPL using the DDPL065 cell-library described in previous section and characterized at layout level, and the other one implemented in SABL style. The PAAs resistance of the two sub-blocks has been already tested in Chapter 2 and compared.

The second block is composed of various cryptographic AES cores, which implement RTL countermeasures to protect the AES encoder from PAAs; the specifications of the cores are provided by a set of VHDL files, each one describing a single component. The codes have been developed in our department in the last years [23] [77] [78] [79]. We will refer to this cryptographic macro-block as *AES-block*. The RTL countermeasures implemented on the AES-block have been evaluated through

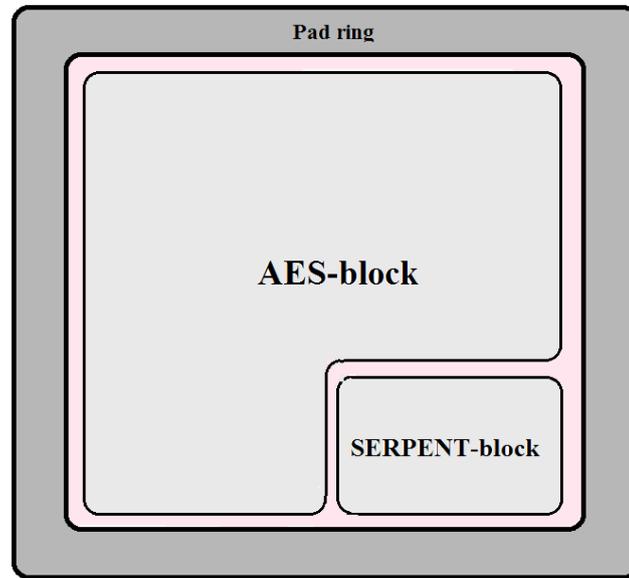


Figure 4.1. Macro-blocks composing the SERPAES chip: the AES-block and the SERPENT-block

several experiments executed on an Altera Cyclone FPGA [78] [79].

4.2 Design of the SERPENT-block

4.2.1 Full custom design methodology

In this section the design steps of the SERPENT-block will be described, starting from the high-level specification of the data-path until the layout of the circuit. The block is composed of two sub-blocks implementing the same data-path: one is built in the iDDPL style, the other one is a SABL implementation. The level of security of the circuits against PAAs has been evaluated and compared in Chapter 2, using the post-layout current traces exported from Cadence. In Chapter 3 we have provided an accurate description of the design of the iDDPL standard-cells, which have been used to build the SERPENT-chip. In the same way we have built a small SABL standard-cell library. The SERPENT-block has been designed in Cadence environment: simulations have been executed in Virtuoso, and the layout has been done using DRC and LVS verification tools of Calibre. The technology is the CMOS065SVTLP process described in the introduction of this work.

The iDDPL style has been presented as a full-custom circuit template for a TEL-featured chip, whose purpose is to overcome the electrical mismatches arising in submicron technologies. The DDPL065 library presented in the last part of Chapter 3 has been designed under the perspective of being integrated in a semi-custom design flow, according to the diagram shown in Fig. 1.17. However the SERPENT-block has been designed using a full-custom approach for two main reasons: first, the chip is conceived as a prototype, therefore any possible issue encountered during the design flow can be highlighted and solved in a more simple way with respect

to a semi-custom design flow; moreover, customizing the layout of the two circuits offers the possibility of controlling each part of the design in order to have a fair comparison between them, which would be more difficult using the standard design flow.

For all these reasons, the design flow we have followed is slightly different from that shown in Fig. 1.17 in the fact that, after having produced a differential netlist of logic gates taken from the DDPL065 library, the layout has been manually executed by placing the logic-cells side by side in the same way for the two circuits and by executing the routing with a regular automatic procedure in Virtuoso Layout Editor environment. Then, after having been tested through post-layout simulations to verify functionality, timing requirements, delays and corners, it has been integrated in the back-end flow as a macro-block (file *macros.lef* in Fig. 1.17).

We point out that the full-custom approach is compatible with the need of manufacturing a prototype chip in iDDPL following a step-by-step design procedure, in order to control each phase and have a fair comparison with SABL; however the gates of the DDPL065 library can be adopted for a standard semi-custom flow thanks to their balanced power consumption which do not require additional constraints on the place and route, as proved by PAAs results presented in Chapter 2.

4.2.2 Data-path of the circuit

The data-path of the circuit has been already described in Sec. 2.6.2, depicted in Fig. 2.18, and reported in Fig. 4.2 for simplicity. It is a 4-bit architecture which implements a bit-slice unit of the first stage of the Serpent encrypting processor shown in Fig. 2.17. As discussed in Sec. 2.6, Serpent can be efficiently implemented with a bit-slice structure where the S-Boxes can be implemented as combinational blocks in order to have a compact area occupation and exploit parallelism.

In the SERPENT-block, the S-Box S_0 has been chosen as a case study; the logic table of the S-Box S_0 is reported in Fig. 1.11, whereas in Appendix A the Boolean equations implementing the logic table are reported; in Table A.1 two different implementations are presented: the first one is not optimized (full) being composed of 57 instructions (17 AND, 18 NAND, 1 XOR, 21 NOT); the second implementation (lightweight) is described by a set of 16 instructions (3 AND, 1 OR, 10 XOR, 2 NOT).

The logic synthesis of the S-Box and the design of the combinational path put our logic style in a very pessimistic case. Indeed, we have chosen the full implementation of S_0 because using a non-optimized implementation allows to give a better estimation on the functionality of the iDDPL cells in terms of timing specification and area overhead in a worst case situation. The critical path of this implementation is equal to 7 logic gates (Fig. 2.18), higher than the critical path of the lightweight implementation (5 gates). If also the XOR gate before the S-Box is considered, the critical path is equal to 8 logic gates, which represents a conspicuous number. We remark that the minimum allowable delay δ depends on the number of cascaded gates, in accordance to Eq. 2.13. Furthermore the non-optimized implementation requires almost a quadrupled area with respect to the lightweight version, increasing the amount of expected parasitic capacitance. Thus, in practical implementations and using more modern technology nodes, performances of the iDDPL style can be

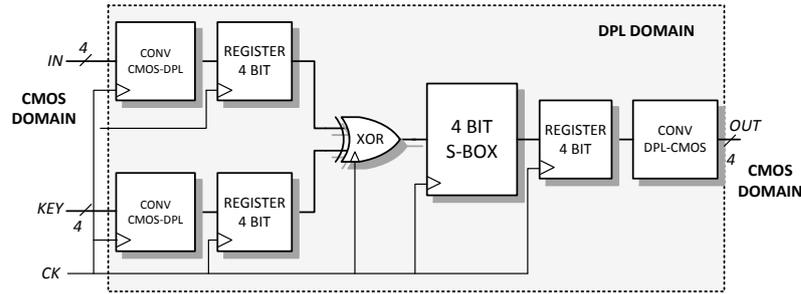


Figure 4.2. Data path of a DPL-feathered 4-bit unit implementing the first round of *Serpent* processor.

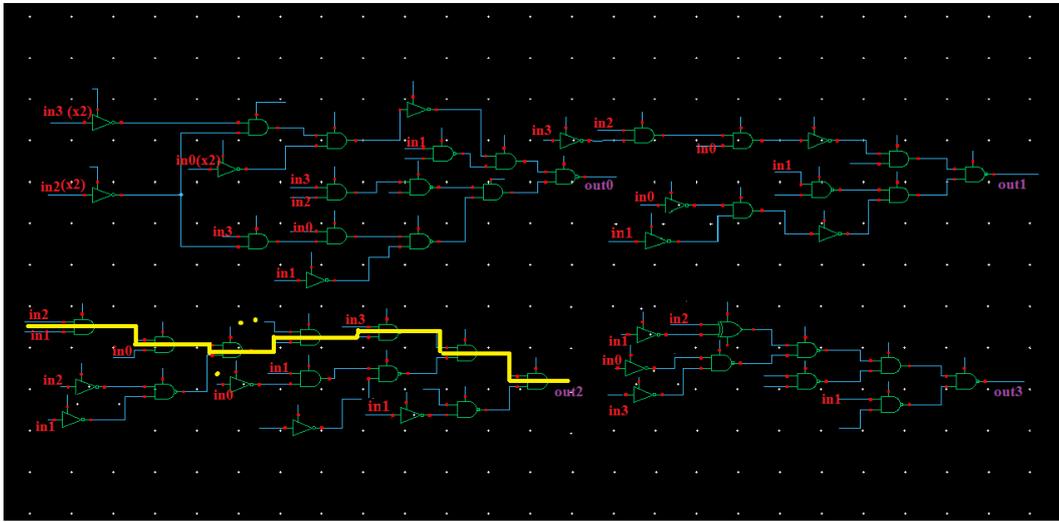


Figure 4.3. Schematic view of the S-Box S_0 from Serpent algorithm implemented only with combinational CMOS gates; the critical path is also indicated in figure.

dramatically improved.

The CMOS implementation of the netlist using the tech library CMOS065SVTLP is composed by 54 logic gates, using minimum fanout cells (Fig. 4.3). The *Add-KeyRound*, not shown in figure, is composed by 4 parallel XOR gates with a higher fanout, given that signals $IN[3:0]$ are routed into more than one gate: IN_0 is routed to 8 logic gates, IN_1 to 10 logic gates, IN_2 to 7 logic gates, and IN_3 to 7 logic gates.

4.2.3 Design of the iDDPL sub-block

Duplication of the signal nets

The iDDPL implementation of the circuit can be built by executing the substitution of the CMOS logic gates in schematic of the chip design with the logic gates from the DDPL065 library. The iDDPL style is a differential logic family, then the first step is the duplication of all the nets and the removal of the CMOS inverters of the entire core. Indeed it must be noted that in DPL circuits the presence of CMOS inverters is redundant, because in order to generate the non-asserted signal

of a differential pair, the differential wires can be simply swapped [72].

Insertion of the iDDPL gates

Then, the CMOS gates of the schematic have been substituted with the iDDPL gates from the library DDPL065. Provided that iDDPL is a dynamic differential logic style, the clock signal must be routed into all the combinational gates of the schematic. The design of the clock tree deserves an in depth description which will be provided in the following paragraph.

At this design step we have also placed the CMOS-TEL converters and the TEL-CMOS converters at the IO interface of the data-path. Furthermore the iDDPL flip-flops as well as the input converters require the routing of the static voltage V_{bias} for the generation of the TEL data encoding. In accordance to the delay element Δ presented in Chapter 3 (Fig. 3.7), this voltage has been globally generated by a polarization circuit, which will be also described in the following, according to the data-path of Fig. 4.2.

Design of the clock tree

The design of the clock tree is one of the most important phases of the design flow, in particular for dynamic logic circuits where the clock is provided to both combinational and sequential cells and has usually a higher fanout with respect to the CMOS counterpart. As described in next paragraphs, during the semi-custom back-end design flow the assisted place and route procedure automatically inserts and optimizes the clock tree according to the timing specifications elaborated during the synthesis through several iteration steps. In a full-custom design, the clock tree must be manually designed with a very good accuracy, therefore bigger is the circuit, more complex is the design of the clock tree. In synchronous circuits, clock signals must be distributed to all the devices ideally at the same time instant in order to avoid skews. The clock skew is a deterministic effect for which the clock signal distributed by the clock tree arrives at different components at different times without changing its duty cycle (i.e. constant frequency), due for example to different propagation times of the clock paths, too much high wire-interconnect length, or differences of fanout (i.e. different input capacitances of the clocked devices).

There are two well-known strategies widely adopted to implement a clock tree in an ASIC: the H-topology and the buffers tree. The buffers-based tree is probably the most used methodology for clock distribution; it is based on the insertion of cascaded buffers with an increasing drive strength and fanout; each stage drives the input capacitance of the following stage balancing the propagation times of the signals and reducing the skew from a stage to the next.

Given the low number of gates (the circuit is composed of 62 clocked elements: 12 flip-flops and about 50 combinational gates) and the low complexity of the circuit, we have chosen the buffers-based clock tree. The first step has been setting the correct number of stages of the clock tree as well as the sizes of the buffers. The total fanout of the clock tree can be determined in the following way. According to the circuit topology of the combinational iDDPL cells, each gate has a total input capacitance equal to 3 gate capacitances, whereas a flip-flop has a total capacitance

of 6 gate capacitances. Therefore, the total fanout of the clock tree is equal to 222 gate capacitances.

After having determined the fanout of the clock tree, we have divided the clock domain into 10 sub-domains. According to the total fanout of the tree, each output buffer should have a minimum fanout to drive an amount of gate capacitance equal to about 22. The first buffer in the tech library which meets this requirement is the CNBFX24.

The five stages of the clock tree have been implemented through standard CMOS inverters with an increasing fanout. The designed clock tree is shown in Fig. 4.4. In Fig. 4.5 we have calculated the propagation times of the waveforms at the output of the clock tree with respect to the input clock; furthermore, we have calculated the maximum deviation of the clock signals due to the lack of precision of the clock tree. As seen in figure, the total propagation time of the clock tree is around 205.7ps, whereas the maximum deviation is equal to 7.9ps (i.e. 3.8% of the propagation time), which is an acceptable value. The layout can have a strong impact on the propagation time as well as the clock skew, therefore these parameters will be re-calculated after post-layout simulations.

Electrical specification of the power domain

The SERPENT-block is conceived as a prototype chip to evaluate the PAAs resistance of the iDDPL style and possibly detect any leakage source of the circuit. The combinational and sequential logic cells are expected to give a different contribution to the overall leakage; furthermore, under the perspective of mounting DPA/CPA attacks as well as LPA attacks, we have decided to keep separated the power supply wires of the different section of the circuit. For this purpose, 6 different power domains have been defined for the iDDPL circuit:

- net V_{DD0} is connected to the clock tree, to the global polarization element and to the the output converters, which are not of interest for the leakage analysis and represent on-chip noise in the application;
- net V_{DD1} is connected to the input converters;
- net V_{DD2} is connected to the first stage of flip-flops;
- net V_{DD3} is connected to the XOR gates which implement the *AddRoundKey* layer of the algorithm;
- net V_{DD4} is connected to the combinational S-Box block of the circuit, which implements the *Substitution* layer;
- net V_{DD5} is connected to the second stage of flip-flops.

Each power net should be connected to a dedicated pad of the final chip, as it will be described in next paragraphs. Even if the power domains are kept separated, the same voltage must be applied. Furthermore, under the perspective of drawing different power rings in the layout of the circuit, some capacitors have been added on each power wires. Given that the SERPENT-block will be inserted as a macro-block in the final design of the SERPAES chip, these capacitors are fundamental

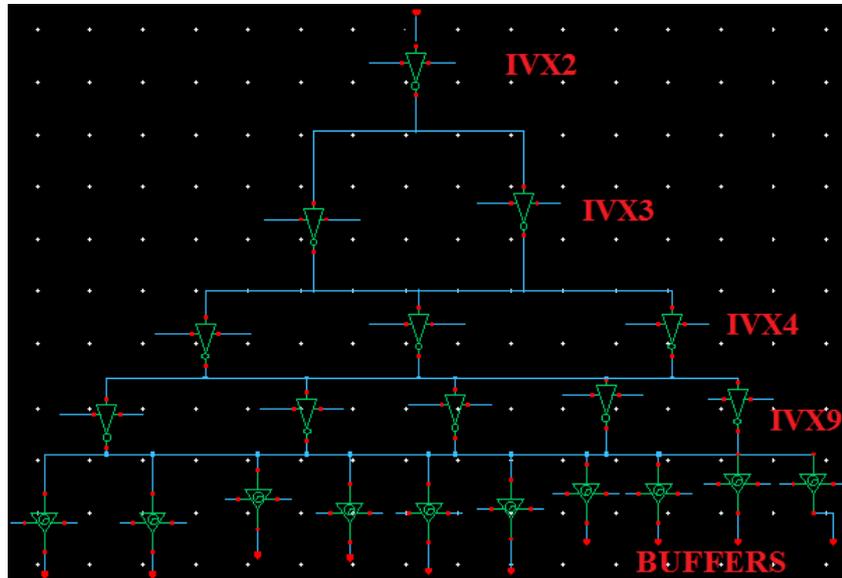


Figure 4.4. Clock tree of the iDDPL core.

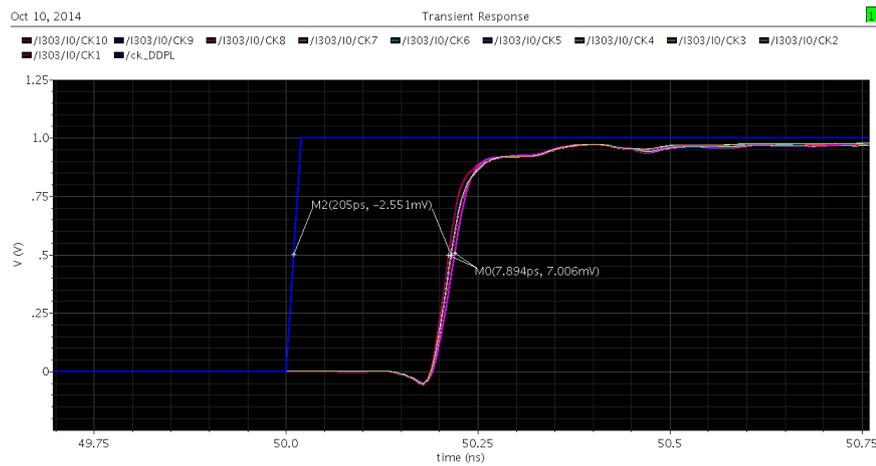


Figure 4.5. Deviation of the clock signals at the output of the buffers (pre-layout).

because filter high frequency noise on the power rails which could compromise the functionality of the circuit. Moreover, according to the analysis done in Chapter 2 for TEL circuits, this capacitance contributes to reduce the transient effect on the dynamic current consumption and reduce the data-dependence already at layout level. On each power net the amount of decoupling capacitance is equal to 8pF. Any additional capacitance can be added during the back-end flow of the chip, or externally on the chip board.

Choice of δ and design of the V_{bias} generator

At this step, we have defined the level of security for our chip by choosing δ in the range $[1ns \div 1.5ns]$, which corresponds to a minimum resolution required from the attack setup between 700MSamples/sec and 1GSamples/sec. We start from a value of δ equal to 1ns.

An important issue in the design of the iDDPL core is the design of the polarization circuit to generate the static voltage V_{bias} which must be routed into the sequential elements and the input converters. For this purpose there are many strategies that can be carried out. The most simple way is to use two series resistors which act like a potential divider of the supply voltage V_{DD} to generate the voltage V_{bias} . This is the solution adopted in the SERPENT-block, given the simplicity of the scheme (see Fig. 4.6). A potential divider allows also to have direct control of the internal polarization voltage from the external of the chip through a dedicated pin V_{bias} in the padding.

The resistors are the High-Resistance P+ Poly (*rhiporpo*) models found in the tech library; they are three ports components (one port is the body which must be set to ground) implemented in poly-silicon to guarantee high resistance. The segment resistance R_S , which is the resistance per area unit, is equal to $6k\Omega$; the resistance can be calculated by the formula:

$$R = R_S \cdot \frac{L}{W} \quad (4.1)$$

and the minimum recommended length L is equal to $5\mu m$.

The tuning range curve of the polarization circuit has been reported in Fig. 3.8 for the case $V_{DD} = 1V$. We decided to supply the chip with a voltage equal to 1.2V, which represents the nominal value in the adopted CMOS065 technology, thus the tuning range curve is re-calculated (Fig. 4.7). As indicated in figure, in order to obtain $\delta = 1ns$, V_{bias} must be set to 731.8mV, and the potential divider must have been sized in this way:

$$V_{bias} = V_{DD} \cdot \frac{R_1}{R_0 + R_1} = 731.8mV \rightarrow R_1 = 1.563 \cdot R_0 \quad (4.2)$$

Using the values available in the technology, by setting for example $R_1 = 18.530k\Omega$ ($W = 1.6\mu m$), we obtain $R_0 = 28.98k\Omega$. The nearest value in the technology is $29.00k\Omega$ ($W = 1.1\mu m$), from which $V_{bias} = 733mV$ and $\delta = 1.044ns$.

The potential divider has the drawback of producing a voltage which is dependent on the value of the supply voltage. Any noise on the voltage V_{DD} is reflected into an error on the nominal value V_{bias} . In digital circuits an important source of noise

is the clock signal, detectable on each internal signal of the circuit. In order to eliminate this noise component on the V_{bias} wire we have inserted two Poly N+ Nwell (*cpo25nw*) capacitors. Provided that the cryptographic circuits work at frequencies lower than 10MHz we have sized the capacitors in order to low pass at this frequency:

$$\frac{1}{2\pi \cdot R_0 // R_1 \cdot C} \leq 10MHz \quad (4.3)$$

from which we obtain $C \geq 1.7pF$. Using the values in the technology, two parallel capacitances equal to 3.3pF are inserted, for a total capacitance of 6.6pF. This way noise components at frequencies greater than 2.6MHz are filtered off; smart-cards work at about 4MHz, but if the application requires lower working frequencies it could be more convenient to add more capacitance in order to reduce clock noise. We point out that the V_{bias} is routed directly into the gate of a transistor of the starved inverter. In order to avoid antenna errors in the DRC, we have added a couple of diodes shunted to ground.

The proposed solution is very simple and offers the possibility to control the internal V_{bias} by connecting it directly on a pad of the chip. The external pad allows to change with a fine-grain resolution the static voltage from the nominal value of 733mV, in the case for example that a different power supply voltage is applied or a different δ is required. In literature there are more efficient solutions to perform the same thing in practical applications, but they do not offer the possibility to apply an external analogue voltage. For example, one methodology is to use a programmable polarization circuit adopting a switched-current mirror which can be externally digitally controllable. An example can be found in [60]. The current flowing into the starved inverter is drawn by n current mirrors, which are activated by switching transistors. By externally setting the n -bits it is possible to increase or reduce the amount of current for the starved inverter in order to change the output delay. This structure is flexible and robust, provides a good noise immunity and is not susceptible to the offset and drift phenomena and is suggested by us for improved applications.

Pre-layout simulations in the nominal case

The first set of simulations has been done on the final schematic of the iDDPL circuit in order to validate the functionality of the design in the nominal case and assess the corners of the circuit. In the testbench also a CMOS implementation of the same data-path is simulated in parallel to the iDDPL circuit. The circuit has been simulated using a supply voltage of 1.2V and a clock frequency of 10MHz, which are typical in cryptographic applications. The duty cycle of the clock is 50%, whereas the rise and fall times are set to 10ps.

Results of transient simulations for several random input data are plotted in Fig. 4.8, where the input, the clock, and the output waveforms of both CMOS and iDDPL are depicted. Simulations have been done in the nominal case, with a temperature equal to 27°C.

Waveforms *test_FF* and *test_CP* are test signals, obtained by XORing the iDDPL signals at the output of one flip-flop and at the output of the last stage of the critical path, respectively. The peaks in these waveforms allow to calculate



Figure 4.6. Polarization circuit to generate the static voltage V_{bias} in the iDDPL core.

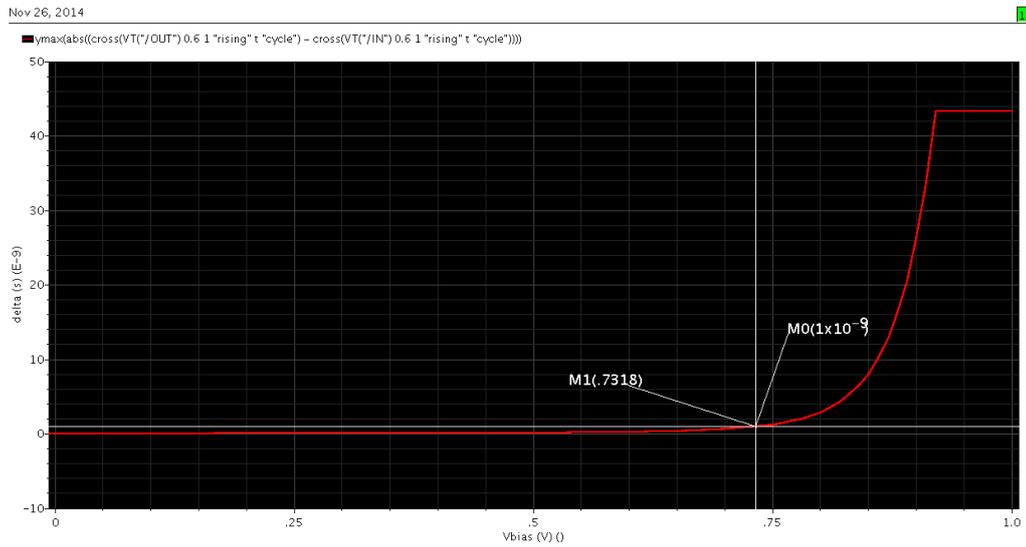


Figure 4.7. Dependence of the delay δ with the control voltage V_{bias} , for $V_{DD} = 1.2V$.

immediately δ and δ_{CR} for each clock cycle. The difference between these two values represents a first evaluation of the propagation time of the critical path, defined in Eq. 2.10 as the maximum variation of δ along the circuit:

$$\max[t_{CP}] = \delta - \min[\delta_{CP}] \quad (4.4)$$

This value is very important because allows to define the functionality mask of the circuit, once the critical path is designed and the level of security is fixed, and it will be calculated with more accuracy after post-layout simulations by taking into account also mismatch variations and interconnect delays. The functionality of the circuit has been tested with success also for different frequencies (i.e. from some MHz up to 200 MHz).

Analysis of the variability of the tuning range curve with process-voltage-temperature

The analysis of the corners of a digital circuit represents an important issue before chip manufacturing. In this regard, it must be pointed out that for the case of an iDDPL chip, the critical block is represented by the starved inverter which generates the bias voltage for all the delay elements distributed in the circuit. The tuning range curve of the delay element (i.e. δ vs V_{bias}) highlights a certain variability of the delay δ with V_{bias} , thus it must be investigated which is the impact of process-voltage-temperature (PVT) variations on the functionality of the circuit. Anyway, it must be also pointed out that in our implementation, we decided to externally control the static voltage V_{bias} , and this allows to change the value of δ also in presence of variations, thus we expected that they don't have a direct impact on the chip functionality because the operating point can be calibrated by a manual control. Nevertheless, the analysis performed in this paragraph is useful to understand the variability of the tuning range curve in order to design further improved implementation of the circuit.

With reference to the starved inverter in Fig. 3.7, the output δ of the device is approximately equal to the propagation time t_{pLH} , which represents the time that P2 and P3 take to charge the input capacitance of the output CMOS inverter:

$$\delta \approx t_{pLH} = R_p C_L \quad (4.5)$$

where R_p is the path resistance of the pull-up network represented by transistors P2 and P3 of the starved inverter, and C_L is the output capacitance. The resistance R_p is given by the sum of the voltage-controlled resistance P2 and the resistance of P3. The resistance of P2 has the purpose to increase the propagation time t_{pLH} with respect to t_{pHL} in order to change the duty cycle of the waveform and generate δ .

The voltage V_{bias} is chosen in order to polarize P2 in the weak inversion region, so to have a sufficiently high channel resistance and allow a large tuning range. In order to increase the path resistance of P2, it can be sized in order to have a very low aspect ratio, for example using $W = W_{min} = 0.135\mu m$ and a large L ; this way P2 can work in the strong inversion region where the relation between gate voltage and drain current is linear, and its path resistance is high enough to increase δ . The drawback of this solution is that high values of L lead to a big area overhead, which

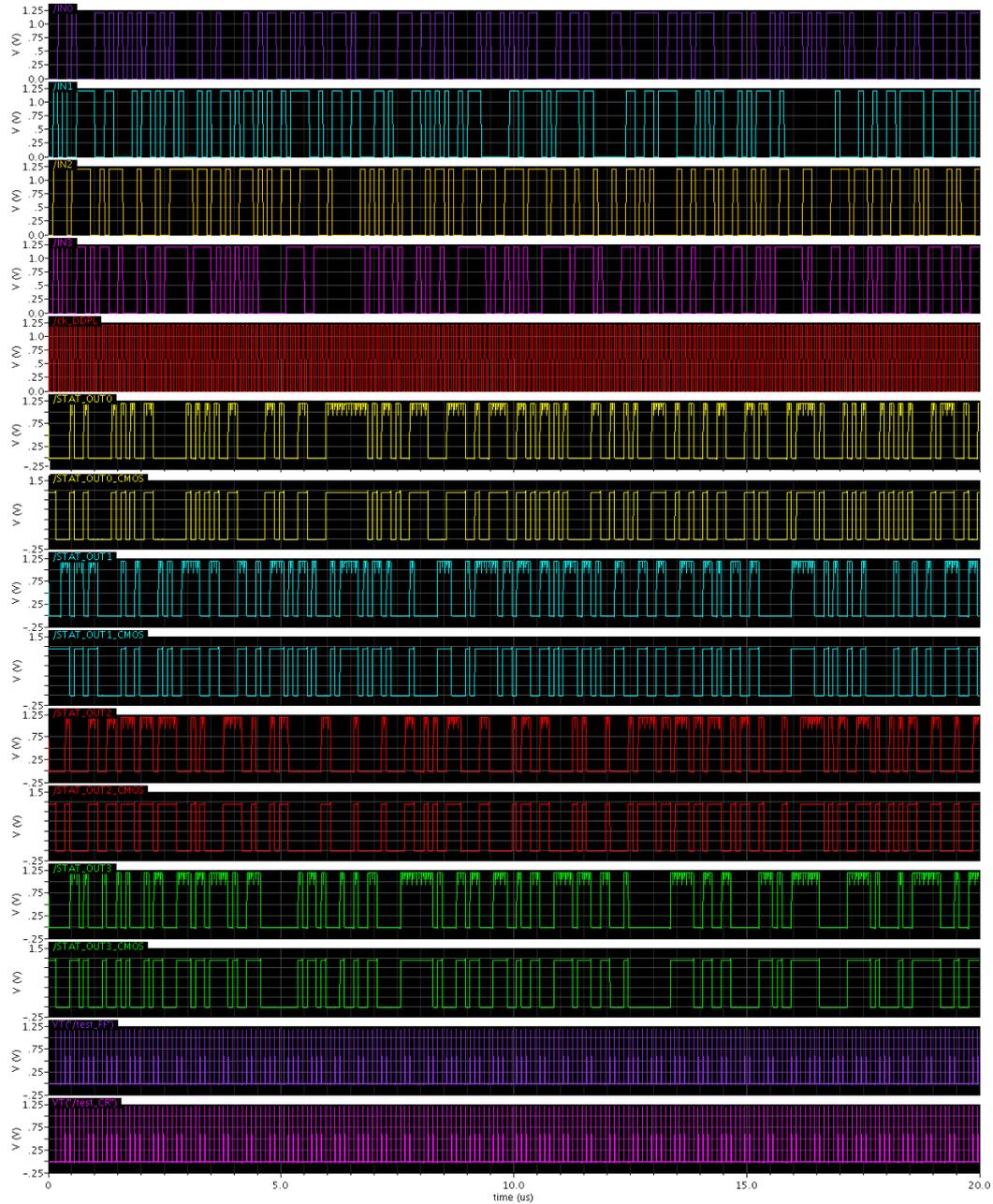


Figure 4.8. Waveforms of the input/output, clock and test signals of the iDDPL circuit before layout.

can be unacceptable in low area applications. Furthermore, it is very difficult to polarize P2 so that it works always in the linear region. For this reason, we decided to use the minimum value of L in this technology to save area and polarize P2 in the weak inversion region, where the channel resistance is very high and allows to have a better control on the calibration of δ . The drawback of this solution is that the relation between the gate voltage V_{bias} and the drain current is highly non linear, in accordance to the formula:

$$I_d = I_{d0} \cdot e^{\frac{V_{SG} - |V_t|}{n \cdot kT/q}} \cdot (1 - e^{-\frac{V_{DD} - V_S}{kT/q}}) \quad (4.6)$$

This is the reason for the device works in the non linear region of the tuning range curve, as already seen in Fig. 3.7. The path resistance of the pull up is given by the formula:

$$R_p = R_2^{weak.inv} + R_3^{strong.inv} = \frac{V_{DS2}}{I_{d2}} + \frac{V_{DS3}}{I_{d3}} \approx \frac{V_{DS2}}{I_{d0} \cdot e^{\frac{V_{DD} - V_{bias} - |V_{tp2}|}{n \cdot kT/q}}} + \frac{V_{DS3}}{\mu_p C_{ox} \frac{W}{L} (V_{SG3} - |V_{tp3}|)} \quad (4.7)$$

where the voltage V_{SG2} in the expression of I_{d2} is:

$$V_{SG2} = V_{DD} - V_{bias} = V_{DD} - \frac{R_1}{R_1 + R_2} \cdot V_{DD} = \frac{R_2}{R_1 + R_2} \cdot V_{DD} = k \cdot V_{DD} \quad (4.8)$$

According to Eq. 4.7, when V_{bias} is increased (up to $V_{DD} - V_{tp2}$), the term R_2 reaches its maximum value, and R_p and δ undergo the maximum increase. When $V_{bias} = V_{DD} - V_{tp2}$, $\delta = \delta_{MAX} = \frac{T_{CK}}{2}$. On the contrary, when V_{bias} is reduced, P_2 starts to work in strong inversion, and R_p and δ decrease. When $V_{bias} = V_{biasMIN}$, $R_p = R_{pMIN}$ and $\delta = \delta_{MIN}$. This is in accordance to the plot of the tuning range curve depicted in Fig. 4.7. It must be pointed out that if $V_{bias} < 0$, the pn junctions are directly biased, and increasing further the module of V_{bias} may generate higher direct currents and latchup effects which are dangerous for the devices.

Due to the exponential term in the expression of P2, this transistor has a higher impact on the corners of the circuit. The variability of the path resistance with the temperature is dominated by the effect of P2: in the weak inversion, P2 behaves like a bipolar transistor and the drain current increases with the temperature. On the contrary, in the strong inversion region the carrier mobility $\mu_p(T)$ has a quadratic inverse relation with the temperature [101]. The overall effect is that R_p decreases when the temperature is increased.

In regard to the variations of the power supply voltage, when V_{DD} is increased (reduced), R_p is reduced (increased) as well as δ , as it is clear from Eq. 4.7 and Eq. 4.8.

Furthermore, the resistance R_p is affected only by the variations of the pMOS devices: when the pMOS transistors are fast (slow), R_p is reduced (increased) and δ is also reduced (increased).

This analysis allows to determine the variability of δ such as:

- Fast pMOS, $T_{MAX}, V_{DDMAX} \rightarrow \delta = \delta_{MIN}$
- Slow pMOS, $T_{MIN}, V_{DDMIN} \rightarrow \delta = \delta_{MAX}$

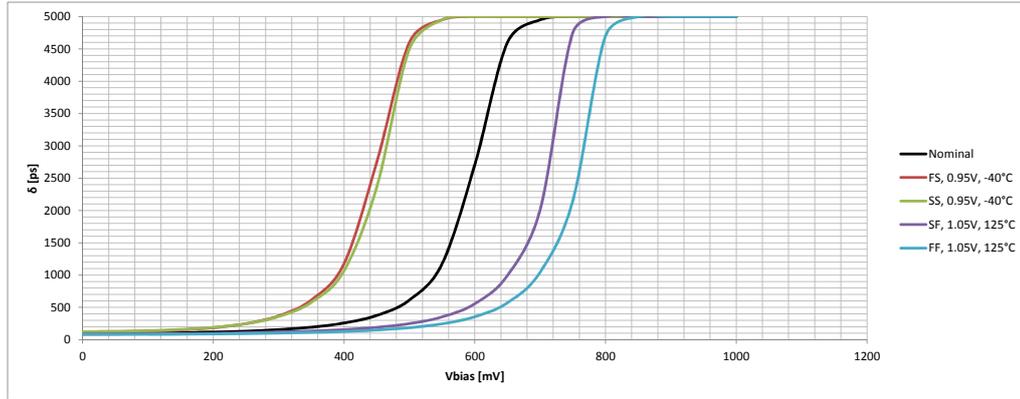


Figure 4.9. Variation of the tuning range curve for the different corners of the circuit.

In order to have a sound evaluation of the variation of the tuning range curve with temperature and voltage, we consider the range for worst and best case equal to $[-40^{\circ}\text{C } 125^{\circ}\text{C}]$ for the temperature and $[0.95\text{V } 1.05\text{V}]$ for the supply voltage. The tuning range curves for the different corners are then calculated using these values and performing parametric simulations. They are reported in Fig. 4.9 for the case $f_{CK} = 100\text{MHz}$. A similar plot can be obtained by changing the operating frequency, as discussed in previous chapter. If the polarization of the delay element changes because of PVT variations, the voltage V_{bias} can be also modified in accordance to the tuning range in order to reset the circuit to the desired value of δ .

We would like to point out that in practical applications several techniques exist in order to generate a fixed (constant) voltage V_{bias} irrespective of power supply variations, temperature changes and loading of the devices, as for example bandgap voltage references [31] or voltage controlled delay locked loops (DLL), but this is outside the objective of this work. The range of variability of δ constrains the range of variability of V_{bias} (Fig. 4.9) required from the control circuitry.

Layout of the circuit

In the SERPENT-block, the layout of the iDDPL core has been done by manually abutting the cells, like in a side-by-side placement strategy (see Section 4.5.1). The routing of the gates has been performed using the automatic router of Virtuoso Layout Editor. This way, it is possible to monitor the power consumption of the circuit in the general case of randomly mismatched interconnect capacitances, in accordance to a semicustom design procedure, as widely discussed in previous chapters.

The first important step is to trace the power nets VDD and GND by adequately selecting the layer of metallization and the wire width. According to pre-layout simulations, the average current on a clock cycle is in the order of few tenths of micro Ampere, more precisely around $40\mu\text{A}$, which is very low and does not represent a strict constraint for the width of the metal wires. This allows to draw metal wires with the minimum allowable width defined in the tech library. For this purpose, the GND and VDD nets have been drawn using the lowest metal layer (M1) with the

minimum width of $0.09\mu m$; using the rule in the tech library, the maximum I_{AV} with $W = W_{min} = 0.09\mu m$ is equal to $160\mu A$ at $100^\circ C$; considering that the I_{AV} of the circuit is around $30\mu A$ in normal operating conditions, all the internal metal wires M1 connected to VDD and GND have been drawn with the minimum width. Furthermore, the amplitude of the dynamic current peak I_{peak} is in the order of few milli Ampere, more precisely $4mA$, for a length of few hundreds of pico seconds; by using the design rules in the tech library, the maximum allowable current peak is equal to $830mA/\mu m$, which with $W = W_{min} = 0.09\mu m$ leads to a maximum current peak of $75mA$, largely greater than the current peak of the design.

The internal VDD and GND rows has been designed under the perspective of performing a comb-like layout; the wires have been drawn separated by $2.52\mu m$. Then, the iDDPL standard cells have been placed side-by-side inside the rails. The placement have been done following the bit-slice structure of the circuit. In the design enough room has been reserved to host the polarization element, which is placed in the high part of the core. The polysilicon capacitances of the polarizer are quite big and occupy much area than expected. In any case, as discussed in previous paragraph more efficient strategies can be adopted to generate V_{bias} .

The clock tree has been inserted in the middle part of the design. Clock buffers have been placed near the clock pins of the respective logic domain in order to minimize the propagation time as well as the clock skew. It must be remarked that clock skew can be positive or negative, according to the direction of the clock signal with respect to the signal flowing in the circuit. In general, a positive clock skew reduces the maximum working frequency, whereas a negative clock skew can dangerously impact the requirement on the hold time [101]. For these reasons, clock buffers have been connected in the inverse direction (right to left) with respect to the signal flow (left to right), so that the occurring clock skew is negative, impacting at least the maximum clock frequency, but without any effect on the hold time. In general, a good clock tree design must balance as well as possible the arrival times of the clock at each device in order to balance the clock latency from the clock source (i.e. the clock IO pad) to the gate of the devices and result in a reduced absolute value of the skew.

The V_{bias} net has been routed in parallel to V_{DD} , whereas the CK net in parallel to GND in order to avoid cross-talk effects. Finally the routing of the interconnect wires of the cells has been done using the Automatic Route Tool of Virtuoso without any constraint and using all the 7 metal layers. The layout of the core is depicted in Fig. 4.10.

Around the core we have also created the power rings for the power nets: 6 VDD nets for each power domain and 1 GND net, using the highest metal layers in this technology (i.e. M6 and M7). These wires have been drawn with a width equal to $1.3\mu m$, which is enough to support the overall current of the circuit.

DRC verification, parasitic extraction and cross-talk

The layout of the chip has been processed for the final verification with Calibre. DRC and LVS are clean and the circuit meets all the design rules. Then the netlist with all the parasitics annotated has been generated for post-layout simulations

(*CalibreView* in Cadence).

An important issue is the analysis of the distribution of the parasitic capacitances of the design after the automatic routing procedure. Even if the circuit is quite small, after the routing phase the capacitances of the differential interconnect wires are expected to be mismatched; even if the processor tries to minimize the wire paths, there is no constraint on the balance of the differential wires and the parasitic capacitances are distributed in a non predictable way. The mismatch of the differential capacitances for the S-Box block is represented in Fig. 4.11; a moderate unbalance (i.e. $MF > 2$) is detectable in correspondence to the output differential wires of the flip-flops, whereas for the combinational gates of the S-Box the mismatch factor is low (i.e. $MF \in [1, 2]$).

Note that the active area is rather limited (less than $3.000\mu m^2$ considering also the polysilicon capacitances of the polarizer). Even if some free room has been left between the cells in order to minimize the parasitic capacitances on the wires and reduce the mismatch, the level of unbalance is noticeable, and it is expected to dramatically increase in bigger designs. In order to take into account and provide a modelization of this phenomenon, we have inserted two parallel polysilicon capacitances on one of each differential wire; the capacitances are equal to 3fF and 6fF respectively and each one is attached to the netlist by a switching transistor; through 2 control bits it is possible to externally control the amount of additional capacitance C_{add} to be inserted on the wires: 0fF, 3fF, 6fF or 9fF. The new mismatch factor MF' is greater than the nominal case:

$$MF' = \frac{\max[C'_{L1}, C'_{L2}]}{\min[C_{L1}, C_{L2}]} = \frac{\max[C_{L1}, C_{L2}] + C_{add}}{\min[C_{L1}, C_{L2}]} = MF + \frac{C_{add}}{\min[C_{L1}, C_{L2}]} \quad (4.9)$$

Dynamic designs are sensitive on cross-talk: floating wires can be influenced by adjacent wires. Even if the layout has been carried out in order to guarantee a minimum safety distance on the most critical wires, the amount of cross-talk on the differential dynamic nets has been also verified by extracting the cross-coupling capacitances in Calibre (C + CC). The most critical wires are the internal nets of combinational gates and master latches of the flip-flops, which are floating during a half circuit before being refreshed. DRC extraction proves that the amount of capacitive cross-talk is limited (in the order of some hundred of aF on the critical wires) and balanced on the differential wires, therefore coupling does not represent an issue and the functionality of the circuit is not affected, as it will be proved by post-layout simulations. Cross-coupling inductances are not extracted, being working frequencies very low.

As a final remark, the red zones in the layout shown in Fig. 4.10 represent the polysilicon capacitances of the polarization element; as already discussed, in practical projects where the voltage V_{bias} cannot be externally modified, more efficient solutions can be designed in order to generate a fixed voltage and save area.

Condition on the maximum number of stages on the critical path

Parasitic capacitances of the interconnect wires increase the propagation delay of the combinational logic gates and impact the timing constraints of the circuit. Post-layout simulations indicate that the maximum propagation delay of AND/NAND

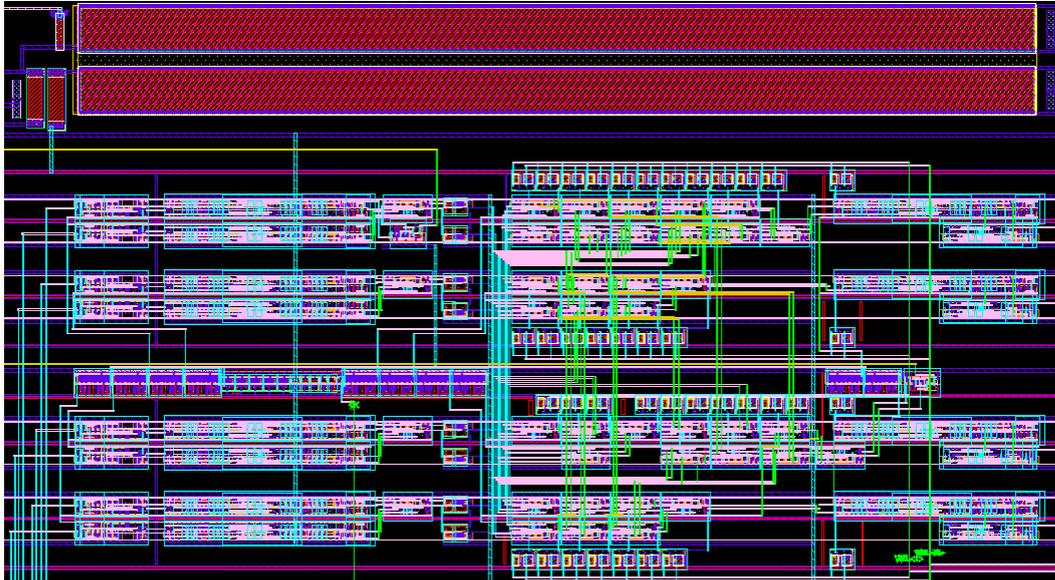


Figure 4.10. Layout of the iDDPL core of the SERPENT-block.

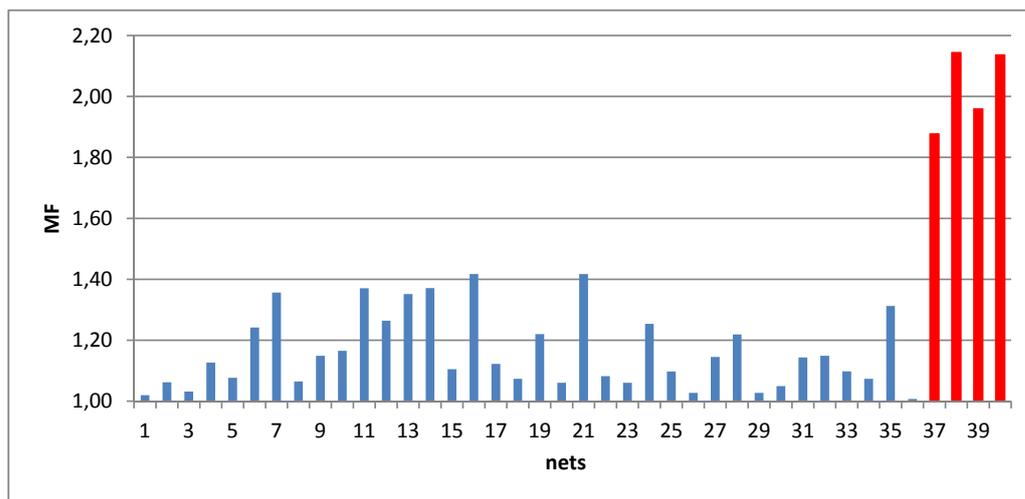


Figure 4.11. Mismatch factor for the differential nets of the S-Box block of the iDDPL circuit: combinational gates (blue) and of flip-flops (red) output wires.

gates in the pipeline is equal to 135ps, slightly higher with respect to the value found in Table 3.9. Then, it must be verified that the value chosen for δ is compatible with the length of the critical path in post layout, also in the worst case of mismatch (see Sec. 3.3.4).

The critical path of the design is composed of 8 logic gates. In the nominal case, considering the value in post layout (i.e. $t_{su} = 90ps$, $\Delta\delta_{max} = 135ps$) and using $\delta = 1.044ns$, we can estimate the maximum number of gates for the critical path:

$$\bar{N}_{max} = \left\lfloor \frac{\delta - t_{su}}{\Delta\delta_{max}} \right\rfloor = \lfloor 7.1 \rfloor = 7.1 \quad (4.10)$$

which is lower than the length of the critical path of the design. However, in presence of mismatch variations, this value can be even lower than 7. According to the plots of Fig. 3.20, $\sigma \approx 0.01 \cdot \delta$. Thus, the actual maximum number of stages in the pipeline should be:

$$N_{MAX} = \left\lfloor \bar{N}_{max} + \frac{3\sigma}{2\Delta\delta_{max}^2} \cdot [3\sigma - \sqrt{9\sigma^2 + 4\Delta\delta_{max} \cdot (\delta - t_{su})}] \right\rfloor = \lfloor 6.5 \rfloor = 6 \quad (4.11)$$

In order to take into account the additional delay due to the parasitic capacitances and also the mismatch variations, there are two possible strategies to overcome this problem: re-synthesize the S-Box with a smaller number of combinational stages in the critical path or increase the value of δ . We have chosen the second possibility, which can be simply executed by a slight change of the value of V_{bias} instead of re-doing the design. It must be pointed out that if the value of δ cannot be changed (e.g. the security requirements are strict), the only possibility is to re-synthesize the design in order to reduce the number of stages in the critical path. An efficient value for δ is about 1.3ns. Using this value, we obtain:

$$N'_{MAX} = \bar{N}_{max} + \left\lfloor \frac{3\sigma}{2\Delta\delta_{max}^2} \cdot [3\sigma - \sqrt{9\sigma^2 + 4\Delta\delta_{max} \cdot (\delta - t_{su})}] \right\rfloor = \lfloor 8.13 \rfloor = 8 \quad (4.12)$$

which is compatible with the length of the critical path of the design. In order to re-set the value of δ from 1.044ns up to 1.3ns, a slight change of R_0 and R_1 can be made in order to modify V_{bias} (in the order of few tenth of mV in accordance to the tuning range curve).

Post-layout simulations and maximum working frequency

Post-layout simulations of the circuit with annotated capacitances, both parasitics and cross-coupled, have been executed. Results are reported in Fig. 4.12 for $f_{CK} = 10MHz$ and $V_{DD} = 1.2V$.

The functionality of the chip is confirmed for different operating frequencies between 1MHz and 100MHz, even if the circuit is not expected to work at so high frequencies. In order to measure the clock skew after layout, the clock signals at the output of the clock buffers are plotted in Fig. 4.13. From this figure it is visible that, although the deviation of the clock signals is increased by the clock skew, the effect is acceptable. If compared to Fig. 4.5, the propagation time is increased from 205ps to 291.7ps, and the clock skew from 7.9ps to 10.82 ps; in any case the percentage of

skew on the propagation time is almost the same (i.e. 3.7%).

In Fig. 4.14, an histogram representing the distribution of the values of δ on all the differential signal pair is plotted in the case of nominal process parameters. The part of the graph within the red axes between $\delta = 1.3ns$ (i.e. the value at the output of the flip-flop) and $\delta = t_{su} = 90ps$ (i.e. the setup time of the flip-flop) defines the functionality mask of the chip in accordance to the timing constraints. As previously described, the nominal δ is equal to 1.3ns, which ensures an adequate margin in order to guarantee that the differential signals at the output of the critical path are such that $\delta > t_{su}$. Indeed it must be ensured that the values of δ in each part of the chip fall within the mask. The difference between $\delta_{CP} \approx 190ps$ and the setup time $t_{su} \approx 90ps$ is just the margin on δ , which is necessary in order to preserve functionality also in the worst case of mismatch variations.

According to the analysis of the timing constraints of iDDPL executed in previous chapter, we can calculate two outstanding operating frequencies: the maximum frequency to preserve the TEL encoding f'_{MAX} and the maximum frequency to preserve the functionality of the circuit f_{MAX} :

$$f'_{MAX} = \frac{1}{2 \cdot (t_{CK-Q} + \delta + T_2)} \approx \frac{1}{2 \cdot (0.1n + 1.3n + 0.7n)} \approx 240MHz \quad (4.13)$$

$$f_{MAX} = \frac{1}{2 \cdot (t_{CK-Q} + T_1)} \approx \frac{1}{2 \cdot (0.1n + 1.2n)} \approx 380MHz \quad (4.14)$$

In the range of frequencies between about 240MHz and 380MHz, the chip works with a reduced level of security. For frequencies beyond 380MHz, the circuit starts to fail to sample data and the chip does not work anymore. It must be pointed out that for frequencies in the order of some hundreds of kHz, the flip-flops start to suffer from charge sharing effect on the internal floating wires, which is common in dynamic circuits.

A final verification has been done to evaluate the residual noise of clock switching on the static voltage V_{bias} . The waveform V_{bias} is plotted in Fig. 4.15. As expected, the residual noise is a superimposed periodic waveform with the same period of the clock, with a peak-to-peak value equal to about 5mV. Even if noise amplitude is less than 1% of the nominal value, it may create a residual deviation from the expected value of δ (in the range of some tenth of picoseconds), and this must be taken into account in post-layout simulations. Again, this problem could be overcome by sizing P2 with a higher L, so that it works in the linear zone of the tuning range curve and the variability of δ decreases with V_{bias} .

4.2.4 Design of the SABL sub-block

Together with the iDDPL circuit, we have designed a SABL implementation of the same circuit in order to provide the iDDPL core with a reference circuit for functionality evaluation and security comparison. For this purpose, we have designed a basic SABL065 cell library which contains AND/NAND, OR/NOR, XOR/NXOR gates, as well as IO converters and flip-flops. The cells have been built using the circuit scheme found in literature, whereas the layout has been done by preserving the symmetric topology of the gates and balancing the internal wires

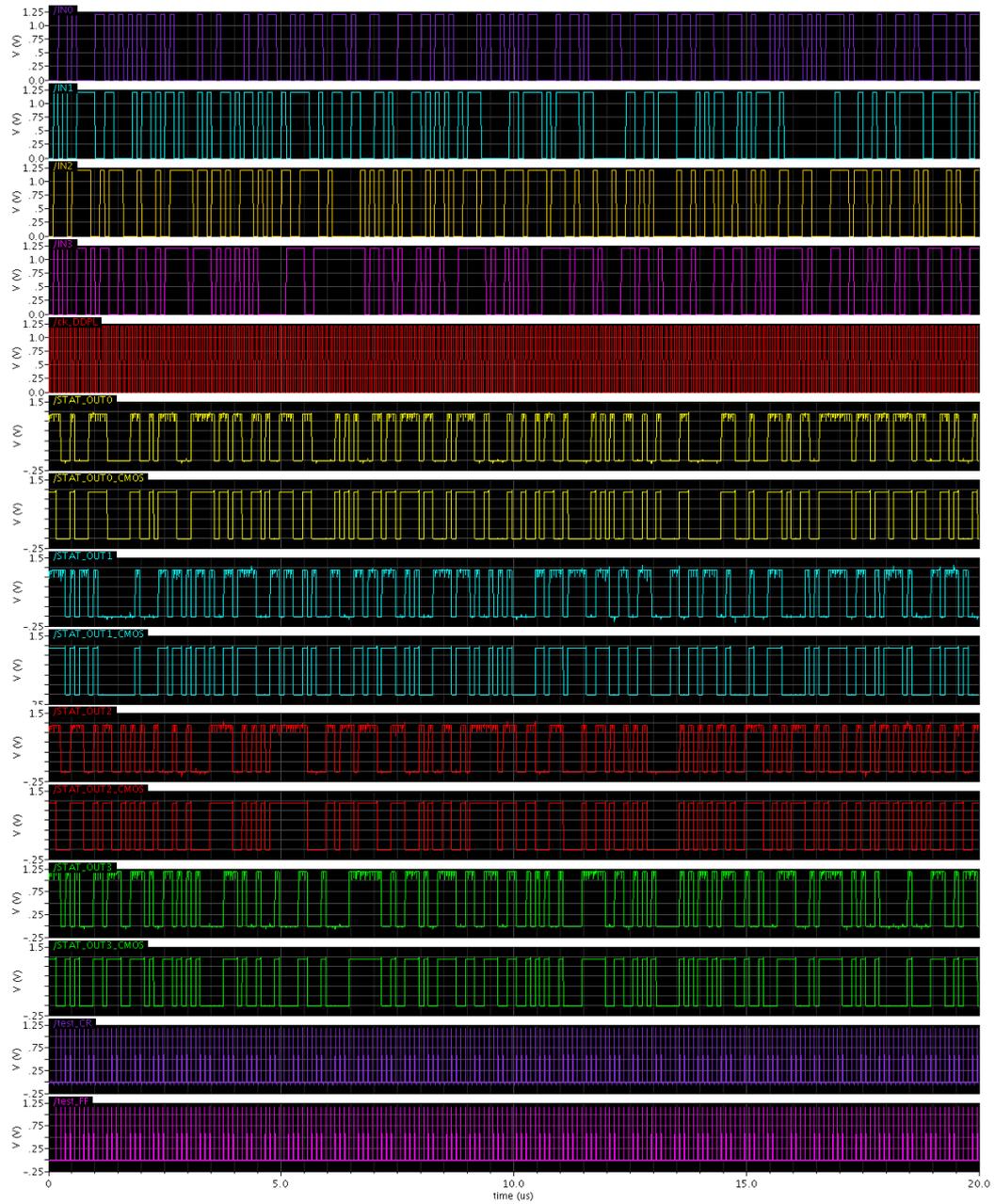


Figure 4.12. Waveforms of the input/output, clock and test signals of the iDDPL circuit after layout.

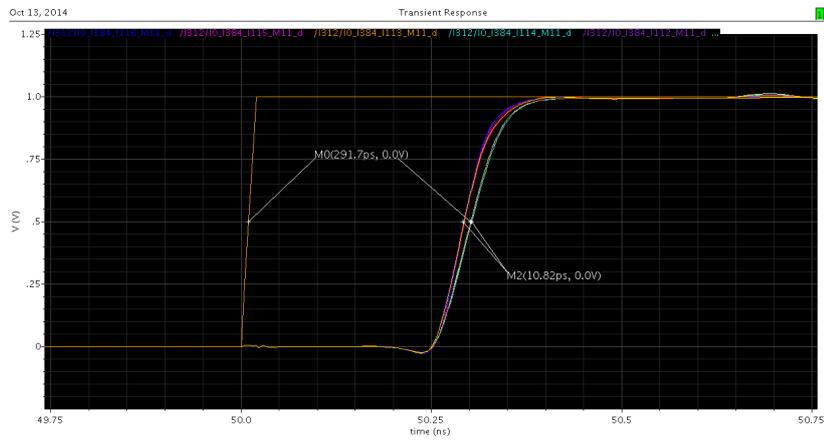


Figure 4.13. Deviation of the clock signals of the circuit due to the clock skew after layout.

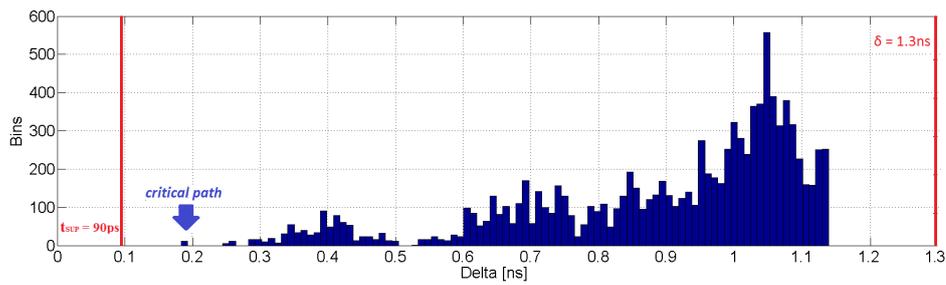


Figure 4.14. Distribution of the value of δ on each differential combinational net of the iDDPL implementation of the S-Box for every input data.

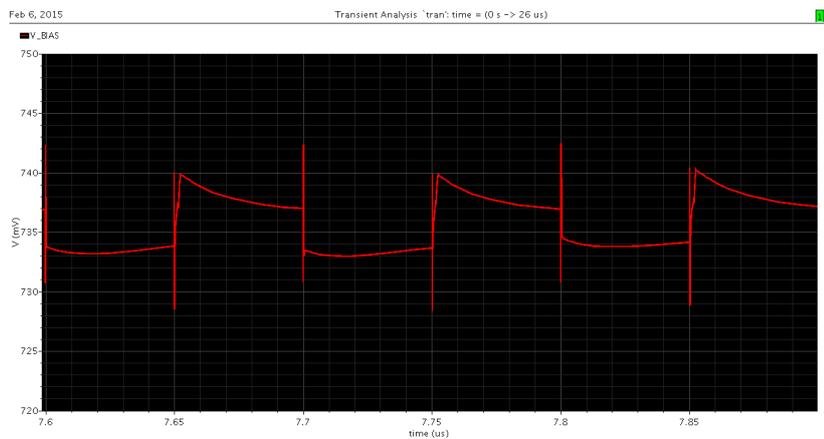


Figure 4.15. Waveform of V_{bias} signal after post-layout simulations.

Table 4.1. Performances of the designed SABL and iDDPL circuits compared to CMOS.

	# Transistors	Area overhead	Active area [μm^2]	max[P_{AV}] @10MHz [μW]	max[f_{CK}] [MHz]	Throughput
CMOS	564	x1.0	n.a.	n.a.	n.a.	x1
SABL	1538	x2.7	1703	23.3	215	$\approx x\frac{1}{2}$
iDDPL	1983	x3.5	2323	36.1	380 (240)	$\approx x\frac{1}{2}$

as much as possible. The design procedure is similar to the case of the iDDPL circuit, therefore the detailed description is omitted. The cells have been manually placed side-by-side and the routing has been done automatically in Virtuoso Layout Editor. The layout of the SABL circuit is depicted in Fig. 4.16 The overall mismatch factor on the differential wires is in accordance to the values obtained in the iDDPL implementation, and this allows to have a fair comparison.

4.2.5 Design of the complete SERPENT-block

Final architecture of the core

The two circuits have been placed side-by-side inside a single macro. They share the ground ring, but have separated power nets (6 for the iDDPL and 2 for the SABL circuit).

An overview of the architecture of the block is reported in Fig.4.18. Provided that only one circuit at a time can run, we have inserted some multiplexers which send the input and the key bits from the input pads into the active core, according to the value of the SEL bit; in the same way, some de-multiplexers take the output bits from the active core and send them into the output pads.

The performances of the two implementations have been tested in post-layout and compared to a CMOS implementation (Table 4.1). As expected, the iDDPL circuit leads to a higher area and energy overhead with respect to SABL, but the difference is slight. With respect to CMOS, the area overhead factor is about 3.5, as expected. Finally, the maximum working frequency of the iDDPL is set by the minimum value of δ , which is equal to about 1ns; but the chip continues working with a RTZ data encoding.

The total number of IO pins of the SERPENT-block is 17: 4 input pins for the plaintext, 4 input pins for the key, 1 input pin for the bit which selects the active core, 2 input pins for setting the additional amount of mismatch, 1 input pin for the clock of the iDDPL circuit, 1 input pin for the clock of the SABL circuit, and 4 output pins for the ciphertexts. The multiplexers require a specific power net, which is provided through the global net VDD_SYS . By considering also the power supply voltages, the pins of the macro are 28, and the SERPENT-block will require 28 pads in the padding of the final chip.

Layout and generation of LEF file

The final layout of the SERPENT-block is shown in Fig.4.17, where also the power rings are depicted. In order to use the full-custom chip as a macro-block for the place and route in SoC Encounter, we have extracted the *Library Exchange Format*

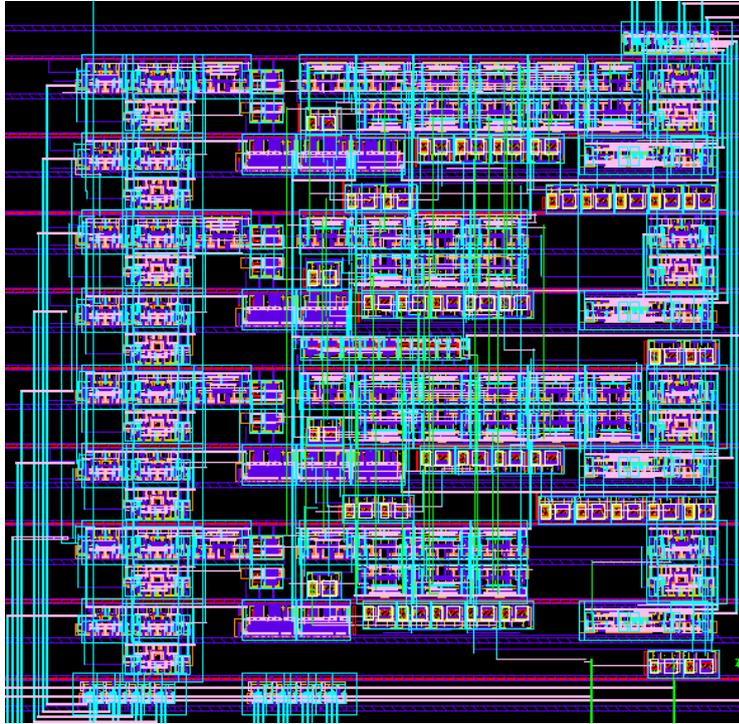


Figure 4.16. Layout of the SABL core of the SERPENT-block.

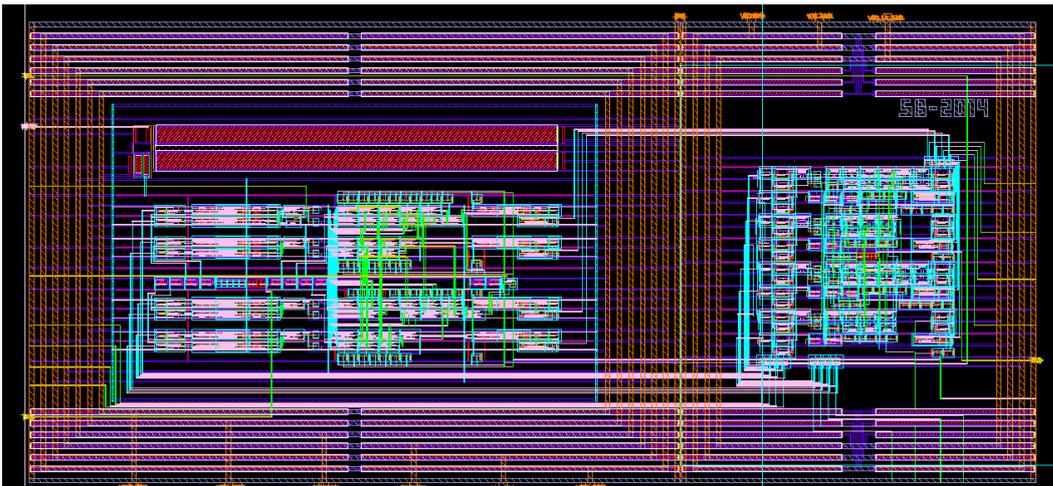


Figure 4.17. Complete layout of the SERPENT-block.

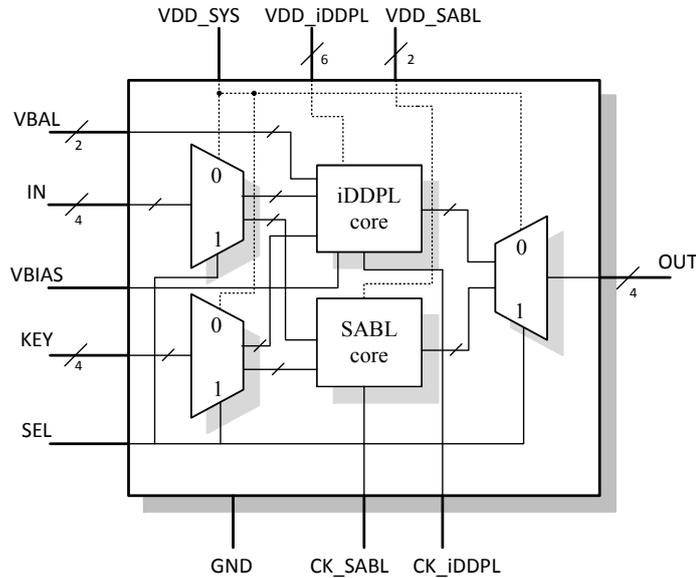


Figure 4.18. General architecture of the SERPENT-block.

(*LEF*) file of the block. *LEF* file is a specification to represent the physical layout of an integrated circuit in an ASCII format, and includes design rules and abstract information about the cells. More specifically this file contains all the geometrical information of the design, such as the size of the core, a description of each pin (shape, direction, metal layer, the exact xy-locations) and of each geometry shape (metal layer, the exact xy-location), and the obstruction shapes, that are the areas where the routing is forbidden. A screenshot of the *LEF* file of the SERPENT-block is represented in Fig. 4.19.

4.3 Description of the AES-block

4.3.1 From FPGA to ASIC

In this section we briefly describe the sub-blocks of the AES-block of SERPAES. The cryptographic cores which compose the AES-block have been developed in our department. The effectiveness of these cores in protecting the AES cipher against PAAs has been proved through extensive experimental measurements on a Cyclone FPGA [78] [79]. The results of PAAs showed good results when implemented on the above mentioned FPGA. However experiments on FPGA have the drawback of being strongly dependent on the synthesis: as discussed in [127], the feasibility of PAAs in terms of number of traces depends on whether the synthesizer uses the RAM cells of the FPGA or also the combinational elements.

Another issue on the validation of these countermeasures on FPGA regards the power consumption of the cores, which is also strongly dependent on the synthesis. In order to have a good criterion to evaluate the countermeasures, it would be useful to compare their power consumption. However, the synthesizer usually performs synthesis optimizations without guaranteeing the same initial conditions

```
NAMECASESENSITIVE ON ;
BUSBITCHARS "[]" ;
UNITS
  DATABASE MICRONS 1000 ;
END UNITS

MACRO SERPENT-block
CLASS BLOCK ;
SIZE 313.000 BY 153.500 ;
SYMMETRY R90 X Y ;

-- list and description of the pins
PIN CKD
  DIRECTION INPUT ;
  PORT
    LAYER M5 ;
    RECT 0.875 23.005 1.305 23.435 ;
  END
END CKD

[...]

-- obstruction area
OBS
  LAYER M1 ;
  RECT 1.305 4.285 288.115 136.495 ;
  LAYER M2 ;
  RECT 1.305 4.285 288.115 136.495 ;
  LAYER M3 ;
  RECT 1.305 4.285 288.115 136.495 ;
  LAYER M4 ;
  RECT 1.305 4.285 288.115 136.495 ;
  LAYER M5 ;
  RECT 1.305 4.285 288.115 136.495 ;
  LAYER M6 ;
  RECT 1.305 4.285 288.115 136.495 ;
  LAYER M7 ;
  RECT 1.305 4.285 288.115 136.495 ;
END
END SERPENT-block

END LIBRARY
```

Figure 4.19. LEF file of the SERPENT-block (screenshot).

not only between two different coprocessors, but also between two versions of the same coprocessor. Sometimes a slight change in the VHDL code is enough to get great differences in the use of the resources of the FPGA. So, for an evaluation of the differences of the current absorption it would be appropriate to have all the different versions of the countermeasured cores implemented with the same conditions.

In order to solve all the limitations due to the FPGA synthesis, the AES cores have been synthesized to be manufactured on a specific chip, which guarantees the same working conditions for the experiments and allows to do a fair comparison between power and area overhead.

4.3.2 Architecture of the AES-block

In this paragraph the architecture of the AES-block is described. The AES-block is composed of 5 cores, listed below and described in more detailed in the next paragraphs:

- A basic AES encryption unit (AES-0) [75].
- The random precharged interleaved pipeline countermeasure (AES-1) [23].
- The random interleaved pipeline countermeasure (AES-2) [23].
- The XOR-series countermeasure (AES-3) [78] [79].
- The XOR-parallel countermeasure (AES-4) [78] [79].

As it will be shown in next paragraph, AES-0 is a basic AES processor implemented with combinational logic. The other four cores are obtained by changing the VHDL code of AES-0 in order to modify the data-path and enhance the PAAs resistance of the encoder. AES-0 is characterized by a simple and easy-to-modify pipelined architecture. Each countermeasure consists in a modification of the pipeline through the insertion of some additional logic elements such as registers and multiplexers and random or pseudo-random data along the pipeline itself. All these countermeasures have been designed in our department and published [23] [78] [79], and the VHDL codes are available in our laboratory where the ASIC has been designed.

During the activity of the chip, one core at a time can run, while the others stay in the stand-by mode. This operation can be externally controlled by a 3-bit word. In the chip there is a control logic which selects the active core according to the logic state of the input word, and at the same time selects the *start* and the *done algorithm* signals which sets the beginning and the end of the algorithm, respectively. Each one of the five words among the 8 possible combinations (i.e. 000, 001, 010, 011, 100) univocally corresponds to a core. In order to efficiently control this operation from the external environment, three dip switches could be adopted. At the output of the circuit, also an input/output control block is inserted which works like an interface for the serial input/output data. The block *key* indicates that the key is hardwired in the chip and fed into all the cores.

The output signals, such as *ciphertext output*, must be connected through multiplexers to their own main output signals in order to minimize the fanout which

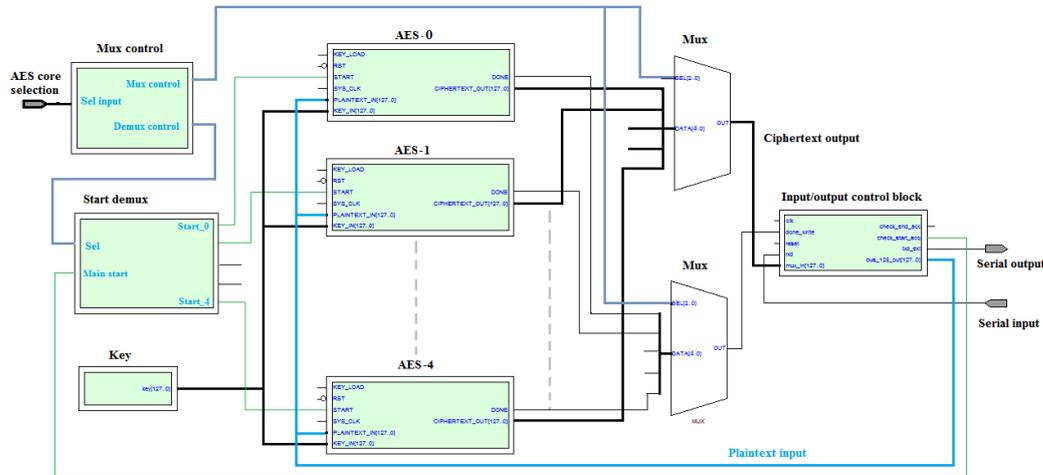


Figure 4.20. Simplified schematic diagram of the AES-block of the SERPAES chip.

can create problems during the synthesis. All the input signals from the external world except the *start* signal will be directly connected to each core. A simplified schematic diagram of this concept is depicted in Fig. 4.20.

In figure, the *Random Number Generator (RNG)* and the global signals, such as the *clock* and *reset*, are not indicated. There are several possible implementations of RNG in literature (one for example is in [24]), in this case a simple *Linear Feedback Shift Register (LFSR)* is used as pseudo-RNG. The input/output block and the selection blocks may differ according to the VHDL code and the synthesis settings.

4.3.3 The basic AES encryption unit (AES-0)

The AES encryption unit shown in Fig. 4.21 represents the basic cryptographic circuit we have implemented in the SERPAES chip (AES-0) [75]. The data path of the core is characterized by an iterative structure, with an inner-round pipeline composed of four separate blocks, each performing one layer of the AES round: *AddRoundKey*, *SubByte*, *ShiftRows* and *MixColumn*. Each of these four blocks is a combinational network, and at the output there is a state register where the datum is stored. In the architecture of the core, also a finite state machine, some additional control logics to interface the data, and the key scheduler of AES are present, but they are not depicted in Fig. 4.21.

The data-path processes data blocks of 128 bit; the input and output 128 bit buses are separated, and the time to load a plaintext in the pipeline and to send a ciphertext is equal to one clock cycle.

Each combinational layer elaborates during a clock cycle, and at the clock edge the datum is stored in the output register; thus four clock cycles are required to elaborate a round, and the entire encoding process of a block of plaintext (10 rounds plus a final 3-layers round) takes 44 clock cycles, thus the throughput is about 3 bytes per clock cycle.

Some published works showed that this crypto-core is vulnerable if for example the output of the *SubByte* block is attacked. In [127] the PAAs resistance of the

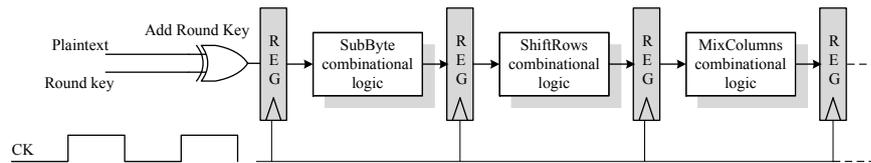


Figure 4.21. Block diagram and clock signal of the basic AES encryption unit (AES-0).

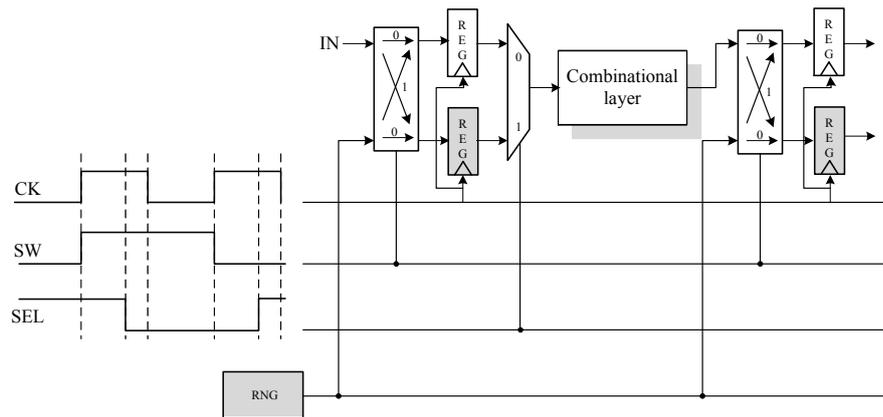


Figure 4.22. Block diagram and synchronization signals of AES-1.

core is relatively low when the S-Box is synthesized with the RAM cell of the FPGA, whereas if the SubByte block is synthesized using combinational cells (e.g. wired registers and multiplexers), more plaintexts are needed. In any case, the core can be attacked with around 100k traces.

4.3.4 The random precharged interleaved pipeline countermeasure (AES-1)

The data-path of the first countermeasure for AES-0, shown in Fig. 4.22, has been presented in [23], and experimentally tested in [127]. The countermeasure can be applied to each combinational layer of the architecture of the AES processor shown in Fig. 4.21.

In the data-path of the countermeasured core, each register is doubled and a commutator, implementable for example with 2 multiplexers, is used to store the real datum and a random quantity produced by the RNG in the two parallel registers. When a random datum is stored in one of the registers, the real datum is stored in the other one and viceversa, then a multiplexer feeds the combinational logic alternating the correct and the dummy data. The combinational networks are precharged. In the following paragraphs, we refer to this core as AES-1.

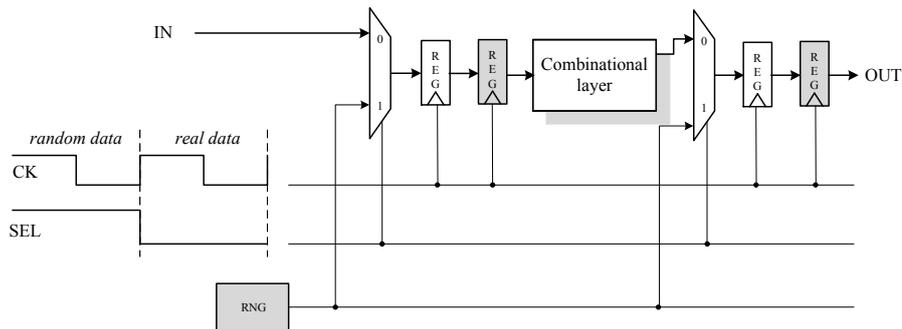


Figure 4.23. Block diagram and synchronization signals of AES-2.

4.3.5 The random interleaved pipeline countermeasure (AES-2)

The second countermeasure is published in [23], and experimentally tested in [127] (Fig. 4.22) after a slight modification of the data-path (Fig. 4.23).

The random precharged interleaved pipeline of the previous core is changed by removing the commutators and the parallel registers, and inserting two cascaded registers and a multiplexer at the beginning of each layer. In this design, the multiplexer selects the real and the dummy data alternatively, thus real and random data pass through each register in sequence, and both data streams are processed by the pipeline simultaneously. This architecture requires a doubled clock frequency in order to ensure the same data rate of the original core; furthermore also the power consumption is expected to double. In the following paragraphs, we refer to this core as AES-2.

4.3.6 The XOR-series countermeasure (AES-3)

The third countermeasure is presented and experimentally tested in [78] [79] (Fig. 4.24).

This architecture is based on the presence of two countermeasure to protect the data-path of the algorithm. The first countermeasure is conceived to protect the round 0 of the algorithm by duplicating the XOR gates and combining the datum with the key and with the logical inverted key. This way, the output of a XOR is always the negated of the other one. Basically the duplication of the XOR implements the same principle adopted to design dual-rail logic families but on an architectural level.

The second countermeasure is based on the duplication of the accumulator registers at the output of each layer block. Accordingly, the complementary data at the output of the XOR gates are multiplexed with the dummy data generated by the RNG and processed by two parallel branches consisting of two cascaded registers, which implement the random interleaved principle described in the previous paragraph. After being processed by the *SubByte* block, the data are again stored in two output cascaded register for the next combinational layers.

The above described countermeasure has the purpose of protecting each stage of the processor. The duplication of the XOR gates aims at balancing the power

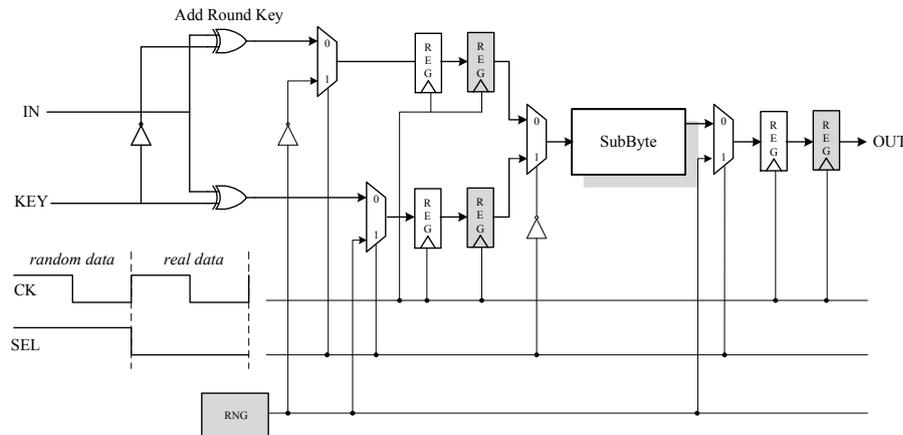


Figure 4.24. Block diagram and synchronization signals of AES-3.

consumption of the *AddRoundKey* of round 0; the duplication of the accumulators in two parallel branches enhances the resistance of the entire processor by using a random interleaved pipeline at the beginning of each layer block. This architecture has the drawback of requiring a clock with a double frequency with respect to AES-0. In the following paragraphs, we refer to this core as AES-3.

4.3.7 The XOR-parallel countermeasure (AES-4)

The fourth and last countermeasure is presented and experimentally tested in [78] [79] (Fig. 4.25).

Similarly to the case of XOR-series countermeasure, the *AddRoundKey* layer is based on the XOR duplication (Fig. 4.24). On the contrary, the random interleaved pipeline inserted to protect the other layers is based on the presence of a commutator, similarly to the case of the random precharged interleaved pipeline shown in (Fig. 4.22).

In order to further improve the "randomicity" of the pipeline and the security of the other layers, the parallel registers after the commutator are clocked with a halved version of the clock signal, which can be generated by a counter; furthermore, the output multiplexer is provided with the *SEL* signal sampled by $\frac{CLK}{2}$, so that the precharge is done during the first semi-period of the clock. In the following paragraphs, we refer to this core as AES-4.

A further description of these cryptographic cores, as well as an accurate evaluation of their security on FPGA, can be found in the aforementioned papers. Furthermore, a complete dissertation can be found in [127].

4.4 Logic synthesis of SERPAES

In this section the logic synthesis of the SERPAES core is described in more detail. The synthesis has been done using *Design Compiler* (DC) from Synopsys in Linux environment. DC offers the possibility to use a TCL shell-command interface,

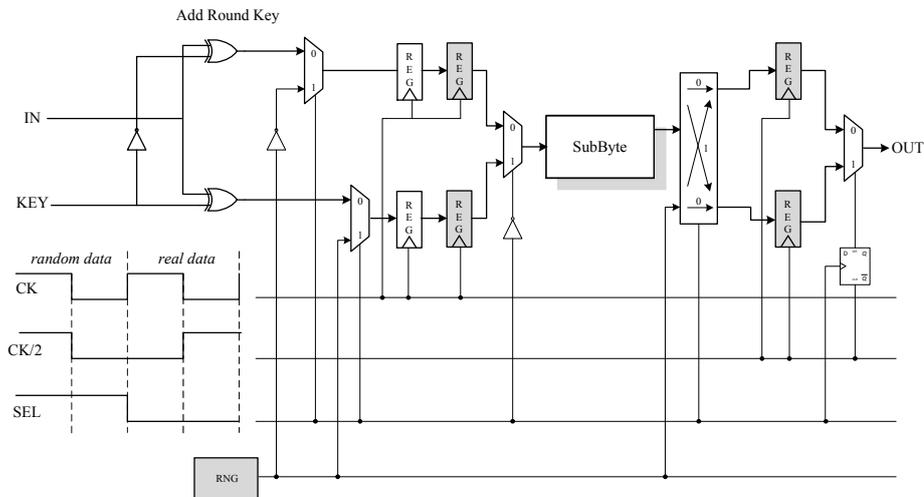


Figure 4.25. Block diagram and synchronization signals of AES-4.

named *dc_shell*, or a GUI application based on a user window, named *design_vision*. Using TCL scripts offers the designer a wider range of design settings and the possibility to customize the flow, for this reason the synthesis has been performed using *dc_shell*; anyway, *design_vision* has been also used to display the synthesized netlist and verify the correctness of the design.

The basic synthesis flow is shown in Fig. 4.26. The design steps will be described in next sections.

4.4.1 General information on the synthesis methods in DC

Before describing the synthesis steps, some general information regarding the synthesis in DC is provided [119].

Synopsys recommends different synthesis strategies which depend on the structure of the design. The most common are:

- *Top-down hierarchical compile synthesis.*
- *Bottom-up synthesis or time-budgeting synthesis.*
- *Compile characterize write script recompile (CCWSR) synthesis.*
- *Design budgeting synthesis.*

The synthesis strategies are described in the following [13].

1. *Top-down hierarchical compile synthesis.* Top-down approach is usually adopted when the design is small and can be described by a single VHDL source file. Using this method, the source is compiled by reading the entire design, thus the design constraints and attributes are applied only at the top-level entity. Another advantage is that the processor elaborates the project as a single unit and allows to better optimize the design. However this synthesis methodology is slow and not flexible: if a part of the design is changed, all the project

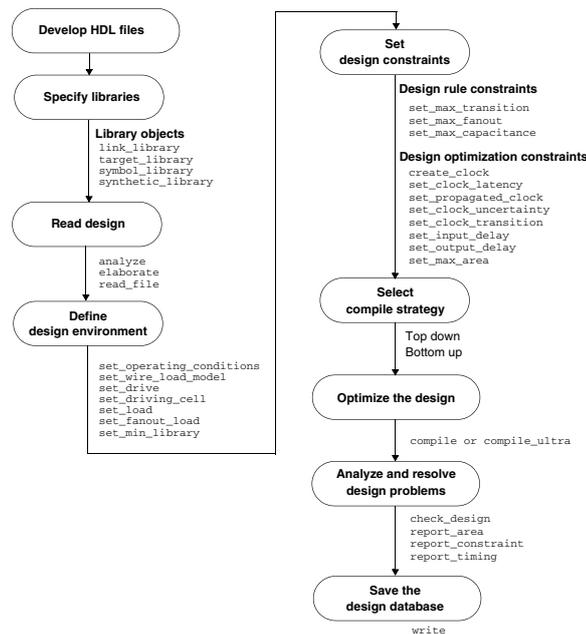


Figure 4.26. Block diagram of the basic synthesis flow in Synopsys DC [119].

must be re-synthesized; for example, any incremental change to the sub-blocks requires complete re-synthesis. Furthermore, performances are reduced if the design requires more than one clock signal (also internally generated).

2. *Bottom-up synthesis or time-budgeting synthesis.* Bottom-up synthesis is suitable for medium or medium-big designs. In this synthesis method the design has been partitioned properly with timing specifications defined for each block of the design, i.e. the designer has time budgeted the entire design, including the inter-block timing requirements. Therefore, each block has a synchronization signal with specific timing constraints, and requires to be synthesized individually. Many synthesis scripts must be used, one for each internal module, starting from low level blocks up to the top (bottom-up synthesis). Multi-scripts synthesis allows to manage several clock signals, one for each module, thus there is no need of re-synthesizing all the design if only one module is changed (e.g. incremental changes to any block). However the compilation time is higher and the update of the script can be difficult if the design has several modules; an incremental compilation is usually performed in order to fix DRC errors. Furthermore, a non-critical signal path at low level can become critical at the top-level. In general, a bottom-up is better than a top-down synthesis because it is more flexible and easy to be optimized.
3. *Compile characterize write script recompile (CCWSR) synthesis.* This strategy is used when the design is big and no inter-blocks specifications are defined. First, each sub-block is compiled, then the constraints are provided once at the top-level, finally they are loaded by each single block, which is re-compiled. The CCWSR synthesis takes less memory and requires a script for each block,

optimizing the synthesis. However synthesis scripts are complex and not easy to be updated. Furthermore once a script is changed, all the design must be recompiled, and during the synthesis step it could be difficult to achieve convergence between blocks.

4. *Design budgeting synthesis.* This method is useful when no good inter-block specifications are available, but cannot be used directly from the RTL stage. The design must be synthesized first to a mapped gate level netlist before it can be budgeted, and the top-level design specifications are automatically allocated to the lower-level blocks. Once the design is synthesized the budgeter is run on the entire design and scripts are generated for the sub-blocks. The number of hierarchical levels to budget is under full control of the user. Namely, the budgeter will allocate budgets for any number of hierarchical levels that is defined by the user. The generated scripts with accurate constraints are subsequently used to re-synthesize each block in parallel (to reduce runtime) with accurate constraints. The design budgeting method is very useful for gate-level optimization, but requires a long iterative process. In addition, this method can be used even after the layout stage in order to produce more accurate constraints. This is accomplished by budgeting the back-annotated design.

4.4.2 Synthesis design flow for SERPAES

As described in previous sections, the high level specifications of the blocks of SERPAES have different sources: the cores of the AES-blocks are described through VHDL files, whereas the SERPENT-block has been pre-designed at layout level in Cadence Virtuoso environment and exported as a single macro-cell described by the *.lef* file (Fig. 4.19). Each module requires specific synchronization signals. All the VHDL source files are compiled starting from the bottom level blocks up to the top level block, in accordance to the bottom-up synthesis procedure.

Logic synthesis in DC requires three main elements: the HDL source codes written in VHDL (file format *.vhd*) or Verilog (file format *.v*), the physical and geometrical description files of the standard-cells library (file format *.lib* and *.lef*) and the synthesis command scripts (file format *.tcl*). In our design the HDL description of all the hardware modules is provided through VHDL files.

In order to integrate these files in a single synthesis flow, the procedure consists of two sub-parts:

1. *Behavioral synthesis of the AES-block.* The VHDL source files describing the internal blocks of the AES cores are compiled and elaborated with a behavioral synthesis using the above described bottom-up approach; the AES-block has been then tested through pre-synthesis simulations using a VHDL testbench file.
2. *Structural synthesis of the SERPAES chip.* The AES-block and the SERPENT-block have been elaborated in a structural synthesis which generates the files for the final back-end design flow; for this purpose, the VHDL file *SERPAES.vhd* has been developed, in which the SERPAES chip is described as a single entity,

and in the architecture of the entity all the digital modules which must be used in the synthesis (i.e. the AES-block, the SERPENT-block, the IO pads, etc.) are instantiated as single components.

We point out that at this level of elaboration, the SERPENT-block is handled as a black-box to be placed as a macro-cell during the back-end flow, whereas all the other instantiated blocks will be described through the technology files available in the target library.

As it will be described in next section, both steps of synthesis are characterized from two sub-phases: in the first phase, the source files are analyzed and pre-compiled to obtain a RTL netlist, then the netlist is built using the logic gates of the technology library with their physical description. In accordance to this synthesis procedure, four different scripts are launched in succession, first for the behavioral synthesis of the AES-block and then for the structural synthesis of the entire SERPAES chip:

```
AES-block\_elab.tcl
AES-block\_sint.tcl
SERPAES\_elab.tcl
SERPAES\_sint.tcl
```

These scripts will be described in more detailed in the following sections.

4.4.3 Preliminary steps before the synthesis

Setup of the program

Before setting the command shell in the program path and launching the line *DC_shell*, the configuration file of Synopsys DC *.synopsys_dc.setup* must be adequately set. This file contains project- or design-specific variables that affect the optimizations of the design, such as the technology libraries and other settings, more specifically:

- the values of the variables used during the synthesis;
- the current work directory;
- the target library path for the digital blocks to be used in the project ;
- the link library path for files and libraries to be added into the synthesis;
- the symbol library path for the logic symbol views used in the schematic;
- the rules to avoid errors in the command lines on the name of cells, nets and pins of the design.

After having opened the configuration file, the following command lines have been inserted at the top of the document:

```
#DEFINITION OF THE LIBRARIES FOR THE CMOS065LPSVT TECH
#DEFINE SEARCH PATH
set search_path [list /{tech_folder_path}
/sw_vlsi/synopsys/syn/libraries/syn]
#DEFINE TARGET LIBRARY
```

```

set target_library [list /{tech_folder_path}/CORE65LPSVT_wc_1.05V_125C.db ]
#DEFINE LINK LIBRARY
set link_library [list {*} /{tech_folder_path}/CORE65LPSVT_wc_1.05V_125C.db
/{tech_folder_path}/IO65LPSVTHVT_COMPENSATION_1V8_2V5_50A_wc_1.05V_125C.db
/{tech_folder_path}/IO65LP_TF_BASIC_50A_ST_7M4XOY2Z_wc_1.05V_125C.db
/{tech_folder_path}/IO65LPSVTHVT_TF_BDPROG_1V8_50A_7M4XOY2Z_wc_1.05V_1.65V_125C.db]
#DEFINE SYMBOL LIBRARY
set symbol_library [list /{tech_folder_path}/CORE65LPSVT.sdb]

```

The target library specifies the name of the technology library that corresponds to the library whose cells the designers want DC to infer and finally map to. It contains all the information on the components that must be inserted in the design. In the case of SERPAES we have chosen the tech library CMOS065LPSVT as target library, distributed by ST Microelectronics and described in the introduction of this thesis [117].

Physical parameters describing the behavior of the digital components are contained in a database where different *cellViews* are available. Each *cellView* is accompanied by a *.lib* and a *.db* text file. The *.lib* file is the *liberty* file that can be used by DC, and contains several information on the physical characteristics of the logic gates in the library, such as the type of logic function, the footprint, the pin capacitances, the delays for different fanout and rise/fall times, and the power consumption. The *.db* file is just a compiled (binary) version of *.lib* files. In the *.synopsys_DC.setup* file we set the target library as the *.db* of the digital logic gates in the CORE65LPSVT library (i.e. the instantiated digital components).

The link library defines the name of the libraries that refer to the cells used solely for reference and not inferred by DC. It is a list of the tech libraries describing other components to be used in the design, for instance (but not only) the compensation cell (COMPENSATION lib), the power supply pads (BASIC lib) and the IO signal pads (BDPROG lib), which will be described in next sections. In the symbol library database file (*sdb*) all the graphical *cellView* descriptions of the gates (symbolic) are collected.

The physical behavior of the logic cells depends on the corner, thus in the technology library of CMOS065LPSVT process several *lib* and *.db* files exist for each corner. Accordingly, the worst case condition has been chosen for the synthesis, with a maximum power supply voltage equal to 1.05V and a temperature equal to 125°C.

Finally, the *symbol_library* contains all the information regarding the graphical representation of the cells in the technology library. It is used to represent the gates schematically using the graphical front-end tool of DC (file format *.sdb*).

Definition of the technology file

After having defined the variable describing the technology libraries for the DC setup, the technology file *technology.tcl* of the design is manually created. The command lines in the file are reported below:

```

#PATH AND LIBRARIES DEFINITION
#SET VARIABLES FOR THE TECHNOLOGY
set BufferName HS65_LS_CNBFX62
set CTPrebuffName HS65_LS_CNBFX62
set Process CMOS065

```

```

set TechnologyLib CORE65LPSVT
set CLKlib_db_path /{tech_folder_path}/CLOCK65LPSVT_wc_1.05V_125C.db
set ClockTechnologyLib CLOCK65LPSVT
set MaxCondition wc_1.05V_125C
set MinCondition bc_1.25V_125C
set LibMax wc_1.05V_125C
set LibMin bc_1.25V_125C
#SET VARIABLES FOR PATHS AND LIBRARIES
set MainPath .
set DDCPath $MainPath/DDC
set NetlistPath $MainPath/NETLIST
set LinkPath /{tech_folder_path}/CORE65LPSVT_wc_1.05V_125C.db
set TargetPath /{tech_folder_path}/CORE65LPSVT_wc_1.05V_125C.db
#DEFINE SEARCH PATH
set SearchPath /home/DK/CMOS065/CORE65LPSVT_5.1/libs/
#DEFINE TARGET LIBRARY
set target_library [list /{tech_folder_path}/CORE65LPSVT_wc_1.05V_125C.db]
#DEFINE LINK LIBRARY
set link_library [list {*} /{tech_folder_path}/CORE65LPSVT_wc_1.05V_125C.db
/{tech_folder_path}/IO65LPSVTHVT_COMPENSATION_1V8_2V5_50A_wc_1.05V_125C.db
/{tech_folder_path}/IO65LPSVTHVT_TF_BDPRG_1V8_50A_7M4XOY2Z_wc_1.05V_1.65V_125C.db]
#DEFINE SYMBOL LIBRARY
set symbol_library [list CLOCK65LPSVT.sdb]
#DEFINE CLOCK LIBRARY
set ClockLibPath /home/DK/CMOS065/CLOCK65LPSVT_3.1/libs

```

It must be noted that some information contained in the *technology.tcl* file is just the same than the DC setup file. In addition, the clock library has been inserted, where all the *.db* files describing the buffers for the clock tree are located. This file will be used after compiling the scripts during the synthesis steps, in order to map the logic cells into a physical netlist and give a first estimation of the area, power, and timing performances of the design for the following timing analysis.

Definition of the timing constraints file

The digital modules in the AES-block require two clocks and a reset signal, already defined in the VHDL source files. The timing constraints of the chip must be defined at pre-layout in order to allow the synthesizer to perform an optimize synthesis. For this purpose a timing constraint file is prepared (*constraints_2clk.tcl*) where the parameters of the clock signals are defined. The file is shown below.

```

#TIMING CONSTRAINTS
#CLOCK PARAMETERS
set fClockScaling_1 1.0
set fClockScaling_2 2.0
set period 10
set period_1 [expr period / $fClockScaling_1]
set period_2 [expr period / $fClockScaling_2]
#DEFINITION OF THE FIRST CLOCK SIGNAL
create_clock -period $period_1 -waveform [list 0 [expr $period_1 / 2]] $CLK_1_name
set_clock_uncertainty -setup 0.1 [get_clocks $CLK_1_name]
set_clock_uncertainty -hold 0.1 [get_clocks $CLK_1_name]
set_clock_transition 0.1 [get_clocks $CLK_1_name]
#DEFINITION OF THE SECOND CLOCK SIGNAL
create_clock -period $period_2 -waveform [list 0 $period_2 / 2] $CLK_2_name

```

```

set_clock_uncertainty -setup 0.1 [get_clocks $CLK_2_name]
set_clock_uncertainty -hold 0.1 [get_clocks $CLK_2_name]
set_clock_transition 0.1 [get_clocks $CLK_2_name]
#DEFINITION OF THE SETTTLING TIMES
set_input_delay -max [expr $period_1/3] -clock $CLK_1_name [remove_from_collection
[all_inputs] "$CLK_1_name $RESET1_name"]
set_output_delay -max [expr $period_1/3] -clock $CLK_1_name [all_inputs]
#FIX HOLD TIMES
set_fix_hold $CLK_1_name
set_fix_hold $CLK_2_name
#AREA OPTIMIZATION DISABLED
set_max_area 0
#SET MAXIMUM FANOUT FOR THE CELLS
set_max_fanout 16 $active_design
#SET BUFFERS IN IO/OUT PATHS
set_fix_multiple_port_nets -feedthroughs -outputs -buffer_constants

```

In the following sections, we will refer to *CLK_1* as *SYS_clk* and *CLK_2* as *clk2*. After having defined some global variables such as clock periods and scaling factor, clock signals are generated using the command *create_clock*. Given that our design is expected to work at frequencies in the order of some MHz, we have defined a clock period equal to 10ns for the global clock *CLK_1*. This high value for the working frequency (i.e. 100 MHz) provides a high degree of freedom in case of any setup violations, and allows to have a large margin to fix timing errors. Note that *CLK_2* has a double period with respect to *CLK_1*. The period of the clock is set through the *-period* option, whereas the duty cycle of the clock is set to 50% through the *-waveform* option. It must be noted that the clock is parametric, and any change in the settings can be efficiently done by modifying the value of the scale parameters *fClockScaling_1* and *fClockScaling_2*.

The command *set_clock_uncertainty* defines the amount of skew. Basically this skew is inserted already during the synthesis in order to have a certain amount of margin on the clock, both for the setup and the hold time. The uncertainty is set to 100ps both on setup and hold, which is 1% of the clock period and represents a pessimistic case.

The command *set_clock_transition* indicates a value for the transition at each port/pin: given that before the layout the clock tree is not present in the chip, clock signal is distributed as a single net for all the logic gates and the DC associates a high load to the clock source; thus forcing a clock transition enables the compiler to calculate realistic delays for the logic gates being fed by the clock and allows to avoid any error in their output transition.

The command *set_input_delay* specifies the arrival time of the signals at the input ports with respect to the edge of the clock *CLK_1_name*. The *-max* option indicates that the signal must be stable within a certain value, in this case a third of the clock period, thus it defines the settling times for input signals before (setup time) and after (hold time) the clock edge. This command has been applied for each input apart from clock and reset signals. Similarly, the command *set_output_delay* with the *-max* option specifies the maximum arrival time before the edge of *CLK_1_name*.

The command *set_fix_hold* has the purpose to fix hold violations at registers during compilation with respect to the clock signals. This command inserts some buffers at appropriate locations in order to delay the data with respect to the clock.

Fixing gross hold time violations already during synthesis helps to reduce the layout iterations.

Other optimization tasks are executed by the script before doing the synthesis. The command *set_max_area* specifies the maximum allowable area for the design; given that fixing hold time violations increases the area overhead, setting this option to 0 helps to minimize area overhead during optimization. The command *set_max_fanout* specifies a maximum fanout constraints on every driving pin and input port. Finally the command *set_fix_multiple_port_nets* inserts a buffer in direct paths between input and output ports (feedthrough) in order to isolate input and output ports and, at the same time, drive the output port; the *-buffer_constants* option enables the compiler to buffer the constants driving the output port.

4.4.4 Logic synthesis of the AES-block

Analysis of the VHDL source files

Synthesis in Synopsys can be executed in two ways: using the *read* command or using the *analyze/elaborate* commands. The *read* command is preferred to elaborate pre-compiled design or netlist, whereas the *analyze/elaborate* commands are more powerful and efficient to synthesize a netlist from VHDL or Verilog source files.

Therefore, as first step we use the commands *analyze/elaborate* to compile the VHDL files. The *analyze* command initially verifies if any syntax error occurs in the VHDL files and executes the translation in a RTL netlist of combinational and sequential gates contained in an internal Synopsys independent library (GTECH); these logic cells are unmapped representations of Boolean functions and serve as a place holders for the technology dependent library, which is executed by the command *elaborate*. The command lines are reported below.

```
#ANALYZE/ELABORATE THE AES-BLOCK SOURCE FILES
#CLEAR OLD DESIGN REFERENCES
remove_design -designs
#PATH AND LIBRARIES DEFINITION
set des AES-block
set MainPath .
set DDCPath $MainPath/DDC
set NetlistPath $MainPath/NETLIST
set VHDLPath $MainPath/vhdl
#ANALYZE BOTTOM LEVEL MODULES
analyze -format vhd $VHDLPath/AES_128_0.vhd
analyze -format vhd $VHDLPath/AES_128_1.vhd
analyze -format vhd $VHDLPath/AES_128_2.vhd
analyze -format vhd $VHDLPath/AES_128_3.vhd
analyze -format vhd $VHDLPath/AES_128_4.vhd
...
#ANALYZE TOP LEVEL BLOCK
analyze -format vhd $VHDLPath/AES_128_top.vhd
#BEHAVIORAL SYNTHESIS OF THE AES-BLOCK
elaborate AES-block -arch "Behavioral" -library WORK -update
#RESOLVE MULTIPLE INSTANCE REFERENCES
uniquify
#GENERATE NETLIST AND QUIT
write -format ddc -hierarchy -output $MainPath/$DDCPath/${des}_elab.ddc
quit
```

In the header of the script, the command *remove_design* first removes all the references to previous designs in the memory; then the variable describing the library paths are defined, such as the design name, the local directory path, the folder where the internal DC database file (format file *.ddc*) is generated, the folder where the netlist (format file *.v*) is created and the folder from which VHDL source files are read.

As first synthesis step, the VHDL source files are analyzed one by one through the command *analyze -format vhdl path/name-module*. With this command the program compiles the digital modules of the AES-block individually, elaborates the packages used in the entity modules of the VHDL files, and check the variable, constant, type, function, structure and process definitions. Then, the compiler creates an intermediate file containing the definitions in the package and all the digital components of the entity; in this file also the interconnections and the required information are located. Note that in accordance to the bottom-up synthesis, all the low-hierarchy blocks are first compiled; finally, the top level block is elaborated by the compiler.

Then, the command *elaborate* executes the synthesis of the design, generating a RTL netlist of logic gates. It must be noted that at this design step no information about the physical characteristics of the logic gates is still available because DC has not recalled the technology file *technology.tcl*. The attribute *-arch "Behavioral"* indicates that the synthesis is behavioral, and this must be taken into account during post-synthesis simulations.

The command *uniquify* avoids that multiple instances are inserted in the top hierarchy block and that different netlists are connected to the same port of an entity.

Finally the command *write* saves the hierarchy of the design in the *.ddc* file, that is a Synopsys format file which describes the pre-compiled netlist with all the information related to the design as area, number of cells, ports, nets, etc. At this abstraction level, a schematic level description of the top level entity is provided according to the internal GTECH library. The synthesizer will associate each symbolic gate to a standard-cell of the technology library, according to the information contained in the *.ddc* file.

Synthesis of the AES-block

The second step of the synthesis of the AES-block is executed by the script *AES-block_sint.tcl*. The DC takes the pre-compiled netlist *AES-block_elab.ddc* generated by the script *AES-block_elab.tcl*, executes the synthesis by substituting each standard cell in the netlist with the specific logic gates in the technology library, and optimizes the design after some iterations in order to meet the constraints. The command lines are reported below.

```
#SYNTHESIS OF THE AES-BLOCK
#CLEAR ALL DESIGN REFERENCES AND SET ACTIVE DESIGN
remove_design -all
set active_design AES-block
#DEFINE CLOCK AND RESET SIGNALS
set CLK_1_name SYS_clk
set CLK_2_name clk2
```

```

set RESET1_name reset
#PATH AND LIBRARIES DEFINITION
set MainPath .
set DDCPath $MainPath/DDC
set NetlistPath $MainPath/NETLIST
#RECALL THE TECHNOLOGY FILE
source ./scripts/tecnology.tcl
#READ AND SET THE PRE-COMPILED NETLIST
read_ddc "$DDCPath/${active_design}_elab.ddc"
current_design $active_design
#LINK TOGETHER ALL DESIGN PARTS
link
#RECALL THE CONSTRAINTS FILE
source $MainPath/scripts/constraints2clkA.tcl
#DEFINE CONSTRAINTS FOR THE WIRES
set_wire_load_mode enclosed
set_dont_touch_network [list $CLK_2_name $RESET1_name]
#OPTIMIZE SYNTHESIS
compile -map_effort high
#FIX GROSS HOLD VIOLATIONS
fix_hold clk2
#REPORT AND WRITE OUTPUTS
report_area
write -format ddc -hierarchy -output $DDCPath/$active_design.ddc
change_names -rules verilog -hierarchy -verbose > $MainPath/change_names/${active_design}_change_names_verilog.log
write -format verilog -hierarchy -output $NetlistPath/$active_design.v $active_design
report_timing
write_sdc ./sdc/$active_design.sdc
write_sdf -version 2.1 -significant_digits 3 ./sdf/${active_design}_max.sdf
quit

```

The first command lines have the purpose of removing any references to old designs (command *remove_design*) and defining the environment variables. For this purpose, the variable *active_design* allows to define and parametrize the name of the module to be synthesized (i.e. AES-block) so that it can be used throughout the script. It is important that the value of the design variable is the same as the entity of the top level block. Also the clocks and the reset signal of the design are declared.

After having recalled the above described technology file, the command *read_ddc* is used to load the *.ddc* file generated by the script *AES-block_elab.tcl*. The command *current_design* takes the variable *active_design*; this command is important when a hierarchical project is synthesized, and if neglected the netlist can be incomplete.

The command *link* solves design references, and links together all the parts of the design. If needed, some standard-cells from the technology library can be also excluded by the synthesis through the command *source ./scripts/dont_use_cells.tcl* moreover design errors can be checked already at this step through the command *check_design*. However these two commands are omitted in the script in order to reduce synthesis iterations.

The command *source ./scripts/constraints_2clk.tcl* recalls the constraints file describing clock signals. The command *set_wire_load_mode* has the purpose of providing a model of the connections inside the project by using the parameters defined in the tech library. The option *enclosed* indicates that the DC must get the wire models by the top-hierarchy blocks. Instead the command *set_dont_touch_network*

prevents any change or substitution of the nets indicated in the list (*CLK_2_name* and *RESET1_name*), avoiding the insertion of clock buffers.

The command *-map_effort* allows to set the synthesis effort; increasing the efforts boosts the probability of having good results but increases also the synthesis time. The attribute *high* drives the DC to optimize the synthesis of the logic in the critical path in order to meet the timing constraints; hold errors detected on the *clk2* net in a previous iteration of the script can be fixed by the command *fix_hold*.

The commands *report_area* and *report_timing* allow to save reports for area and timing, and this will be described in next paragraph.

Finally, the last commands are executed to save the output files. First, the DC writes the *.ddc* file where all the components in the hierarchy are saved, as well as all the information related to the netlist such as area, number of cells, ports, nets, etc. Then, the command *change_names* changes the names of all ports, cells and nets according to the denomination rules, in order to prevent any syntax errors when the Verilog netlist (file format *.v*) is written and then imported by other tools. After exporting the Verilog netlist, the DC saves the *.sdc* and the *.sdf* files. Files *.sdc* and *.sdf* are text files and contain the information about constraints and physical parameters of the netlist, and will be described in more detail in next section.

Area and timing report

After the synthesis, the DC writes *.log* files where area and timing information on the synthesized netlist are reported.

The area report of the AES-block is obtained through the command *report_area*, and it is shown in Fig. 4.27. It describes the number of ports, nets, cells, combinational gates, sequential gates, macros, buffer/inverter cells, and references. Furthermore it gives an estimation of the combinational, non combinational, net interconnect and overall area. At this step there is no information about place and route, thus these parameters are only approximated. It must be noted that in accordance to the chosen wire load model, the synthesizer cannot give an estimation of the net interconnect area, thus the amount of total area is a rough estimation.

The timing report of the AES-block is obtained through the command *report_timing*, and it is shown in Fig. 4.28 and Fig. 4.29 for clock signals *SYS_clk* and *clk2* respectively. The timing report highlights how well the compiler did during the synthesis about speed. It lists the worst-case timing path (i.e. the critical path) in the circuit in terms of delay at each stage of the path for each clock group, and for each clock group a startpoint, an endpoint, the number and the name of cells through which the signal propagate, and the relative delay are reported. The synthesizer calculates the data arrival time by adding up these delays. Then it calculates the data required time in accordance to the speed target defined in the timing constraint file (e.g. clock period, clock network delay, and clock uncertainty) and the library setup time. Finally, the slack is calculated as the difference between the required time and the arrival time. The slack is the difference between the time instant expected for a signal to arrive at a logical gate and the time instant in which the signals actually arrives. Therefore, a positive slack means that a signal arrives at a logical gate before the required instant and meet the timing constraints. It must be pointed out that the timing analysis has been done only on the setup time, thus

```

*****
Report : area
Design : AES_128_block
Version: E-2010.12-SP2
Date   : Wed Feb 19 10:56:59 2014
*****

Information: Updating design information... (UID-85)
Warning: Design 'AES_128_block' contains 2 high-fanout nets. A fanout number
of 1000 will be used for delay calculations involving these nets. (TIM-134)
Library(s) Used:

      CORE65LPSVT (File:
/home/DK/CMOS065/CORE65LPSVT_5.1/libs/CORE65LPSVT_wc_1.05V_125C.db)

Number of ports:          13
Number of nets:          1921
Number of cells:         1012
Number of combinational cells: 609
Number of sequential cells: 393
Number of macros:        0
Number of buf/inv:       83
Number of references:    23

Combinational area:      181100.396282
Noncombinational area:  91971.876660
Net Interconnect area:   undefined (Wire load has zero net area)

Total cell area:        273072.272942
Total area:             undefined

```

Figure 4.27. Report of the area occupation of the AES-block

the information is related to the worst *reg-to-reg* paths which is critical and limits the clock frequency. If the timing constraints on the setup time are satisfied, in the report we find *slack(MET)* and the relative delay margin.

4.4.5 Logic synthesis of the SERPAES chip

Design of the SERPAES top entity

The second part of the synthesis flow has the purpose of putting together the AES-block and the SERPENT-block in the same design through a merely structural synthesis. For this purpose a specific top entity module must be specified, in which all the the digital modules of the chip must be instantiated as components. At this step the architectural and electrical specification of the chip must be also defined in order to define the connection of the signal nets. The output of the synthesis will represent the final netlist to be processed by the place and route processor.

For this purpose, a specific VHDL file, named *SERPAES.vhd*, has been written, which represents the top level block of the synthesis. In this file, a top entity called *SERPAES* is defined. The ports of the entity are just all the input and output signals of the AES-block and the SERPENT-block, whereas all the modules are instantiated in the architecture as hardware components, taking care of mapping the internal signals. A list of all the digital components instantiated in the top level block is below provided.

- **The AES-block and the SERPENT-block.** The two main blocks are instantiated as two separated components.
- **The IO and power supply pads.** Pad ring is located at the periphery of the chip and has the purpose to provide an electrical interface between the

```

*****
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : AES_128_block
Version: E-2010.12-SP2
Date   : Wed Feb 19 10:57:08 2014
*****
# A fanout number of 1000 was used for high fanout net computations.
Operating Conditions: wc_1.05V_125C Library: CORE65LPSVT
Wire Load Model Mode: enclosed
Startpoint: aes_serie_par_inst/sw_count_reg_0_(rising edge-triggered flip-flop clocked by clk2)
Endpoint: aes_serie_par_inst/INST_SubBytes/reg_1/dummy_reg_reg_51_(rising edge-triggered flip-flop clocked by SYS_clk)
Path Group: SYS_clk
Path Type: max
Des/Clust/Port Wire Load Model Library
-----
AES_128_block wl_zero CORE65LPSVT
AES_128_serie_par wl_zero CORE65LPSVT
SubBytes_par_0 wl_zero CORE65LPSVT
reg_1 wl_zero CORE65LPSVT
Point
-----
clock clk2 (rise edge) 5.00 5.00
clock network delay (ideal) 0.00 5.00
aes_serie_par_inst/sw_count_reg_0_/CP (HS65_LS_DFFRQX9) 0.00 # 5.00 r
aes_serie_par_inst/sw_count_reg_0_/Q (HS65_LS_DFFRQX9) 0.25 5.25 f
aes_serie_par_inst/U434/Z (HS65_LS_IVX9) 0.11 5.35 r
aes_serie_par_inst/U389/Z (HS65_LS_BFX9) 0.18 5.53 r
aes_serie_par_inst/U409/Z (HS65_LS_BFX9) 0.15 5.68 r
aes_serie_par_inst/U391/Z (HS65_LS_BFX9) 0.16 5.84 r
aes_serie_par_inst/U474/Z (HS65_LS_AO22X9) 0.19 6.03 r
aes_serie_par_inst/INST_SubBytes/SubBytes_IN[49] (SubBytes_par_0) 0.00 6.03 r
aes_serie_par_inst/INST_SubBytes/U5750/Z (HS65_LS_IVX9) 0.10 6.13 f
aes_serie_par_inst/INST_SubBytes/U138/Z (HS65_LS_BFX9) 0.15 6.28 f
aes_serie_par_inst/INST_SubBytes/U1760/Z (HS65_LS_IVX9) 0.14 6.42 r
aes_serie_par_inst/INST_SubBytes/U274/Z (HS65_LS_NOR2X6) 0.13 6.54 f
aes_serie_par_inst/INST_SubBytes/U232/Z (HS65_LS_NOR2X6) 0.21 6.75 r
aes_serie_par_inst/INST_SubBytes/U200/Z (HS65_LS_IVX9) 0.22 6.97 f
aes_serie_par_inst/INST_SubBytes/U109/Z (HS65_LS_BFX9) 0.16 7.13 f
aes_serie_par_inst/INST_SubBytes/U42/Z (HS65_LS_IVX9) 0.04 7.17 r
aes_serie_par_inst/INST_SubBytes/U3250/Z (HS65_LS_OAI22X2) 0.08 7.26 f
aes_serie_par_inst/INST_SubBytes/U3248/Z (HS65_LS_NOR4AEX2) 0.14 7.39 r
aes_serie_par_inst/INST_SubBytes/U4100/Z (HS65_LS_OAI212X5) 0.11 7.50 f
aes_serie_par_inst/INST_SubBytes/U4099/Z (HS65_LS_AO22X4) 0.28 7.78 f
aes_serie_par_inst/INST_SubBytes/reg_1/reg_in[51] (reg_1) 0.00 7.78 f
aes_serie_par_inst/INST_SubBytes/reg_1/U68/Z (HS65_LS_IVX9) 0.05 7.82 r
aes_serie_par_inst/INST_SubBytes/reg_1/U377/Z (HS65_LS_OAI22X6) 0.06 7.88 f
aes_serie_par_inst/INST_SubBytes/reg_1/dummy_reg_reg_51_/D (HS65_LS_DFFRQX9) 0.00 7.88 f
data arrival time 7.88
clock SYS_clk (rise edge) 10.00 10.00
clock network delay (ideal) 0.00 10.00
clock uncertainty -0.10 9.90
aes_serie_par_inst/INST_SubBytes/reg_1/dummy_reg_reg_51_/CP (HS65_LS_DFFRQX9) 0.00 9.90 r
library setup time -0.11 9.79
data required time 9.79
data arrival time -7.88
-----
slack (MET) 1.91

```

Figure 4.28. Time report of the critical path associated to the clock *Sys_clk*.

```

*****
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : AES_128_block
Version: E-2010.12-SP2
Date   : Wed Feb 19 10:57:08 2014
*****
# A fanout number of 1000 was used for high fanout net computations.
Operating Conditions: wc_1.05V_125C  Library: CORE65LPSVT
Wire Load Model Mode: enclosed
Startpoint: aes_serie2_xor1_inst/sw_count_reg_0 (rising edge-triggered flip-flop clocked by clk2)
Endpoint:  aes_serie2_xor1_inst/INST_SubBytes/reg_1/int_reg_reg_59 (rising edge-triggered flip-
flop clocked by clk2)
Path Group: clk2
Path Type:  max
Des/Clust/Port  Wire Load Model  Library
-----
AES_128_block  w1_zero  CORE65LPSVT
AES_128_serie_2_xor_1_1
SubBytes_serie_2_0 w1_zero  CORE65LPSVT
reg_ser128_1  w1_zero  CORE65LPSVT
Point
-----
clock clk2 (rise edge) 0.00 0.00
clock network delay (ideal) 0.00 0.00
aes_serie2_xor1_inst/sw_count_reg_0/CP (HS65_LS_DFPRQX9)
0.00 # 0.00 r
aes_serie2_xor1_inst/sw_count_reg_0/Q (HS65_LS_DFPRQX9)
0.32 0.32 f
aes_serie2_xor1_inst/U244/Z (HS65_LS_BFX9) 0.16 0.48 f
aes_serie2_xor1_inst/U176/Z (HS65_LS_IVX9) 0.11 0.59 r
aes_serie2_xor1_inst/U152/Z (HS65_LS_BFX9) 0.18 0.77 r
aes_serie2_xor1_inst/U132/Z (HS65_LS_BFX9) 0.18 0.95 r
aes_serie2_xor1_inst/U113/Z (HS65_LS_IVX9) 0.15 1.09 f
aes_serie2_xor1_inst/U233/Z (HS65_LS_AO22X9) 0.22 1.32 f
aes_serie2_xor1_inst/INST_SubBytes/SubBytes_IN[56] (SubBytes_serie_2_0)
0.00 1.32 f
aes_serie2_xor1_inst/INST_SubBytes/U5441/Z (HS65_LS_IVX9)
0.12 1.44 r
aes_serie2_xor1_inst/INST_SubBytes/U148/Z (HS65_LS_BFX9)
0.16 1.60 r
aes_serie2_xor1_inst/INST_SubBytes/U673/Z (HS65_LS_NOR2X6)
0.14 1.74 f
aes_serie2_xor1_inst/INST_SubBytes/U225/Z (HS65_LS_NOR2X6)
0.24 1.98 r
aes_serie2_xor1_inst/INST_SubBytes/U193/Z (HS65_LS_IVX9)
0.22 2.20 f
aes_serie2_xor1_inst/INST_SubBytes/U108/Z (HS65_LS_BFX9)
0.16 2.36 f
aes_serie2_xor1_inst/INST_SubBytes/U55/Z (HS65_LS_IVX9)
0.04 2.40 r
aes_serie2_xor1_inst/INST_SubBytes/U3354/Z (HS65_LS_OAI22X2)
0.08 2.49 f
aes_serie2_xor1_inst/INST_SubBytes/U3352/Z (HS65_LS_NOR4BK2)
0.14 2.62 r
aes_serie2_xor1_inst/INST_SubBytes/U5781/Z (HS65_LS_OAI21X5)
0.11 2.73 f
aes_serie2_xor1_inst/INST_SubBytes/U5780/Z (HS65_LS_AO22X4)
0.27 3.00 f
aes_serie2_xor1_inst/INST_SubBytes/reg_1/reg_in[59] (reg_ser128_1)
0.00 3.00 f
aes_serie2_xor1_inst/INST_SubBytes/reg_1/U65/Z (HS65_LS_AO22X9)
0.15 3.15 f
aes_serie2_xor1_inst/INST_SubBytes/reg_1/int_reg_reg_59/D (HS65_LS_DFPRQX9)
0.00 3.15 f
data arrival time 3.15
clock clk2 (rise edge) 5.00 5.00
clock network delay (ideal) 0.00 5.00
clock uncertainty -0.10 4.90
aes_serie2_xor1_inst/INST_SubBytes/reg_1/int_reg_reg_59/CP (HS65_LS_DFPRQX9)
0.00 4.90 r
library setup time -0.10 4.80
data required time 4.80
data arrival time -3.15
-----
slack (MET) 1.65

```

Figure 4.29. Time report of the critical path associated to the clock *clk2*.

internal logic and the pins of the chip package. Each pad includes several wide tracks and circuitry to drive IO signals, provide global power supply, guarantee ESD protection, and host the bonding pad, and must be declared as single component.

- **The corner cells.** Corner cells are important for the floorplan of the pad ring, because they provide to route the internal pad nets (for instance the VDDE and the GNDE nets, but also other control signals) and to fill the corners of the chip.
- **The compensation cell.** Compensation of process-voltage-temperature (PVT) variations is important in analogue circuits as well as in digital. The semiconductor vendors usually provide a specific cell in the technology library which senses the variations of the working conditions of the circuit and adjusts the performances by sending some bit sequences to the pads in order to change their electrical properties. There are also some control bits that can be externally programmed by the designer.
- **The IOFILLERCELL_REF_ASRC cell.** This cell is a fictitious pad, thus it is inserted in the pad ring. The purpose of this component is to route the signal to and from the core compensation cell for the pads of the padring.
- **The IOFILLERCELL_TRIGGER cell.** This cell is a fictitious pad, thus it is inserted in the pad ring. The purpose of this component is to act as trigger circuit if this option is enabled.

Furthermore, a list and the number of primary input/output and power supply nets of SERPAES are below provided. Note that the overall number is 56, which represents the minimum number of pins of the manufactured chip and thus of the pads for the pad ring.

- IO signals for the AES-block: 13
- IO signals for the SERPENT-block : 17
- Power supply signals for the pads: 8
- Power supply signals for the AES core: 8
- Power supply signals for the iDDPL core: 6
- Power supply signals for the SABL core: 3
- Power auxiliary signals: 1

Definition of the electrical specifications and interconnections

The electrical specifications of the chip and signal interconnections must be also defined in the top level block. For this purpose, the definition of the pad ring is very important and can compromise the success of the design if not accurately performed. First of all, it must be pointed out that the pads differ according to the

type of signals which they must handle. In accordance to the signal list reported in previous paragraph, there are two types of signals that can be interfaced by a pad and distributed inside the chip as global signals: input/output (IO) and power supply signals. According to the technology library, the IO pads are described in the library BDPROG, whereas the power supply pads are described in the library BASIC.

As well as the internal logic, also the pads require a voltage supply, which is distributed directly in the pad ring and usually is different from the internal voltage supply of the chip. According to the information on the technology provided in the introduction of this thesis, the nominal voltage is $VDD = 1.2V$, whereas the pads can be supplied with a voltage equal to $VDDE = 1.8V, 2.5V$ or $3.3V$, which are standard values. We have chosen 1.8V-pads for the pad ring.

In order to generate two voltages, 1.2V and 1.8V, the standard-cell library provides two different power supply pads, which must be separately instantiated in the *SERPAES.vhd* file. In our chip we have decided to dedicate 4 pads for the voltage VDDE and 4 pads for the pad ground GNDE (which is separated from the ground GND of the internal logic for electrical reasons). Then, one VDDE pad and one GNDE pad is placed on each side of the chip; this represents a standard design strategy which aims at satisfying the electrical rules defined in the library. A similar strategy is adopted for the VDD and GND pads of the internal logic.

For the SERPAES project we have chosen to define different voltage supply domains for the SERPENT-block in order to perform separate power measurements: the iDDPL core needs 6 separated voltage supply nets, whereas the SABL core needs 3 separated voltage supply nets. All these power nets have the same value 1.2V, but they must be routed as separated wires. This represents an important issue which poses a constraint on the design of the pad ring, because each power net requires a specific power supply pad, as it will be shown in next section.

The iDDPL core requires also an auxiliary static voltage V_{bias} which must be globally routed. The V_{bias} will be handled like a power supply net, therefore it will be interfaced by a VDD pad. It must be pointed out that the static voltage V_{bias} is not a power supply voltages because it must be fed on the gate of a transistor and does not require current from the external supply. However, it is more convenient to handle it like a global net during the back-end flow.

As above mentioned, each pad must be defined and instantiated in the *SERPAES.vhd* file as a single component. As it will be shown in next section, the *nanoroute* engine of SoC Encounter automatically connects the pins of the IO pads of the design to the primary input/output nets in accordance to the synthesized netlist. On the contrary, the pins of the power supply pads are left floating in the VHDL top level block: they are considered as global nets which will be preliminary routed during the floorplan phase using the *specialRoute* engine of the SoC Encounter.

The connection of the pins of the IO pads is of crucial importance and deserves a special description. According to the direction of a signal, a pad can get it from the external environment and buffer it into the internal logic (input pad), or alternatively can get it from the internal logic and sends it into the external environment (output pad). An IO pad of the tech library BDPROG has 12 pins which must be adequately assigned [117]:

<pre> P1_CK_DDPL: BDPROGSCARUDQP_1V8_TF_LIN PORT MAP (ZI => CK_DDPL_int, A => '0', EN => '1', TA => '0', TEN => '0', TM => '0', PUN1V8 => '0', PDN1V8 => '1', PROGA1V8 => '0', PROGB1V8 => '0', ENZI => '0', IO => CK_DDPL); </pre>	<pre> P1_1_STAT_OUT_DDPL_1: BDPROGSCARUDQP_1V8_TF_LIN PORT MAP (A=>STAT_OUT_DDPL_int_1, ZI=>open, EN=>'0', TA=>'0', TEN=>'0', TM=>'0', PUN1V8 => '0', PDN1V8 => '1', PROGA1V8 => '0', PROGB1V8 => '0', ENZI=>'1', IO=>STAT_OUT_DDPL_inout_1); </pre>
---	--

Figure 4.30. Screenshot of the VHDL file: assignment of the pins of an input pad (left) and an output pad (right).

- **ZI** is the output pin of the pad, which is buffered from the pad directly to the core logic; if the pad is an input pad, it is connected to one port of the entity, whereas if the pad is an output pad, it is unconnected (open).
- **EN** is the control bit which indicates if the pad is input ('1') or output ('0').
- **TA** is used in the test mode.
- **TEN** is used in the test mode.
- **TM** is the control bit which indicates if the pad is in the test mode ('1') or in the normal mode ('0').
- **PUN1V8** is used for enabling the $50k\Omega$ pull-up transistor of the logic inside the pad.
- **PDN1V8** is used for enabling the $50k\Omega$ pull-down transistor of the logic inside the pad.
- **PROGA1V8** and **PROGB1V8** are used for programming the pad to adjust performances in presence of different loads.
- **ENZI** is used to enable the input buffer ('0', input pad), or disable it ('1', output pad).
- **IO** is the input/output pin of the pad with the external world.

As an example, in Fig. 4.30 the assignment of two signals (one input and one output) in the VHDL file is shown.

Definition of the updated timing constraints file

The SERPENT-block requires two additional clock signals, one for the iDDPL and one for the SABL circuit. Thus, a novel timing constraints file containing all the clock signals must be defined, named *constraints_4clk.tcl*. The file is just equal to *constraints_2clk.tcl*, apart from the presence of the additional clocks, *CK_DDPL*

(indicated with the variable name *CLK_3_name*) and *CK_SABL* (indicated with the variable name *CLK_4_name*). We point out that these clocks are fictitious, because the clock tree of the SERPENT-block has been already designed inside the macro, but must be defined so that synthesis can be performed. Clock parameters as frequency, uncertainty and transition of the clock signals are the same used in the previous timing constraint file.

```
#DEFINITION OF THE CK_DDPL
create_clock -period $period -waveform [list 0 period/2] $CLK_3_name
set_clock_uncertainty -setup 0.1 [get_clocks $CLK_3_name]
set_clock_uncertainty -hold 0.1 [get_clocks $CLK_3_name]
set_clock_transition 0.1 [get_clocks $CLK_3_name]
#DEFINITION OF THE SECOND CLOCK SIGNAL CK_SABL
create_clock -period $period -waveform [list 0 period/2] $CLK_4_name
set_clock_uncertainty -setup 0.1 [get_clocks $CLK_4_name]
set_clock_uncertainty -hold 0.1 [get_clocks $CLK_4_name]
set_clock_transition 0.1 [get_clocks $CLK_4_name]
```

Analysis of the VHDL source files

After having defined the *SERPAES.vhd* top level file and the new timing constraints file, the structural synthesis of SERPAES is executed. Similarly to the case of the AES-block, the synthesis is done in two steps, first using the *analyze/elaborate* commands and then the *read_ddc* command in two different files.

The *SERPAES_elab.tcl* script compiles the VHDL files describing all the digital modules of the chip. Note that this may represent a redundant operation, but this time the compiler analyzes also the top level block *SERPAES.vhd* and provide to link together the design paths in accordance to a structural synthesis. The command lines are the following:

```
#ANALYZE BOTTOM LEVEL MODULES
analyze -format vhd1 $VHDLPath/AES_128_0.vhd
analyze -format vhd1 $VHDLPath/AES_128_1.vhd
analyze -format vhd1 $VHDLPath/AES_128_2.vhd
analyze -format vhd1 $VHDLPath/AES_128_3.vhd
analyze -format vhd1 $VHDLPath/AES_128_4.vhd
...
#ANALYZE TOP LEVEL BLOCK
analyze -format vhd1 $VHDLPath/SERPAES.vhd
#STRUCTURAL SYNTHESIS OF THE SERPAES CHIP
elaborate $des -arch "Struct" -library WORK -update
#RESOLVE MULTIPLE INSTANCE REFERENCES
uniquify
#LINK TOGETHER ALL DESIGN PATHS
link
#GENERATE NETLIST AND QUIT
write -format ddc -hierarchy -output $MainPath/$DDCPath/SERPAES_elab.ddc
quit
```

Note that the structural synthesis is executed through the attribute *-arch "Struct"*.

Synthesis of the SERPAES chip

The synthesis is executed by the script *SERPAES_sint.tcl*, which is reported below.

```

#SYNTHESIS OF THE SERPAES CHIP
#CLEAR ALL DESIGN REFERENCES AND SET ACTIVE DESIGN
remove_design -all
set active_design SERPAES
#DEFINE CLOCK AND RESET SIGNALS
set CLK_1_name SYS_clk_AES
set CLK_2_name clk2_AES
set CLK_3_name CK_DDPL
set CLK_4_name CK_SABL
set RESET1_name reset_AES
#PATH AND LIBRARIES DEFINITION
set MainPath .
set DDCPath $MainPath/DDC
set NetlistPath $MainPath/NETLIST
set VHDLPath $MainPath/vhdl
#READ THE PRE-COMPILED NETLIST AND THE TOP LEVEL FILE
read_file -format ddc $DDCPath/AES_128_block.ddc
read_file -format vhdl -rtl $MainPath/$VHDLPath/SERPAES.vhd -lib WORK
current_design $active_design
#LINK TOGETHER ALL DESIGN PARTS
link
#RESOLVE MULTIPLE INSTANCE REFERENCES
uniquify
#RECALL THE CONSTRAINTS FILE
source $MainPath/scripts/constraints4clkA.tcl
#DEFINE CONSTRAINTS FOR THE WIRES
set_wire_load_mode enclosed
set_dont_touch_network [list $CLK_1_name $CLK_2_name $CLK_3_name $CLK_4_name $RESET1_name]
set_dont_touch AES_128_block true
#COMPILE
compile
#REPORT AND WRITE OUTPUTS
set_dont_touch AES_128_block false
report_area
write -format ddc -hierarchy -output $DDCPath/$active_design.ddc
change_names -rules verilog -hierarchy -verbose > $MainPath/change_names/${active_design}_change_names_verilog.log
write -format verilog -hierarchy -output $NetlistPath/$active_design.v $active_design
report_timing
write_sdc ./sdc/$active_design.sdc
write_sdf -version 2.1 -significant_digits 3 ./sdf/${active_design}_max.sdf
quit

```

This script is very similar to the script *AES-block_sint.tcl*, with some important differences. First, in the script the additional clock signals for iDDPL and SABL circuits are declared. Then, the *read_file* command analyzes both the synthesized netlist of the AES-block and the previously described VHDL file describing the SERPAES entity. Another important difference is that in this script the technology file *source ./scripts/technology.tcl* is omitted because the technology mapping involves only the AES-block which has already been synthesized. We point out that at this design stage the SERPENT-block is handled by the synthesizer as a black-box and does not require the technology mapping. Finally, the command *set_dont_touch AES_128_block* with the attribute *true* prevents the synthesizer to modify the netlist *AES_128_block.ddc*.

Table 4.2. Gate count for each core of the SERPAES chip after synthesis.

Countermeasure	Core name	Number of gates
Basic AES encryption unit	AES-0	30750
Random precharged interleaved pipeline	AES-1	38780
Random interleaved pipeline	AES-2	37388
XOR-series	AES-3	40992
XOR-parallel	AES-4	41964

The remaining part of the script is just the same as the *AES-block_sint.tcl* script and it is not commented.

Area and timing report of the final design

The area report of the complete design is shown in Fig. 4.31. The area occupation is slightly higher with respect to the AES-block synthesis report, due to the presence of the pads and the other digital components which increase the overhead. As visible in figure, the DC handles the SERPENT-block as an unknown component, therefore it is not considered in the active area calculation. In Table 4.2 the number of logic gates for each single core is reported.

About the timing report, we point out that *CK_DDPL* and *CK_SABL* clock signals do not require buffers because the clock tree of the iDDPL and the SABL circuits have been designed inside the macro SERPENT-block, therefore the timing report for these signals is meaningless and therefore omitted. Furthermore, the timing constraints for *clk2* and *SYS_clk* are again met. This is reasonable, because the synthesizer has not changed the synthesis done in the previous step thanks to the command *set_dont_touch_network [list clocks]*.

Output file of the synthesis

The DC generates four important files after synthesis: the *Standard Delay Format* (*.sdf*) file, the *Synopsys Design Constraint* (*.sdc*) file, the *Verilog* (*.v*) file and the already mentioned *.ddc* file.

The *.sdf* file contains information on the delays of the paths of each net (i.e. the propagation times of the loaded cells in the path and the transition times of the signals), and the delays of the interconnect wires and the timing constraints (i.e. setup and hold times) which are technology dependent. At this design step the calculation of the delay is a rough estimation based on the *wire_load_model*. This information represents a forward annotated netlist which can be used by other tools for doing the *Static Timing Analysis* (*STA*).

The *.sdc* file contains the design constraints and the timing assignments which were written in the file *constraints_4clk.tcl*, such as the period, the settling times and the uncertainty of the clock signals. This file can be used by the place and route processor (e.g. SoC Encounter) for doing a first placement of the standard-cells without violating these timing specifications at the very first step of the place and

```

Report : area
Design : SERPAES
Version: E-2010.12-SP2
Date   : Wed Feb 19 11:07:03 2014
*****

Information: Updating design information... (UID-85)
Warning: Design 'SERPAES' contains 2 high-fanout nets. A fanout number of
1000 will be used for delay calculations involving these nets. (TIM-134)
Library(s) Used:

    CORE65LPSVT (File:
/home/DK/CMOS065/CORE65LPSVT_5.1/libs/CORE65LPSVT_wc_1.05V_125C.db)
    IO65LPSVTHVT_TF_BDPROG_1V8_50A_7M4X0Y2Z (File:
/home/DK/CMOS065/IO65LPSVTHVT_TF_BDPROG_1V8_50A_7M4X0Y2Z_7.0/libs/IO65LPS
VTHVT_TF_BDPROG_1V8_50A_7M4X0Y2Z_wc_1.05V_1.65V_125C.db)
    IO65LP_TF_BASIC_50A_ST_7M4X0Y2Z (File:
/home/DK/CMOS065/IO65LP_TF_BASIC_50A_ST_7M4X0Y2Z_7.2/libs/IO65LP_TF_BASIC
_50A_ST_7M4X0Y2Z_wc_1.05V_125C.db)
    IO65LPSVTHVT_COMPENSATION_1V8_2V5_50A (File:
/home/DK/CMOS065/IO65LPSVTHVT_COMPENSATION_1V8_2V5_50A@7.0.a.UD5357/libs/
IO65LPSVTHVT_COMPENSATION_1V8_2V5_50A_wc_1.05V_125C.db)

Number of ports:                30
Number of nets:                  70
Number of cells:                 66
Number of combinational cells:   65
Number of sequential cells:      0
Number of macros:                0
Number of buf/inv:              1
Number of references:            15

Combinational area:      181102.476282
Noncombinational area:  632572.355176
Net Interconnect area:   undefined (Wire load has zero net area)

Total cell area:         813674.831458
Total area:              undefined

Information: This design contains black box (unknown) components. (RPT-8)
1

```

Figure 4.31. Report of the area occupation of the SERPAES chip.

route (timing driven placement).

The *.v* file is a Verilog description of the synthesized design. In the file all the hardware components are inserted, and all the internal nets are defined and connected.

The *.ddc* has been already described. This file can be open in the *design_vision* GUI environment and allows to load the schematic view of the final design in order to verify if the synthesis was successful. The schematic view of the synthesized core is shown in Fig. 4.32. The AES-block and the SERPENT-block are indicated with arrows.

Post-synthesis simulations

The netlist in Fig. 4.32 can be used to execute post-synthesis simulations. With this purpose, a simulation testbench has been developed. The SERPAES chip has been instantiated as a component of a VHDL file, and a stimulus file has been written. Simulations have been executed by using the *NCSim* tool from Cadence Incisive.

Post-synthesis simulations of the AES-block are in accordance with the pre-synthesis simulations, previously done on Quartus II. In Fig. 4.33 a screenshot of the simulation window, where the signal waveforms are shown for a specific input data combination, is depicted. The test word *correct_output* allows to verify the functionality of the design.

4.5 Place and route of SERPAES

4.5.1 Standard-cell layout

The design of the layout is characterized by mapping the technology cells in the netlist generated by the synthesis, and placing and routing them in accordance of some constraints and design rules. For doing so, SoC Encounter assists the designer through some automatic phases that will be described in more detail in next section.

For the design of SERPAES we have chosen the typical layout methodology of a digital circuit, which is characterized by the presence of several parallel metal stripes designed according to a comb-like geometry and dedicated to the distribution of the voltage supply of the circuit and the ground; the standard-cells are placed side-by-side between the voltage supply and the ground rails and interconnected. Between two adjacent wires, the standard-cells are alternated to an empty channel which is dedicated to the route of the interconnections. An example of the rail-to-rail layout is depicted in Fig. 4.34. The interconnection wires are realized through different level of metal; in the technology we chose for SERPAES, the number of metal layers is 7.

4.5.2 Digital back-end flow

The procedure of place and route is composed of some standard phases which are listed in the order:

1. General settings

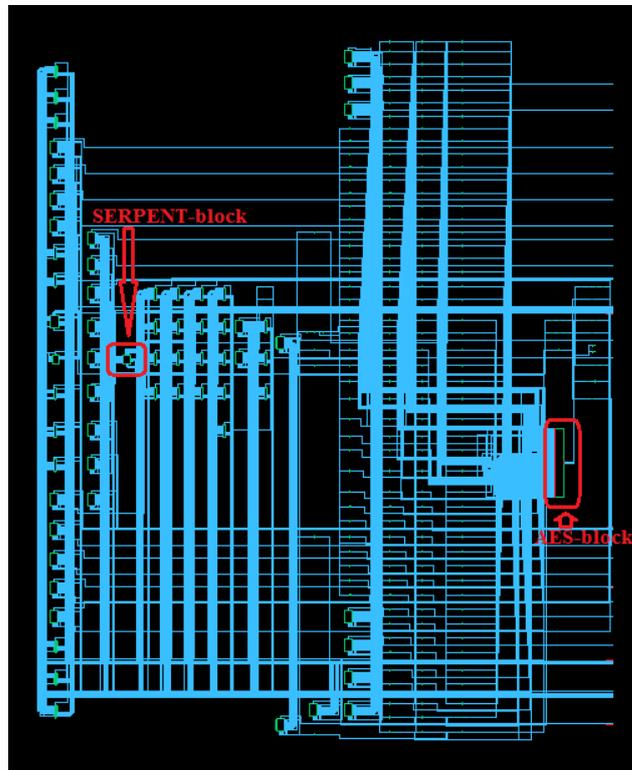


Figure 4.32. Schematic view of the SERPAES chip after synthesis



Figure 4.33. Screenshot of the post-synthesis NCSim simulation.

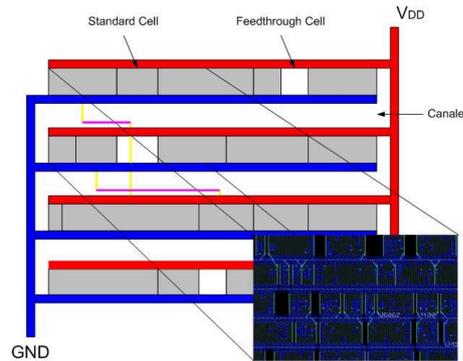


Figure 4.34. Typical layout of a digital circuit, where the standard-cells are placed side-by-side.

2. Floorplan;
3. Placement;
4. Post-placement optimization
5. Clock Tree Synthesis (-CTS);
6. Post-CTS optimization;
7. Routing;
8. Post-route optimization;
9. Sign off

Between two consecutive steps, the processor executes the Static Timing Analysis (STA) for verifying if the timing constraints are satisfied, and executes one or more optimization steps through an iterative procedure. The processor tries to optimize the design by reducing the cell size, inserting buffers, repeating the technology mapping, and so on; this efforts have the purpose of fixing design violations, improve area occupation, and so on. Script commands have been adopted in accordance to the definition found in SoC Encounter manual [26], and are described in the following paragraphs.

4.5.3 General settings

The place and route in SoC Encounter requires some input files which must be adequately specified at the beginning of the design flow: the Verilog netlist generated by the synthesis (file format *.v*), the list of technology and physical libraries (file format *.lef*), the timing libraries for the worst and the best case corners (file format *.lib*), the timing constraint file generated by the synthesis (file format *.sdc*).

All this information is stored in the file *start.info*. In this file we have also set the name of the global supply nets that require to be routed during the floorplan.

4.5.4 Floorplan

The floorplan of the SERPAES chip represented the most critical and time consuming part of the project, because it has required some manual operations. The reason is due to the fact that we have chosen to divide the power supply domain of the SERPENT-block in some sub-domains, each one with the same voltage 1.2V but separately routed. This led to the need of using different power supply pads for these nets. As it will be shown in next sections, this posed a constraint on the design of the pad ring of the entire chip.

In this section we describe the different steps of the floorplan. It was done both launching the commands in the shell and using the options in the GUI of Encounter. In the following we report the command lines for the floorplan, that will be described in more detail in next paragraphs. As usual, the lines preceded with a `#` are comments.

```
#FLOORPLAN OF THE CHIP SERPAES

#GENERAL SETUP
setup environment
restoreDesign "./dbs/setupDesign.enc.dat" ${designName}

#SPECIFICATION OF THE CORE AREA
floorPlan -site CORE -d 1106.8 1095.2 25.0 25.0 25.0 25.0

#PLACEMENT OF THE MACROS
setObjFPlanBox Instance P1_COMPENSATION 211.18 705.0 421.18 901.8
setObjFPlanBox Instance DDPL_TESTCHIP_inst 594.29 193.4 907.29 346.9

#DESIGN OF THE PAD RING
loadIoFile padring.save.io

#FILLERING THE PAD RING
addIoFiller -cell IOFILLERCUT_VDD_ST_TF_LIN -prefix FILLER -side s -from 208.400 -to
606.000
addIoFiller -cell IOFILLERCUT_VDD_ST_TF_LIN -prefix FILLER -side e -from 633.200 -to
796.400
addIoFiller -cell IOFILLER16_ST_TF_LIN
addIoFiller -cell IOFILLER8_ST_TF_LIN
addIoFiller -cell IOFILLER4_ST_TF_LIN
addIoFiller -cell IOFILLER2_ST_TF_LIN
addIoFiller -cell IOFILLER1_ST_TF_LIN
#INSERTION OF THE POWER RINGS OF THE CORE
addRing -spacing_bottom 2 -width_left 3 -width_bottom 3 -width_top 3 -spacing_top 2
-layer_bottom M7 -stacked_via_top_layer AP -width_right 3 -around core -jog_distance
2.5 -offset_bottom 3.5 -layer_top M7 -threshold 2.5 -offset_left 3.5 -spacing_right
2 -spacing_left 2 -offset_right 3.5 -offset_top 3.5 -layer_right M6 -nets { GND VDD
GND VDD} -stacked_via_bottom_layer M1 -layer_left M6

#INSERTION OF THE POWER RING OF THE MACROS
selectInst P1_COMPENSATION
addRing -spacing_bottom 2 -width_left 3 -width_bottom 3 -width_top 3 -spacing_top 2-
layer_bottom M7 -stacked_via_top_layer AP -width_right 3 -around selected -jog_dista
nce 2.5 -offset_bottom 0 -layer_top M7 -threshold 2.5 -offset_left 0 -spacing_right
2 -spacing_left 2 -type block_rings -offset_right 0 -offset_top 2.3 -layer_right M6-
nets {GND VDD } -stacked_via_bottom_layer M1 -layer_left M6
```

```

deselectAll
selectInst DDPL_TESTCHIP_inst
addRing -spacing_bottom 10 -width_left 1 -width_bottom 3 -width_top 3 -spacing_top 2
-layer_bottom M7 -stacked_via_top_layer AP -width_right 3 -around selected -jog_dis
tance 2.5 -offset_bottom 3.5 -layer_top M7 -threshold 2.5 -offset_left 0.2 -spacing_
right 2 -spacing_left 1 -type block_rings -offset_right 9.61 -offset_top 0 -layer_rig
ht M6 -nets {GND VDD } -stacked_via_bottom_layer M1 -layer_left M6
deselectAll

#INSERTION OF THE POWER STRIPES
addStripe -block_ring_top_layer_limit M7 -max_same_layer_jog_length 6 -padcore_ring_
bottom_layer_limit M5 -set_to_set_distance 100 -stacked_via_top_layer AP -padcore_rin
g_top_layer_limit M7 -spacing 2 -xleft_offset 60 -merge_stripes_value 2.5 -layer M6
-block_ring_bottom_layer_limit M5 -width 3 -nets {GND VDD } -stacked_via_bottom_lay
er M1 -break_stripes_at_block_rings 1
addStripe -block_ring_top_layer_limit AP -max_same_layer_jog_length 6 -padcore_ring_
bottom_layer_limit M6 -set_to_set_distance 100 -ybottom_offset 53.5 -stacked_via_top
_layer AP -padcore_ring_top_layer_limit AP -spacing 2 -merge_stripes_value 2.5 -dire
ction horizontal -layer M7 -block_ring_bottom_layer_limit M6 -width 3 -nets {GND VDD
} -stacked_via_bottom_layer M1 -break_stripes_at_block_rings 1

#INSERTION OF WELLTAPS AND ENDCAPS
addWellTap -cell HS65_LS_FILLERNPW4 -cellInterval 28.8 -inRowOffset 7 -checkerBoard
addEndCap -preCap HS65_LS_FILLERPPF4 -postCap HS65_LS_FILLERPPF4 -coreBoundaryOnly

#ROUTING THE GLOBAL NETS
#ROUTING THE STANDARD-CELLS WIRES AND THE CORE PINS
sroute -connect { corePin floatingStripe } -layerChangeRange { M1 AP } -blockPinTarg
et { nearestTarget } -checkAlignedSecondaryPin 1 -allowJogging 1 -crossoverViaBottom
Layer M1 -allowLayerChange 1 -targetViaTopLayer AP -crossoverViaTopLayer AP -targetV
iaBottomLayer M1 -nets { VDD GND }
#ROUTING THE BLOCK AND THE PAD PINS
sroute -connect { blockPin padPin } -layerChangeRange { M1 AP } -blockPinTarget { ne
arestRingStripe nearestTarget } -padPinPortConnect { allPort oneGeom } -checkAligned
SecondaryPin 1 -blockPin useLef -allowJogging 1 -crossoverViaBottomLayer M1 -allowLa
yerChange 1 -targetViaTopLayer AP -crossoverViaTopLayer AP -targetViaBottomLayer M1-
nets {allGlobalNets}
#ROUTING THE PAD RING PINS}
sroute -connect { padRing } -layerChangeRange { M1 AP } -blockPinTarget { nearestTar
get } -checkAlignedSecondaryPin 1 -allowJogging 1 -crossoverViaBottomLayer M1 -allow
LayerChange 1 -targetViaTopLayer AP -crossoverViaTopLayer AP -targetViaBottomLayer
M1 -nets {allGlobalNets}

#SETTING THE PINS CONSTRAINTS
setAllowedPinLayersOnEdge -edge N -layer {M2 M4}
setAllowedPinLayersOnEdge -edge S -layer {M2 M4}
setAllowedPinLayersOnEdge -edge W -layer {M3 M5}
setAllowedPinLayersOnEdge -edge E -layer {M3 M5}

#SAVING THE DESIGN
saveDesign "./dbs/floorplan.enc"

```

Specification of the core area

The first two command lines have been executed in order to load the project and the setup environment variables defined in the setting file, as for example the

name of the design (SERPAES). These lines will be repeated at the beginning of each script.

The first step of the design is the definition of the area of the chip. SoC Encounter allows different ways for doing so, for example by specifying the sizes of the core or the area, the density, the xy-dimensions, or the aspect ratio; we have chosen to set a total area for the chip equal to nearly 1mm^2 by specifying the sizes of the core with the command *floorplan* and the option *-site CORE*. The first number indicates the x-dimension, the second indicates the y-dimension; the other numbers refer to the option *-d* and indicate the distances between the internal sides of the pad ring and the sides of the die. All the numbers in Encounter are expressed in micron.

Placement of the macro-blocks

The command *setObjFPlanBox* places the macro-blocks of the chip in the die area. It is followed by the name of the instances of the chip (i.e. the compensation cell and the SERPENT-block) and the exact xy-location where the macros are wanted to be inserted.

For this purpose, we have decided to place the compensation cell in the top left corner of the floorplan, whereas the SERPENT-block has been placed in the bottom right corner. In accordance to this choice, the pads of the SERPENT-block will be located in the bottom and in the right sides of the chip, as it will be shown in next paragraphs.

Design of the pad ring

When the design is loaded for the first time, the program automatically places the pads of the chip. According to the synthesis, there are 56 pads in the pad ring, 2 fictitious pads, and 4 corner cells. Through the command *save->pad_file* in the GUI of the program, it is possible to save the pad position in the text file (*padring.save.io*), specifying the pad order for each side starting from the xy-position (0,0), or alternatively listing them according to their exact xy-location. We have chosen this second option, and the output file is a list of the pads of the chip divided for each side with the exact xy-location.

Then, we have manually changed the order of the pads in the *padring.save.io* file, taking care of correctly setting the new xy-location. We point out that the distances between the pads have been set according to the design rules defined in the technology manual, for instance the minimum pad pitch must be equal to $40\mu\text{m}$, whereas the position of a IO and a power supply pad must be adequately alternated for electrical issues. Finally, the new pad ring is loaded with the command *loadIoFile*.

As said in the previous paragraph, the placement of the pads have been optimized so that power supply pads of the iDDPL core are placed near to it, the same for the SABL core. The layout of the floorplan at this design step is shown in Fig. 4.35(a).

Insertion of the filler cells in the pad ring

Similarly to the design of the internal logic cells, also the design of the pad ring requires to be completed through the insertion of the filler cells. A critical issue in the design of the pad ring is due to the fact that we planned to keep the power

supply networks of the iDDPL and the SABL cores separated, as well as to create different internal power supply domains for both the cores. The presence of different power supply domains requires to adequately insulate the pads of different domains. In fact the layout of the pads of the technology library is designed in order that the pads are located side by side, and the lateral pins are connected using a ring routing. In a single domain design, the VDD pins of the pads are also connected together in a ring routing. But given that each domain has its own VDD wire, the internal connection of the VDD pins of the pads must be cut at the side of each domain. The tech library provides some cells for doing so, the IO filler cut cells; more specifically, the VDD IO filler cut cells have been used.

In the chip we have 11 power supply domains: one for the AES-block, 6 for the iDDPL core of the SERPENT-block and 3 for the SABL core of the SERPENT-block. In the pad ring design, we have chosen to place the 6 power supply pads of the iDDPL core in the bottom side of the pad ring, whereas the 3 power supply pads of the SABL are in the right. The remaining two sections of the pad ring are dedicated for the other pads, which comprise the IO pads of the SERPENT-block and all the pads of the AES-block. Overall there are 12 sections in the pad ring, and the VDD filler cut cells are inserted at the sides of each section through the command *addIoFiller*, specifying the side and the area where these cut must be inserted. The results of this operation is shown in Fig. 4.35(b), where the 12 pad ring sections can be seen as well as the filler cut cells between the power supply pads of the iDDPL and SABL cores.

Finally, the design of the pad ring has been completed by inserting the IO filler cells. The insertion of the filler cells must always done starting from the largest, in this case the IO filler 16 ($3.2\mu m$) down to the smallest, in this case the IO filler 1 ($0.2\mu m$). The floorplan with the complete pad ring is shown in Fig. 4.35(c).

Insertion of the power ring of the cores

The chip is designed in order that only one core at a time runs. Post layout simulations of the SERPENT-block and post-synthesis simulations of the AES-block showed that the average current is limited (in the order of mA in a clock cycle) and the power dissipation is not a constraint at the expected working frequency (maximum some tenths of MHz). For this reason, the internal power ring of the core has been designed using metal wires with a standard width of $3\mu m$ and distance $2\mu m$, which according to the technology manual can sustain even higher currents. As a good rule of thumb, two couples of VDD-GND power rings have been inserted at metal layers M6 and M7. These options are set by the command *addRing*, in which the top metal layers are also specified. Note that in our design, the odd metal layers numbers are used for horizontal metal wires, and the even layers numbers for the vertical wires.

The floorplan after the insertion of the power rings of the cores is shown in Fig. 4.35(d).

Insertion of the power ring of the macros

Each digital module must have a dedicated power ring. For this reason, the command *addRing* has been used also for the insertion of the power ring of a macro-block, after having selected it with the command *selectInst*. In this case only a couple VDD-GND has been inserted at metal layer M6 and M7, with the same distance and the same width of the core power rings.

The floorplan after the insertion of the power rings of the macro-blocks is shown in Fig. 4.35(e).

Insertion of the power stripes

The power stripes connect the power rings and distribute the power supply to the internal logic gates. They can be inserted with the command *addStripe*. Couples of VDD-GND stripes have been set at specific distances ($100\mu\text{m}$) and with an offset from the pad ring in order to have a uniform design. Furthermore, they were cut and connected to the power rings of the macros for avoiding to be routed over the macros with the commands *block_ring* and *break_stripes*.

The command *addStripe* is repeated two times for inserting horizontal and vertical stripes. The floorplan after the insertion of the power ring stripes is shown in Fig. 4.35(f)

Insertion of welltaps and endcaps

The welltap and the encap cells are fundamental and must be placed before the logic cells. The welltap cells are used for connecting the substrate and the n-wells to the VDD and the GND wires respectively, in order to prevent latchup or other electrical issues. The technology library provides the minimum distance at which these connections must be guaranteed; in our design we decided to insert one welltap at each $28.8\mu\text{m}$ using a checkerboard distribution with the command *addWellTap*, followed by the name of the cell in the library, the distance, the offset and the distribution type.

Instead the endcaps are usually placed at the end of rows to handle end-of-row well tie-off requirements and are inserted with the command *addEndCap*, where it must be specified the name of the cells at the beginning and at the end of a row, as well as the location (if only at the boundary of the core or also of the macros).

Furthermore, for avoiding that the processor places welltap and endcap cells too much near to the boundaries of the macros, a blockage area around the macro-blocks has been manually drawn. The blockage area is a chip area where the place and route is interdicted and it avoids that two standard-cells are inserted too much near to the active area of a block.

The floorplan after the insertion of the welltaps and endcaps is shown in Fig. 4.35(g).

Routing the global nets

The global nets are a group of signals that are not defined as ports in the VHDL synthesis file and are not connected to any other signals. The mapping of these

signals and their association to the pins in the design must be specified in the setting file. However, the *NanoRoute* processor doesn't route these signals, which must be instead routed using the *SpecialRoute* processor.

The group of global signals comprises the power supply nets (VDD, GND, VDDE, GNDE) and other special pins of the pads and the compensation cells. The command used for doing the special routing is *sroute*; this command requires to specify the type of pin to be connected, and the top and the bottom layers allowable for the metal interconnections, as well as other options that can be set in the GUI. The command has been launched in this order: first the pins of the cores and the floating stripes, such as the pins of the endcaps and the welltaps, are connected, and this generates the internal M1 stripes where the standard-cells will be inserted; then, the pins of the blocks and the pins of the pads, which refer to the power supply nets, are routed; finally, the pad ring pins, which refer to the special signals of the pads in the pad ring, are connected. The floorplan after the special routing of the global nets is shown in fig. 4.35(h).

The special route procedure has left the power supply nets of the iDDPL and the SABL cores unconnected. For completing the routing of the global nets, we have used the *editWire* option provided by *sroute*, and manually drawn the interconnect wires. In the script, this step has been omitted because it has been done directly in the GUI of Encounter.

Finally, we have defined the pins constraints through the command *setAllowed-PinLayerOnEdge*. After having done some manual corrections and executed the geometry, the connectivity and the antenna checks, which resulted clean, the design has been saved. The layout of the final floorplan is shown in Fig. 4.36.

4.5.5 Static Timing Analysis

Static timing analysis (STA) is crucial for the design of a digital circuit. Basically it is a method of computing the expected timing of a digital circuit in a fast and efficient way, without requiring simulations. The difference with the *Dynamic Timing Simulation (DTS)* is that in the latter it must be specified the input signals of the digital block under analysis, and this can be a time consuming step; moreover STA allows to rapidly detect the critical paths of a circuit, not always allowed by DTS.

The STA is done before and after each design step during the digital back-end flow, because even a slight change of the layout can heavily modify the timing of the entire system. It is used for detecting and possibly fixing timing violations on every paths of the circuits, and can also detect other problems like glitches, slow paths or clock skew.

In synchronized circuits, the parameters of interest for the STA are [101]:

- The **setup time**: it is the minimum amount of time during which a signal should be held steady before the clock event, in order that the datum is reliably sampled by the clock. A violation of the setup time occurs when the signal arrives too late, and misses the time when it should advance.
- The **hold time**: it is the minimum amount of time during which a signal should be held steady after the clock event, in order that the datum is reliably

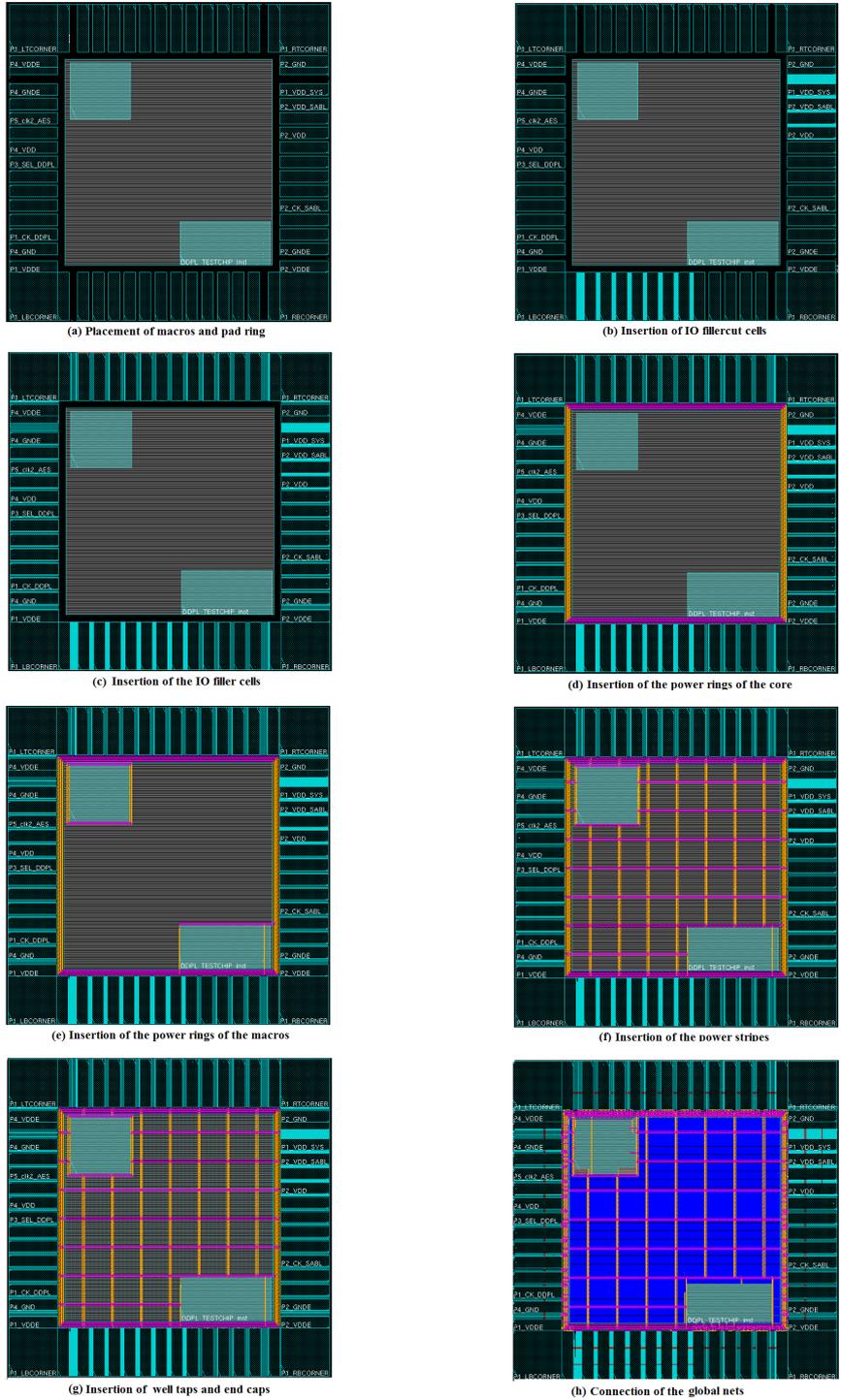


Figure 4.35. Step-by-step design of the floorplan.

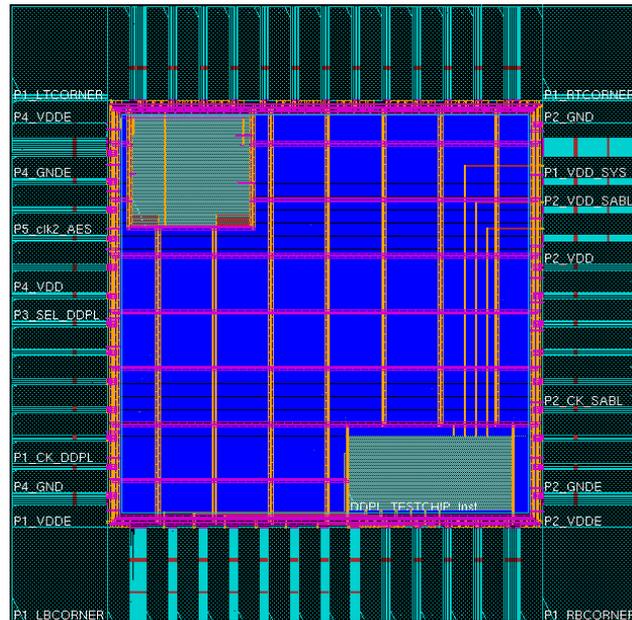


Figure 4.36. Final floorplan of SERPAES.

sampled by the clock. A violation of the hold time occurs when an input signal changes too soon after the clock's active transition.

The setup time depends on the clock period of the circuit. A setup time violation could be fixed for example by reducing the clock frequency. The hold time doesn't depend on the clock, and in presence of hold time violations the designer could be forced to change the design. For this reason fixing hold time violations is usually more time consuming than fixing setup time violations.

In order to correctly understand the results of STA, some basic definitions are recalled [117].

- The *critical path* of a circuit is the path which generates the maximum end-to-end propagation delay path of a signal; the critical path limits the clock frequency of a circuit.
- The *arrival time* of a signal is the time elapsed for a signal to arrive at a certain point; the reference, or time 0.0, is often taken as the arrival time of the clock; the arrival time associated to a path is calculated as the sum of all the delay introduced by the digital components in the path.
- The *required time* is the latest time at which a signal can arrive at a specific clock period without forcing the designer to reduce the clock frequency. STA calculates the required time proceeding in this way: first, at each primary output the required times for rise/fall are set according to the specifications provided to the circuit; then, a backward topological traversal is carried out, processing each gate when the required times at all of its fanouts are known.
- The *slack* associated to each connection is defined as the difference between the required time and the arrival time. A *positive slack* s at a node implies

that the arrival time at that node may be increased by s without affecting the overall delay of the circuit. Conversely, a *negative slack* implies that a path is too slow, and the path must be sped up (or the reference signal delayed) if the whole circuit is to work at the desired speed; the critical paths are characterized by a negative slack.

The first step of STA has been done at a pre-layout level, after the synthesis. In the netlist generated by DC an estimation of the delays of the logic gates and of the interconnections have been done and a back-annotated netlist with all the information on the delays was stored in the *.sdf* file, which is used by the place and route processor for the pre-placement.

Post-layout STA in Encounter is done directly taking into account the real delays of the interconnections in the layout. Furthermore, before the insertion of the clock tree only the setup time violations are calculated because, as previously said, they don't depend on the clock frequency; after the synthesis of the clock tree, also the hold time violations are verified, and at this step a first realistic evaluation on the timing constraints of the design is available.

The script for doing STA is reported in the following:

```
setAnalysisMode -cpr both
timeDesign -{backend_step} -expandedViews -expandReg2Reg -pathReports -drvRep
orts -slackReports -numPaths 50 -prefix ${designName} -outDir "./timingReports"
timeDesign -{backend_step} -hold -expandedViews -expandReg2Reg -pathReports
-slackReports -numPaths 50 -prefix ${designName} -outDir "./timingReports"
```

The command *setAnalysisMode* allows to set some general options for the analysis. The attribute *cpr* stands for *Clock Path Pessimism Removal (CPPR)* or *Clock Reconvergence Pessimism Removal (CRPR)*. Doing an extra pessimistic timing analysis not only requires more time to calculate the delays of the critical paths and fix any violation, but could negatively impact other important parameters such as power and area. In the worst case, it might leave no option but to reduce the functional frequency of the design. For this reason we prudently decided to remove undue pessimism from timing analysis. CPPR is the process of identifying and removing the pessimism introduced in the slack reports for clock paths when the clock paths have a segment in common. It would be also possible to execute a timing analysis with process variations, for instance using the option *-analysisType onChipVariation*, but we omitted this option.

The command *timeDesign* executes the timing analysis by default in the setup mode, whereas in the second line it executes the timing analysis in the hold mode through the option *-hold*. The *timeDesign* command encapsulates several options to run extraction, delay calculation and timing analysis. It generates several types of timing reports based on the specified options. According to the state of the design in which STA is executed, the most common options are: *-prePlace* | *-postPlace* | *-preCTS* | *-postCTS* | *-postRoute [-si]* | *-signoff [-si]* | *-reportOnly [-si]*.

STA checks the timing specifications on all the logic paths of the design. There are 4 types of data path on which the STA executes the calculation, which differ according to the start and the endpoint of the path:

- Register (flip-flop) to register (flip-flop) (*reg-to-reg*): the startpoint is the clock

pin of a sequential element and the endpoint is the data input of the following sequential element.

- Input pin/port to register (flip-flop) (*in-to-reg*): the startpoint is an input port and the endpoint is the data input of the following sequential element.
- Register (flip-flop) to output pin/port (*reg-to-out*): the startpoint is the data input of a sequential element and the endpoint is an output port.
- Input pin/port to output port (*in-to-out*): the startpoint is an input port and the endpoint is an output port (note that in this path there are no registers).

Furthermore, the STA calculates the timing specifications also on the *gated clock paths* (*clkgate*), which are particular clock paths where some gated elements are present.

It is possible to obtain an aggregate report of the timing information of each path using the option *expandedViews*. All the report files are then saved in a specific folder both for the setup and the hold mode.

4.5.6 Placement

The first step of the place and route is the placement. The command lines of the script for the placement are reported in the following:

```
#PLACEMENT

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/floorplan.enc.dat" ${designName}

#PLACEMENT TIMING ANALYSIS
setAnalysisMode -cppr both
timeDesign -expandedViews -expandReg2Reg -prePlace -numPaths 50 -prefix ${designName}_Owlm -outDir "./timingReports"
timeDesign -expandedViews -expandReg2Reg -prePlace -hold -numPaths 50 -prefix ${designName}_Owlm -outDir "./timingReports"

#SETTINGS FOR THE OPTIMIZATION OF THE PLACEMENT
setOptMode -reset
setOptMode -yieldEffort none -effort high -maxDensity 0.85
setOptMode -setupTargetSlack 0.100
setOptMode -simplifyNetlist false -restruct false -preserveModuleFunction true
setOptMode -fixFanoutLoad true
setOptMode -usefulSkew false
setOptMode -keepPort ports.keep

#SETUP PLACEMENT
setPlaceMode -reset
setPlaceMode -timingdriven true -reorderScan false -maxDensity 0.85 -congEffort high -doRPlace true
setPlaceMode -viaInPin true
setPlaceMode -borderPinAccess true

#PLACE THE DESIGN
placeDesign -prePlaceOpt -inPlaceOpt
```

```

setDrawView place

#SAVE THE DESIGN
saveDesignSafely "./dbs/place.enc"

#POST-PLACEMENT TIMING ANALYSIS
timeDesign -expandedViews -expandReg2Reg -preCTS          -numPaths 50 -prefix ${design
Name}_place -outDir "./timingReports"
timeDesign -expandedViews -expandReg2Reg -preCTS -hold -numPaths 50 -prefix ${design
Name}_place -outDir "./timingReports"

```

The script loads the floorplan and executes a first iteration of timing analysis. Note that at this design step the CTS isn't still inserted, thus clocks signals arrive at each gate at the same time. It would be also possible to define some delay models already during the placement which approximate the real delays, anyway we didn't use this option. The first iteration of STA is useful because allows to have a comparison term for the following steps.

Before launching the placement command, the command *setOptMode* sets some global parameters which are used for optimizing the placement. The option *-yieldEffort* has the purpose of setting the level of yield optimization of the library cells, and it is used if there are low and high yield cells in the tech library; we have disabled this option. The effort of the placement is set to high through the option *-effort high*. The option *-maxDensity* indicates that the placement must optimize the design density up to a maximum 85%. The option *-setupTargetSlack* sets the slack of the setup mode (in the worst case) to $0.1\mu\text{s}$; it represents a constraint for the placement, because when the total slack arrives at this value, the optimization ends. The option *-simplifyNetlist* improves the area overhead checking if the timing optimization simplifies the netlist; with this command the processor recovers area, decreases congestion, and improves the runtime by simplifying the netlist in the following ways: by removing dangling output instances, by propagating constants, by removing unobservable logic, by remapping useless logic. The option *-restruct* controls whether the timing optimization swaps pins and in this case restructures the netlist. Both *-simplifyNetlist* and *-restruct* are disabled during the placement. The option *-preserveModuleFunction* determines whether to preserve logical functions at hierarchical module ports. The option *fixFanoutLoad* forces timing optimization to correct fanout load violations, such as for example maximum load or maximum fanout count violations, depending on the libraries. The option *keepPort* specifies a file that contains the hierarchical instances whose port boundaries cannot be changed. For what concerns the *usefulSkew* option, it enables the insertion of some delays in the clock path with the aim of fixing the setup violations during the timing optimization.

The command *setPlaceMode* sets some global variables that will be used by the placement phase. The timing driven placement option is enabled by the option *-timingDriven true*; timing driven means that the processor keeps into account the timing specifications defined in the *.sdf* file, and therefore tries to place the standard-cells in order to reduce the critical paths. The attribute *-noReorderScan* is linked to an option used in the scan test and is disabled. The attributes *borderPinAccess* and *viaInPin* are linked to the position of the pins of the cells and the routing. The command *congEffort* allows to optimize the design in case of congestion, in this case

the effort is set *high*.

The placement is done by the command *placeDesign*. The option *-prePlaceOpt* and *-inPlaceOpt* enables the pre-place and the in-place optimization flow, respectively. The command *setDrawView* is a control line which sets the design view in the design display area.

Finally the design is saved and the post-placement STA is executed.

4.5.7 Post-placement optimization

A first step of optimization has been done during the placement described in the previous paragraph. Once the cells are placed, Encounter extrapolates electrical parasitics, such as resistance and capacitors, for calculating propagation delays through STA. Then the critical paths of the circuit are identified and the placed design can be optimized, using different strategies (e.g. buffer insertion, moving or re-mapping standard cells, etc). Usually an incremental optimization is executed, in which the logic is optimized by starting from the latest configuration and changing step by step the placement in order to find an optimal result.

The placement optimization is executed using the following command lines:

```
#PLACEMENT OPTIMIZATION

#TIMING OPTIMIZATION OF THE PLACED DESIGN
proc optimizePlace_incr {index}
{
#SETTINGS FOR THE OPTIMIZATION OF THE PLACED DESIGN
setOptMode -reset
setOptMode -yieldEffort none -effort high -maxDensity 0.8
setOptMode -setupTargetSlack 0.100
setOptMode -simplifyNetlist false -restruct false -preserveModuleFunction true
setOptMode -fixFanoutLoad true
setOptMode -usefulSkew true
setOptMode -keepPort ports.keep
setUsefulSkewMode -noBoundary true

#OPTIMIZE DESIGN
getOptMode
optDesign -preCTS -incr
}

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/place.enc.dat" ${designName}

#SETUP TIMING ANALYSIS
setAnalysisMode -cpr both

#SETUP GENERAL OPTIMIZATION OPTIONS
set_interactive_constraint_modes default
source ./scripts/setup/dont_use_cells.tcl
set_interactive_constraint_modes { }

#INCREMENTAL OPTIMIZATION
set optimizations 2
for {set index 1} {${index} <= ${optimizations}} {incr index} {
```

```

optimizePlace_incr ${index}
}

#SAVE THE DESIGN
saveDesignSafely "./dbs/placeopt.enc"

#PRE CTS TIMING ANALYSIS
timeDesign -preCTS          -expandedViews -expandReg2Reg -numPaths 50 -prefix ${design
Name}_placeopt -outDir "./timingReports"
timeDesign -preCTS -hold -expandedViews -expandReg2Reg -numPaths 50 -prefix ${design
Name}_placeopt -outDir "./timingReports"

```

The incremental optimization is done by the *for* cycle, which at each iteration recalls the procedure *optimizePlace_incr*. The procedure *optimizePlace_incr* is defined at the beginning, where the general setting for the timing optimization are declared, as well as the command *optDesign* with the option *-preCTS*, which indicates that the optimization is done on the placed design before the insertion of the CTS. By default, *-preCTS* repairs design rule violations and setup violations. The other timing optimization parameters have been already described in the previous paragraph.

The command *set_interactive_constraint_modes* puts the software into interactive constraint entry mode, and after it the processor is provided with a list of cells, written in the file *dont_use_cells*, that must be excluded by the timing optimization. In the file *dont_use_cells* some too low driving cells are indicated, because the timing specifications would be too much sensitive to the routing and couldn't be calculated. Then the program exits by interactive constraint entry mode by repeating the command *set_interactive_constraint_mode* followed by an empty list of objects.

Finally the design is saved and the post placement optimization STA is executed.

4.5.8 Clock Tree Synthesis (CTS)

The algorithm used by SoC Encounter for the synthesis of the clock tree is the following: first, the core is divided in several rooms, the so called *clock domains*; then, the processor starts to trace the clock tree and each branch of the tree feeds one room with a almost zero skew; afterwards, each clock domain is divided in sub-blocks and the algorithm is repeated until the clock domain corresponds to a single standard-cell. The clock tree is then inserted using a distributed buffers synthesis.

The command lines used for synthesizing the clock tree are the following:

```

#SYNTHESIS OF THE CLOCK TREE

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/placeopt.enc.dat" ${designName}

#SETUP TIMING ANALYSIS
setAnalysisMode -cpr both

#CLOCK TREE INSERTION
cleanupSpecifyClockTree
specifyClockTree -file "./scripts/clock/SERPAES_clocks.ctstch"

```

```

#Leaf pin for create_clock P6_clk2_AES/ZI
GlobalLeafPin
+ P5_clk2_AES/IO
LeafPinGroup
+ P5_clk2_AES/IO
GlobalLeafPin
+ P5_clk2_AES/ENZI
LeafPinGroup
+ P5_clk2_AES/ENZI
#-----
# Clock Root : P5_clk2_AES/ZI
# Clock Name : clk2_AES
# Clock Period : 5ns
# Clock Name : clk2_AES
# Clock Period : 5ns
# Clock Name : clk2_AES
# Clock Period : 5ns
# Clock Name : clk2_AES
# Clock Period : 5ns
#-----
AutoCTSRootPin P5_clk2_AES/ZI
Period 5ns
MaxDelay 1ns # sdc driven default
MinDelay 0ns # sdc driven default
MaxSkew 10ps # sdc driven default
SinkMaxTran 150ps # sdc driven default
BufMaxTran 150ps # sdc driven default
Buffer H865_L8_CNBFX10 H865_L8_CNBFX103 H865_L8_CNBFX124 H865_L8_CNBFX14
H865_L8_CNBFX17 H865_L8_CNBFX21 H865_L8_CNBFX24 H865_L8_CNBFX27 H865_L8_CNBFX31
H865_L8_CNBFX34 H865_L8_CNBFX38 H865_L8_CNBFX38_0 H865_L8_CNBFX38_1
H865_L8_CNBFX38_10 H865_L8_CNBFX38_11 H865_L8_CNBFX38_12 H865_L8_CNBFX38_13
H865_L8_CNBFX38_14 H865_L8_CNBFX38_15 H865_L8_CNBFX38_16 H865_L8_CNBFX38_17
H865_L8_CNBFX38_18 H865_L8_CNBFX38_19 H865_L8_CNBFX38_2 H865_L8_CNBFX38_20
H865_L8_CNBFX38_21 H865_L8_CNBFX38_22 H865_L8_CNBFX38_23 H865_L8_CNBFX38_3
H865_L8_CNBFX38_4 H865_L8_CNBFX38_5 H865_L8_CNBFX38_6 H865_L8_CNBFX38_7
H865_L8_CNBFX38_8 H865_L8_CNBFX38_9 H865_L8_CNBFX41 H865_L8_CNBFX45 H865_L8_CNBFX48
H865_L8_CNBFX52 H865_L8_CNBFX55 H865_L8_CNBFX58
NoGating NO
DetailReport YES
RouteClkNet YES
PostOpt YES
OptAddBuffer YES
END

```

Figure 4.37. Description of the clock *clk2_AES* in the file *SERPAES_clock.ctstch*

```

setCTSMODE -multiCorner true
ckSynthesis -forceReconvergent -report ./cts/${designName}_clocks.ctrpt -rguide ./c
ts/${designName}.rguide

#SAVE THE DESIGN
saveDesignSafely "./dbs/cts.enc"

#POST CTS TIMING ANALYSIS
timeDesign -postCTS -expandedViews -expandReg2Reg -pathReports -drvReports -sl
ackReports -numPaths 50 -prefix ${designName}_cts -outDir "./timingReports"
timeDesign -postCTS -hold -expandedViews -expandReg2Reg -pathReports -sl
ackReports -numPaths 50 -prefix ${designName}_cts -outDir "./timingReports"

```

The command *specifyClockTree* is initialized by the command *cleanupSpecifyClockTree*. The command *specifyClockTree* indicates the file *SERPAES_clocks.ctstch* where the timing specifications for each clock signal are listed (e.g. max and min delay, max skew, the number and type of buffer of the tech library associated to the path, etc). A screenshot of the file is shown in Fig. 4.37 for the clock signal *clk2_AES*.

With reference to Fig. 4.37, the following parameters are specified in the file: the name of the pin associated to the clock pad (*ClkGroup*); the period of the clock (*Clock Period*, taken from the *.sdc* file generated by the synthesis); the pin from which the clock is generated (*AutoCTSRootPin*); the maximum and minimum delay (*MaxDelay* and *MinDelay*); the maximum allowable skew (*MaxSkew*); the timing constraints for the maximum transition from the input for clock pin (*SinkMaxTran*) and the buffers (*BufMaxTran*); the names of the buffers of the technology library that are used in the CTS (*Buffer*); an option that allows to stop or continue the CTS for a clock gating logic (*NoGating*); an option that allows to generate a report (*DetailReport*); an option for allowing the routing of the clock tree using the proprietary algorithm NanoRoute (*RouteClkNet*); an option for doing the post-CTS

optimization (*PostOpt*); an option for enabling the insertion of buffers during the optimization (*OptAddBuffer*).

The command *setCTSmode* allows to enable or disable some options related to the clock tree synthesis. The option *-multiCorner* sets the multi-corner mode. The synthesis is done by the command *ckSynthesis*. The option *-forceReconvergent* forces CTS to synthesize a clock with self-reconvergence or clocks with crossover points; without this option, CTS interrupts and issues errors.

Finally the design is saved and the post-CTS STA is executed.

4.5.9 Post-CTS optimization

After the insertion of the clock tree, the timing constraints of the design must be verified and possibly optimized. The post-CTS optimization is executed with the following scripts:

```
#CTS OPTIMIZATION

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/cts.enc.dat" ${designName}

#SETUP TIMING ANALYSIS
setAnalysisMode -cpr both

#SETUP TIMING OPTIMIZATION
set_interactive_constraint_modes default
source ./scripts/setup/dont_use_cells.tcl
set_interactive_constraint_modes { }
setOptMode -yieldEffort none
setOptMode -effort high -maxDensity 0.95
setOptMode -holdTargetSlack 0.1 -setupTargetSlack 0.0
setOptMode -simplifyNetlist true -restruct true -preserveModuleFunction true
setOptMode -usefulSkew true
setOptMode -keepPort ports.keep

#SETUP USEFULSKEW
setUsefulSkewMode -noBoundary true -useCells {buffer_list}

#POST-CTS OPTIMIZATION
optDesign -postCTS
optDesign -postCTS -hold

#SAVE THE DESIGN
saveDesignSafely "./dbs/ctsopt.enc"

#POST-CTS TIMING ANALYSIS
timeDesign -postCTS -expandedViews -expandReg2Reg -pathReports -drvReports -sl
ackReports -numPaths 50 -prefix ${designName}_ctsopt -outDir "./timingReports"
timeDesign -postCTS -hold -expandedViews -expandReg2Reg -pathReports -sl
ackReports -numPaths 50 -prefix ${designName}_ctsopt -outDir "./timingReports"
```

The timing optimization parameters of the CTS optimization script are slightly different than those used for the placement and the post-placement. At this point, the density is enhanced up to 95% of the design, in order to take into account also the presence of the clock tree. Then, the maximum allowable slack for the hold time

mode is also set. Finally, the *-simplifyNetlist*, *-restruct*, and *-usefulSkew* options are set to true.

As above described, the *-usefulSkew* option enables the insertion of some delays in the clock path with the aim of generating skews and fixing the timing violations. A list of buffers to be inserted for generating the skews are specified by the command *setUsefulSkewMode* where the *-useCells* option allows to indicate a list of buffers (*buffer_list*) in the tech library to be inserted in the path.

The *optDesign* command, with specified the *-postCTS* options, optimizes the design by fixing setup (by default) and hold time violations (by the insertion of the attribute *-hold*).

Finally the design is saved and the post-CTS timing analysis is executed.

4.5.10 Routing

The routing is the final step of the back-end flow, which is usually executed in several iterations. The processor which routes the design is *NanoRoute*. During the routing, the processor connects the pins of the standard cells using the metal layers available in the tech library, starting from the low metal layer, and fixes any geometry and interconnection errors.

The script for the routing is reported in the following.

```
#ROUTING

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/ctsopt.enc.dat" ${designName}

#SETUP TIMING ANALYSIS
setAnalysisMode -cpr both

#INSERTION OF DECOUPLING CAPACITANCES
addDeCapCellCandidates {decap_list}
addDeCap -totCap 1000

#INSERTION OF THE FILLER CELLS
addFiller -prefix FILLER_large -cell {filler_list}

#SETTING THE NANOROUTE PARAMETERS
setNanoRouteMode -routeDesignRouteClockNetsFirst true
setNanoRouteMode -routeBottomRoutingLayer 1
setNanoRouteMode -routeTopRoutingLayer 7
setNanoRouteMode -routeAntennaCellName HS65_LS_ANTPROT1 -routeInsertAntennaDiode true
setNanoRouteMode -drouteAutoStop false
setNanoRouteMode -envHonorTrack true
setNanoRouteMode -drouteExpNumCutsBlockPin {( 2 * )}
setNanoRouteMode -routeWithViaInPin true
setNanoRouteMode -routeWithViaOnlyForStandardCellPin false
setNanoRouteMode -routeMergeSpecialWire true
setNanoRouteMode -drouteTaperDistLimit 0
setNanoRouteMode -drouteUseMinSpacingForBlockage false
setNanoRouteMode -drouteHonorStubRuleForBlockPin true

#ROUTING STANDARD WIRES
routeDesign -globalDetail
```

```

#FIX GEOMETRY VIOLATIONS
editDeleteViolations
routeDesign -globalDetail

#SAVE THE DESIGN
saveDesignSafely "./dbs/route.enc"

# Analyze setup/max
timeDesign -postRoute          -expandedViews -expandReg2Reg -pathReports -drvReports -sl
ackReports -numPaths 50 -prefix ${designName}_route -outDir "./timingReports"
timeDesign -postRoute -hold -expandedViews -expandReg2Reg -pathReports          -sl
ackReports -numPaths 50 -prefix ${designName}_route -outDir "./timingReports"

```

Before doing the routing, the decoupling capacitor cells and the filler cells are inserted. The decoupling capacitors have the purpose of reducing the high frequency noise on the VDD wires; in the digital design flow an amount of capacitance is inserted already at layout level. This is done by the command *addDeCap*, where the total amount of capacitance can be specified, and the command *addDeCapCellCandidates*, which requires to specify a list of decoupling cells from the tech library. The role of filler cells is also important, because they fill up the gaps in between the standard-cells in order to have continuity in the design. This is done by the command *addFiller*.

The command *setNanoRouteMode* allows to set some parameters for the routing. The option *-routeDesignRouteClockNetsFirst* checks whether the existing clock nets traced during the CTS were fully or partially routed with the NanoRoute router, and possibly reroutes them before routing the remaining nets. The options *routeBottomRoutingLayer* and *routeTopRoutingLayer* have the purpose of specifying the lowest and the highest layers the NanoRoute router uses for routing. The option *routeAntennaCellName* specifies antenna diode cells from the tech library to use during post-route optimization.

The option *-envHonorTrack* prevents NanoRoute to re-generate tracks. The option *-drouteExpNumCutsBlockPin (2 *)* has the purpose of avoiding mincuts around macros. The option *routeWithViaInPin* forces the router to enclose via geometries completely inside standard cell pins. The option *-routeWithViaOnlyForStandardCellPin* forces the router to use vias or planar access for standard cell pins, but we have disabled this option. The option *-routeMergeSpecialWire* merges overlapping same-net special wires in order to create large rectangles when calculating spacing requirements. The option *-drouteTaperDistLimit* specifies then maximum allowable distance for non-default rule tapering, whereas *drouteUseMinSpacingForBlockage* and *-drouteHonorStubRuleForBlockPin* are other two options of the NanoRoute which are disabled and enabled respectively.

The command *routeDesign* executes both global and detailed routing (option *-globalDetail* of the standard wires. As final step, the processor deletes signal nets with violation markers after thr routing through the option *editDeleteViolations*, then another step of routing is done in order to connect any floating net.

Finally the design is saved and the post-route timing analysis is executed.

4.5.11 Post-route optimization

```
#POST-ROUTE OPTIMIZATION
```

```

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/route.enc.dat" ${designName}

#SETUP TIMING ANALYSIS
setAnalysisMode -cpr both
set_interactive_constraint_modes default
set_clock_propagation "propagated"
set_propagated_clock [all_clocks]
set_interactive_constraint_modes { }

#SETUP OPTIMIZATION
set_interactive_constraint_modes default
source ./scripts/setup/dont_use_cells.tcl
set_interactive_constraint_modes { }

setOptMode -yieldEffort none
setOptMode -effort high -maxDensity 0.95
setOptMode -drcMargin 0.10
setOptMode -holdTargetSlack 0.1 -setupTargetSlack 0.0
setOptMode -simplifyNetlist false -restruct false -preserveModuleFunction true
setOptMode -usefulSkew true
setOptMode -keepPort ports.keep

#SETUP USEFULSKEW
setUsefulSkewMode -noBoundary true -useCells {buffer_list}

#REMOVE FILLER CELLS
deleteFiller -prefix FILLER_large -cell {filler_list}

#OPTIMIZATION OF THE DESIGN
optDesign -postRoute
optDesign -postRoute -drv
optDesign -postRoute -si
optDesign -postRoute -hold

#FILLER-CELLS INSERTION
addFiller -prefix FILLER_large -cell {filler_list}

#SAVE THE DESIGN
saveDesignSafely "./dbs/routeopt.enc"
saveDesignSafely "./dbs/final.enc"

#POST ROUTING OPTIMIZATION TIMING ANALYSIS
timeDesign -postRoute -expandedViews -expandReg2Reg -pathReports -drvReports
-slackReports -numPaths 50 -prefix ${designName}_routeopt -outDir "./timingReports"
timeDesign -postRoute -hold -expandedViews -expandReg2Reg -pathReports
-slackReports -numPaths 50 -prefix ${designName}_routeopt -outDir "./timingReports"

```

In the timing analysis setup, the command *set_propagated_clock* with the option *all_clocks* is inserted. It puts the *propagated_clock* assertion on all the clock waveform objects. This will cause all clock endpoints in the fanout of the specified object to receive propagated clock timing.

The first phase is the removal of the filler cells of the design with the command *deleteFiller*, which must precede the post route timing optimizations. The optimiza-

tions are then executed for the setup and the hold mode, together with the design rule verification (option *-drc*). After the timing optimizations, the filler cells are re-inserted.

Finally the design is saved and the post optimization route timing analysis is executed.

4.5.12 Signoff and final verification

After the post-route optimization, the design is complete. The last script executes the signoff of the flow together with a final iteration of timing analysis. At this step physical and geometry verifications are also executed, and the output files are saved. The signoff script is reported beyond.

```
#SIGNOFF

#GENERAL SETTINGS
setupEnvironment
restoreDesign "./dbs/final.enc.dat" ${designName}

#SETUP TIMING ANALYSIS
setAnalysisMode -cpr both
set_interactive_constraint_modes default
set_clock_propagation "propagated"
set_propagated_clock [all_clocks]
set_interactive_constraint_modes { }

#PHYSICAL CHECKS
verifyGeometry -report ./reports/${designName}.verify.rpt
verifyConnectivity -type all -error 1000 -warning 50 -report ./reports/${designName}.
connectivity.rpt
verifyMetalDensity -detailed -layers {1 2 3 4 5 6 7} -report ./reports/${designName}.
density.rpt
verifyProcessAntenna -reportfile ./reports/${designName}.antenna.rpt -leffile ./model
s/${designName}.antenna.lef -error 1000
clearDrc

#SIGNOFF TIMING ANALYSIS
timeDesign -postroute -expandedViews -expandReg2Reg -pathReports -drvReports -sla
ckReports -numPaths 50 -prefix ${designName}_signoff -outDir "./timingReports"
timeDesign -postroute -hold -expandedViews -expandReg2Reg -pathReports -sla
ckReports -numPaths 50 -prefix ${designName}_signoff -outDir "./timingReports"

#CLOCK REPORTS
setSchedulingFile ./cts/scheduling_file.cts
specifyClockTree -file "./scripts/clock/SERPAES_clocks.ctstch"
setCTSMode -multiCorner true
reportClockTree -postRoute -report ./cts/${designName}_clocks_signoff.ctrpt -macromo
del ./models/${designName}.clockModel

#SETTING DELAY CALCULATION MODE
setDelayCalMode -signoff true

#WRITING SDF OUTPUT FILES
write_sdf -version 3.0 -precision 4 -process worst::worst -voltage 1.15::1.15 -temper
ature 125::125 -view slow_1.15V_125C ./models/${designName}_ss_1.15V_125C.sdf
write_sdf -version 3.0 -precision 4 -process worst::worst -voltage 1.10::1.10 -temper
```

```

ature -40::-40 -view slow_1.10V_m40C ./models/${designName}_ss_1.10V_m40C.sdf
write_sdf -version 3.0 -precision 4 -process best::best -voltage 1.25::1.25 -temper
ature -40::-40 -view fast_1.25V_m40C ./models/${designName}_ff_1.25V_m40C.sdf
write_sdf -version 3.0 -precision 4 -process best::best -voltage 1.25::1.25 -temper
ature 125::125 -view fast_1.25V_125C ./models/${designName}_ff_1.25V_125C.sdf

#WRITING LIB OUTPUT FILES
set_default_view -setup slow_1.10V_m40C -hold fast_1.25V_125C
setAnalysisMode -checkType setup -asyncChecks async -skew true
do_extract_model ./models/${designName}_slow_1.10V_m40C.lib
setAnalysisMode -checkType hold -asyncChecks async -skew true
do_extract_model ./models/${designName}_fast_1.25V_125C.lib
set_default_view -setup slow_1.15V_125C -hold fast_1.25V_m40C
setAnalysisMode -checkType setup -asyncChecks async -skew true
do_extract_model ./models/${designName}_slow_1.15V_125C.lib
setAnalysisMode -checkType hold -asyncChecks async -skew true
do_extract_model ./models/${designName}_fast_1.25V_m40C.lib

#WRITING DEF OUTPUT FILE
defOut -floorplan -placement -cutRow -netlist -routing ./dbs/final.enc.dat/${design
Name}.def

#WRITING GDS OUTPUT FILE
streamOut ./models/${designName}.gds -mode ALL \
        -mapFile /des_local/virtuoso/cdsoa/tech/mapOut

#WRITING VERILOG OUTPUT FILE
saveNetlist ./models/${designName}.v -includePowerGround

#WRITING LEF OUTPUT FILE
lefOut -5.6 ./models/${designName}.lef

```

After having defined the general settings, the formal checks are executed: *verifyGeometry*, *verifyConnectivity*, *verifyMetalDensity*, and *verifyProcessAntenna*. In the report files no errors are detected. The timing reports are also clean for each signal path.

The very final step in SoC Encounter is saving the output files: the entire clock tree timing report on each path (with the command *reportClockTree*), the *.sdf* files for all the corners (with the command *write_sdf*), the *.lib* files for all the corners (with the command *do_extract_model*), the *.def* file (with the command *defOut*), the *.gds* file (with the command *streamOut*, where also the technology mapping is specified), the Verilog *.v* file (with the command *saveNetlist*), and the *.lef* file (with the command *lefOut*). The command *setDelayCalcMode* sets the global parameters for delay calculation for the signoff. The complete layout of the chip is shown in Fig. 4.38.

The finishing phase of the design has been executed in Cadence Virtuoso, where the *SERPAES.gds* file has been exported from SoC Encounter. This phase requires the insertion of the sealring to insulate the die area. Final steps to re-finish the layout, such as fillering, pad substitution (with pad pitch of $50\mu\text{m}$), and tiling have been executed by the foundry.

During the export process Virtuoso requires a map file where all the rules for geometry, pins, and layers are described. Indeed, SoC Encounter and Virtuoso have different map rules, thus in order that each physical element in the layout generated

by SoC Encounter and described in the binary file *.gds* is correctly converted in Virtuoso Layout Editor, this map file can be found in the technology library to complete the stream-in procedure.

The final layout of the chip is shown in Fig. 4.39. The SERPENT-block, treated as a black box by SoC Encounter, has been now correctly loaded in Virtuoso, where physical (i.e. DRC and LVS), density and electrical checks have been executed on the complete chip, as well as full-chip post-layout simulations which confirm the functionality of each module.

4.6 Testing the functionality of SERPAES

4.6.1 Design of the SERPAES board

The SERPAES chip has been taped out on March 2014. It has been packaged with a Ceramic Quad Flat Package with 64 pins (CQFP64), and the wire bonding of the internal 56 pads has been executed by the foundry. A number of eight prototype packaged specimens of the SERPAES chip have been delivered to our department at the end of 2014. A photograph of the die of one chip with the microscope is shown in Fig. 4.40.

We have designed a specific PCB to host the chip, which has been customized in order to mount PAAs. The board is composed of two layers, and all the integrated components are placed on the top layer. No specific requirements on the width of the metal wires must be satisfied, being the current adsorption and the working frequency rather small. Therefore, we have adopted the standard values of 0.2mm for signal interconnections and 0.35mm for power interconnections. Each power wire has been designed as shorter as possible in order to reduce the inductive load which may create ringing on the current measurements. The design board is shown in Fig. 4.41. The SERPAES chip is visible just in the middle part of the board.

The PCB is provided with dedicated jumpers on the power supply nets for each internal core, in order to guarantee some accessible test-points for a current probe and measure the current adsorption of each sub-part. The board takes a 5V voltage as input, which is set to 1.2V and 1.8V by a number of commercial linear regulators. The power supply circuitry has been carefully designed, being the most critical part for PAAs measurements. Further details about the design of the board are omitted.

4.6.2 First measurements on the chip

First measurements on the board have been performed to test the functionality of the chip, and proved that each core is correctly working. For this purpose, we have installed a setup in our laboratory (Fig. 4.42). The chip has been interfaced to the personal computer through a FPGA board (Altera Cyclone II) and a serial interface to handle the IO data transfer with and to Labview. An inductive current probe has been also arranged on the power test-points of the board for current measures. Furthermore, a VHDL testbench has been developed to program the FPGA.

As first examples, in Fig. 4.43 the power trace measured from the AES-0 core is reported. The unit scale is in volts, because the oscilloscope converts the current in a voltage signal. It will be interesting to compare the intensity of this signal with

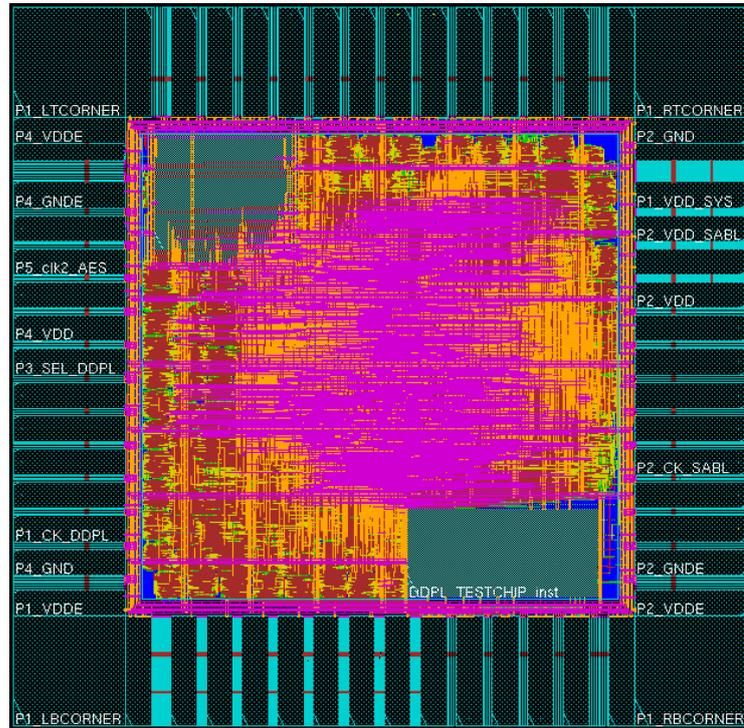


Figure 4.38. Post place and route layout of the SERPAES chip in SoC Encounter.

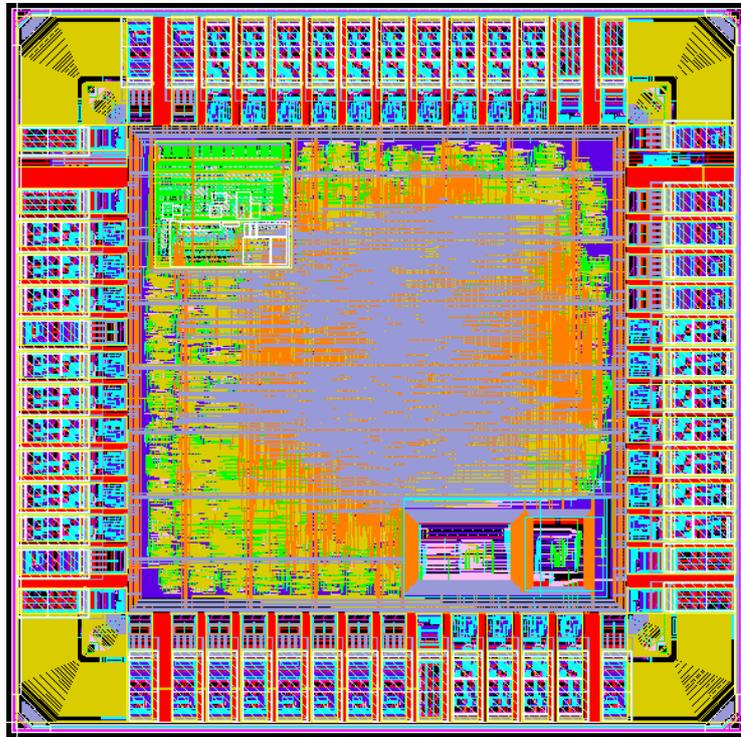


Figure 4.39. Final layout of SERPAES for full-chip verification in Cadence Virtuoso.

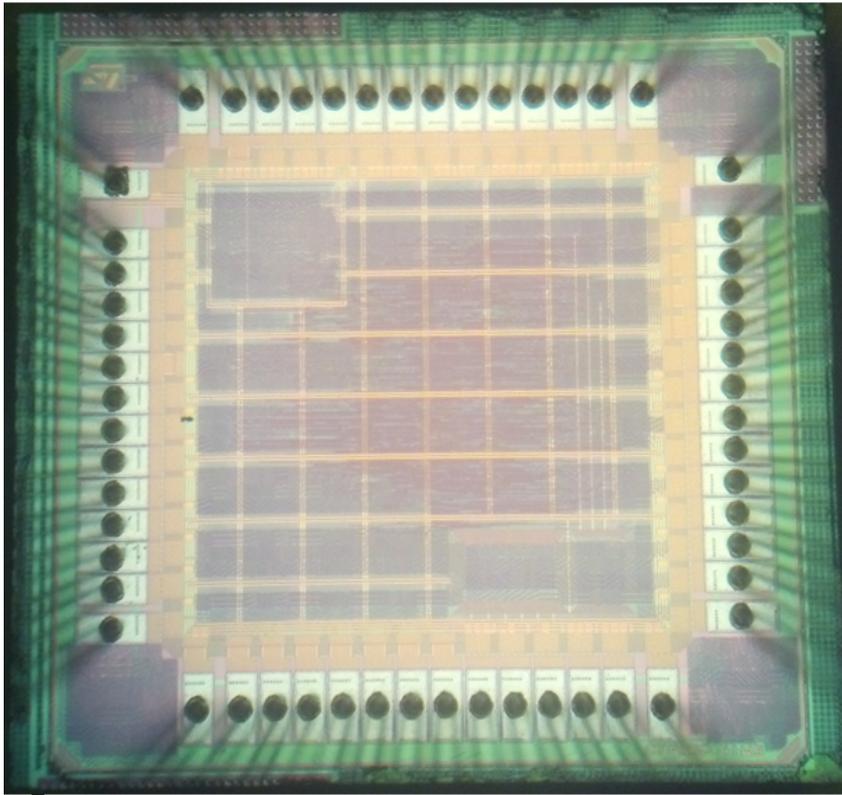


Figure 4.40. Photograph of the SERPAES chip.

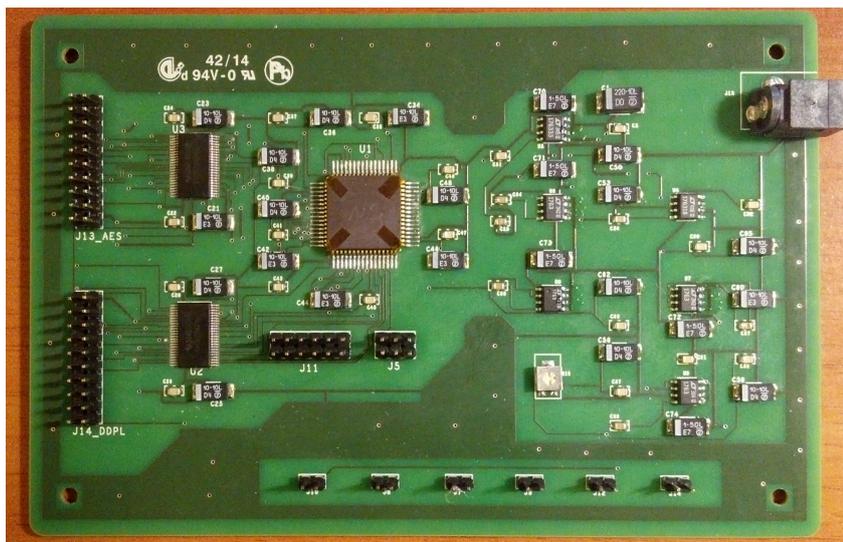


Figure 4.41. Photograph of the board designed to perform measurements on the SERPAES chip.

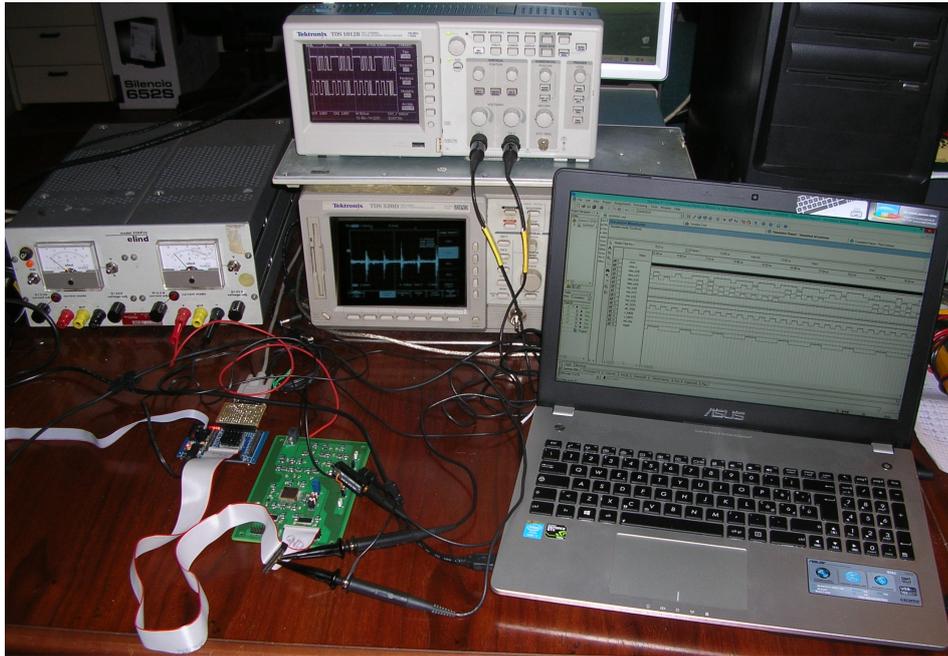


Figure 4.42. Photograph of the workstation in our laboratory for the evaluation of SERPAES.

that measured by the same core implemented on a FPGA (see in next chapter). We have also performed a first PAAs in order to calibrate the setup: the attack result for one byte of the S-Box of the first round is shown in Fig. 4.44, which reports the correlation coefficients for each key for 490k plaintexts; correct key is indicated in green, and highlights a peak as visible near to the sample 1000.

Finally, the full custom cores have been also tested. In Fig. 4.45 the waveforms on the oscilloscope correspond to the predicted signals of the cores iDDPL and SABL in correspondence of the critical path (@5MHz), which confirm the functionality of the manufactured chip. The current traces are depicted in Fig. 4.46 and Fig. 4.47 for three representative clock cycles. Signal power traces are quite noisy and low, thus they will require an adequate post-processing phase in order to extract the exploitable signal fraction for PAAs. Anyway, the balance of iDDPL traces can be deduced already by visual inspection.

4.7 Conclusions

In this chapter we have described in deeper detail all the design steps of the SERPAES chip, which has been also tested with success on functionality. First power measurements have been also executed.

Future activity will be oriented towards the design of a PAAs setup to perform the security evaluation of all the cores in SERPAES, in order to verify if the power model and the strength of the countermeasures are confirmed also through an on silico evaluation.

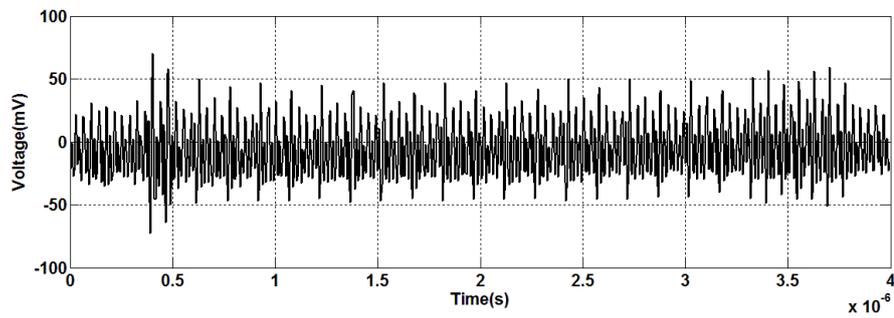


Figure 4.43. Power trace from the core AES0 inside the SERPAES chip measured by the scope.

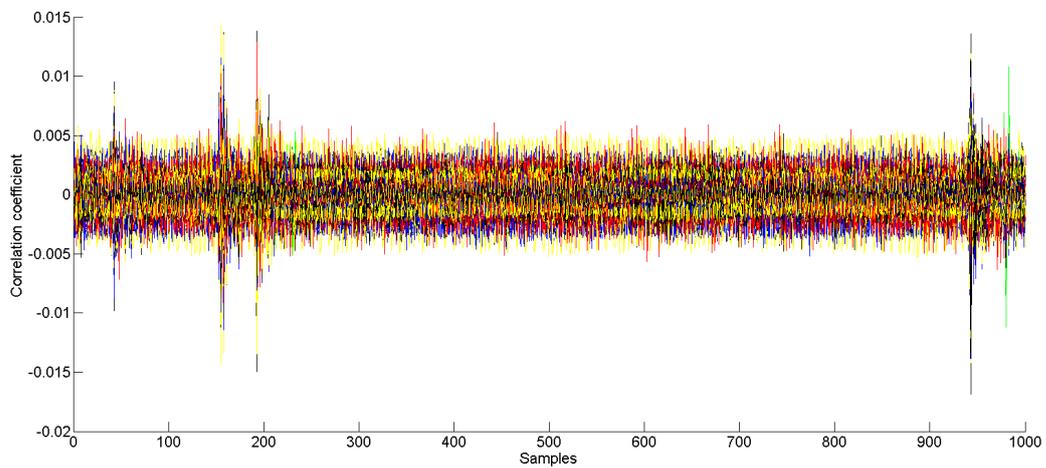


Figure 4.44. Correlation coefficient curves for one byte (15) of the word at the output of the SubstitutionBox layer of round 1 of AES-0 (490k plaintexts).

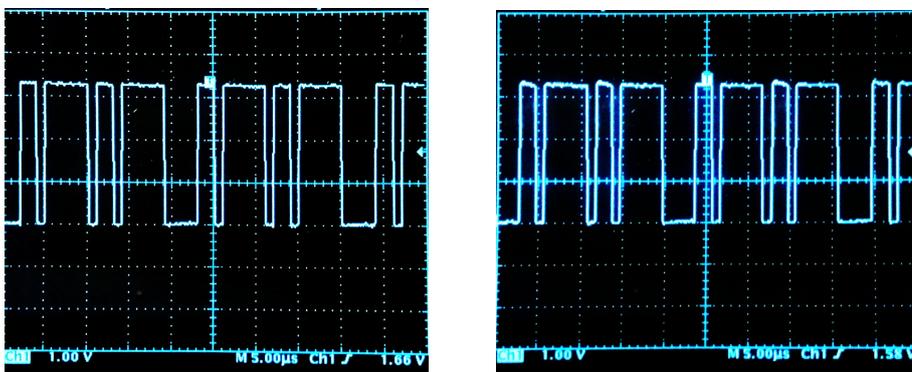


Figure 4.45. Output bit on the critical path of the iDDPL (left) and the SABL (right) blocks of the SERPAES chip.

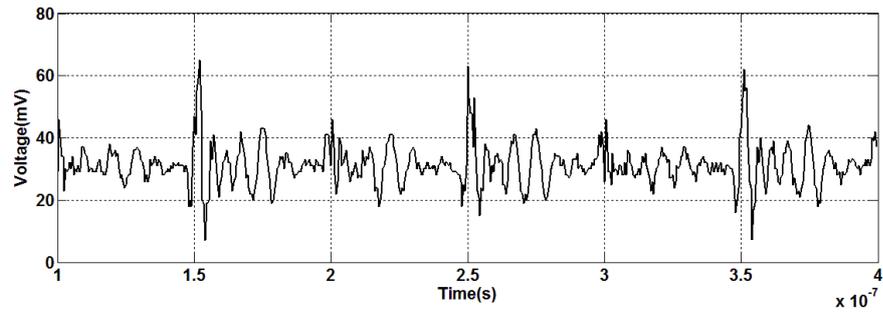


Figure 4.46. Power trace from the iDDPL block inside the SERPAES chip (three cycles) measured by the scope.

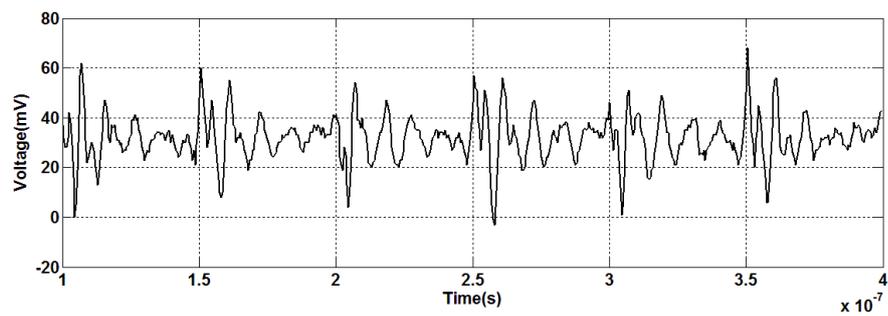


Figure 4.47. Power trace from the SABL block inside the SERPAES chip (three cycles) measured by the scope.

Chapter 5

Leakage Power Analysis attacks against nanoscaled DPL circuits

5.1 Introduction

The most part of literature in the field of PAAs is dedicated to the study and the analysis of the leakage arising from the dynamic power consumption. Many circuit level countermeasures have been introduced in order to reduce the side-channel emission of a device by breaking the link between current consumption and data directly at physical level, also in presence of time and capacitive unbalances, as widely described in previous chapters. The difficulty of reducing these mismatches is a stringent constraint in the design of combinational circuits as well as memory elements. However, one of the most interesting and debatable points in physical observable cryptography is represented by the leakage originated by the static power consumption of CMOS devices, already described in Chapter 1. With the technology scaling, static power becomes a major concern in the design of digital circuits, and in the context of SCAs some recently published works demonstrate that cryptographic circuits can be successfully attacked by using only the current samples correspondent to the static power as subset of observable leakage points representative of the instantaneous power consumption, instead of collecting many points during a period of elaboration.

As discussed in previous chapters, transistor level countermeasures are based on the adoption of logic styles whose power consumption is constant or independent of the processed data; this is typically obtained through differential signaling and precharged logic. DPLs were specifically implemented with the aim of de-correlating the dependence of the dynamic power consumption on the logic data transitions by balancing the energy for each clock cycles and data input. In sub-100 nm technologies, leakage power is well known to be comparable to the dynamic power, and is expected to become an even larger portion of the chip power budget in the future [56].

Leakage Power Analysis (LPA) attacks have been introduced for the first time in [6] and extended in [5]. As opposed to DPA/CPA attacks, these recently proposed attacks are not well understood yet, and deserve further investigation to understand their effectiveness in realistic conditions, i.e. in the presence of noise, process vari-

ations and countermeasures to DPA/CPA attacks (which have to be adopted in applications requiring a high level of security). The impact of process parameter variations on LPA attacks effectiveness has been recently analyzed in [7] and [36] referring to standard CMOS logic style, and some preliminary work has been done in [37] for DPL circuits.

LPA attacks are based on a different approach with respect to DPA/CPA attacks: in standard logic gates, also the subthreshold currents of the transistors exhibit a dependence on the input pattern [45]. Hence, the leakage (static) supply current can reveal a significant amount of information on the secret key, due to its strong dependence on the input of digital blocks [1] [45] [68]. Therefore, power analysis attacks based on leakage of CMOS are expected to be increasingly effective in down-scaled technologies and are becoming a critical issue. Thus instead of measuring the dynamic current drawn by a logic gate during its normal activity, the static power is taken into account as physical observable leakage.

The procedure of LPA attacks presented in [6] is similar to CPA. The main difference is that instead of collecting the current samples representing the instantaneous power consumption during one elaboration, the physical leakage is reduced to one single point representing the static power consumption of the circuit at a precise time instant (i.e. when all transient effects due to the switching of the signals are elapsed and the circuit is in a steady state condition). Therefore, LPA attacks are an example of *univariate* SCAs. In [5] LPA attacks have been formalized for bit-slice implementations and have proved to be potentially effective also against combinational non-bit-slice circuits; furthermore, authors demonstrate by extensive simulations that LPA attacks can be also effective against DPL styles, which are conceived to protect the circuit from PAAs against dynamic power and therefore are potentially vulnerable on the static power.

The contributions of this chapter are various: first, we extend the analysis in [5] and [6] by simulating the static power consumption of a specific case study cryptographic processor, the bit-slice *Serpent* encoder described in previous chapters, implemented with different DPLs in CMOS065 technology; we consider a bit-slice unit of the core, designed with combinational logic cells, and develop a specific simulation testbench to investigate the leakage of the circuit; for this purpose, we adopt some state-of-the-art security metrics and statistical evaluators borrowed by the context of DPA/CPA attacks, like the *Coefficient of Variation (CV)* and the *Point Biserial Correlation (PBC)* coefficient; then, we mount LPA attacks in a more realistic scenario, where also noise and process variations are taken into account, and used the *Minimum number of Traces for Disclosing the correct key (MTD)* for different values of SNR as actual security metric in order to evaluate the success rate and comparing the different implementations; finally, we test the strength of the iDDPL style, which is an implementation of the TEL data encoding, on LPA attacks and promote this kind of circuits also as a promising countermeasure for LPA.

As final remark, all simulations have been performed in Cadence Virtuoso environment using the CMOS065 technology described in the introduction. Experiments were performed by adopting *High Voltage Threshold General Purpose (HVTGP)* BSIM4 model transistors with standard-library sizing factors, but similar results can be obtained by using Standard or Low Voltage Threshold General Purpose

transistors or alternatively Low Power transistors.

5.2 Review of Leakage Power Analysis attacks

5.2.1 Context of LPA attacks

Similarly to conventional PAAs, the objective of LPA attacks is to recover the secret key k , or at least a portion of the key, of a cryptographic device by monitoring the power consumption of the device itself. This is done by comparing the collected current samples with a model of the power consumption (e.g. the Hamming weight, the Hamming distance, etc) of a specific word internally processed by the device. In the case of LPA attacks the observed leakage is represented by a single current sample for each elaboration; thus, unlike conventional PAAs where a leakage trace is composed of several points, in LPA the leakage is univariate and this allows to reduce the memory requirements of the adversary and the complexity of the attack.

A first important remark about the feasibility of LPA attacks regards the definition of the attack strategy. More specifically, LPA attacks are conceived as **non-invasive, passive attacks**, similarly to conventional PAAs, therefore the assumption is that an attacker does not interact with the device during its normal activity. However, in [89] authors state that PAAs on the static power are really dangerous in the concrete case that an attacker has direct control of the clock of the system and can significantly reduce the operating frequency, but this contradicts the assumption on the passively of attacks. Therefore, it is clear that the definition of the attack methodology is of primary interest.

As pointed out in [5], one of the most challenging aspects is to carefully perform the leakage measurements in a LPA attack scenario. When the input is applied to a logic gate, its leakage current is well known to have a transient variation and finally settles to the steady-state value after a period ranging from less than 1ns to a few tens of nanoseconds, on the basis of the technology node. Thus, leakage measures can be performed by considering only the current sample at the end of the clock cycle, when transient currents are elapsed.

In [5] many measurements of the settling time for different standard logic gates are presented for the case of the CMOS065HVTLP technology used in this thesis work. In such cases authors conclude that the settling times are generally comparable or greater than the period required to observe the steady-state leakage for deep submicron technologies: for example, a settling time equal to 100ns in CMOS065 technology means that a working frequency equal to 10MHz is sufficient to reach the steady-state condition at the end of an elaboration cycle. A cryptographic device usually works at frequencies below 10MHz (usually in the order of few MHz), as a consequence, we expect that in highly scaled devices transient effects have such a short duration that the current samples correspondent to the end of the clock cycle are generated by leakage contribution and the adversary is not required to stop the clock to perform static measures. Obviously, this condition must be verified case by case, because there are several logic gates switching inside a cryptographic circuit and the dynamic contributions can be predominant for a long time interval. Anyway, in other cases the attacker may want to calculate another parameter related to the low frequency leakage of a current trace, for example the average value of current

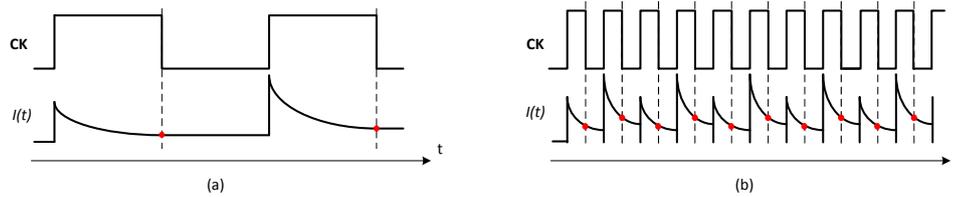


Figure 5.1. Current samples as a function of the operating frequency: static measures are feasible in the case (a), whereas in (b) dynamic power is predominant.

on a clock cycle, if it is not possible to reduce the working frequency or in general to detect the contribution of static power, as done for example in [85].

In accordance to these considerations and on the basis of the results in [5], in the simulation attack setup we do the assumption that the steady state condition is reached either using a sufficiently small operating frequency (in the order of few MHz) or stopping the clock, and only one point for each cycle as a representative leakage current sample is collected. We point out that for our simulations these two strategies are equivalent, and can be adopted indifferently.

In order to better understand to what extent the working frequency influences static measurements, in Fig. 5.1 two possible cases are shown: in figure (a), the working frequency is sufficiently low to guarantee that the steady-state condition is reached, whereas in figure (b) the working frequency is too high and the dynamic power is greater than the static power. This scenario agrees with the discussion in [89]: if the target implementation runs at (or is close to) the maximum frequency, the information leakage provided by static power remains substantially lower than that of dynamic leakage, so it is essentially useless; this is because its signal amplitude is significantly lower and a limited number of samples are available to exploit it. On the contrary, if clock frequency of a target device is sufficiently low or if the adversary can reduce it, the information provided by static power can be detected. This is because a low clock frequency may significantly increase the portion of the traces containing only static leakage.

In the simulations presented in this section we make some basic assumptions: first, the attacker knows exactly the architecture of the circuit under attack; then, even if the attacker does not have direct access to the internal clock of the system, the clock period is supposed to be much higher than the settling times of the dynamic current; finally, temperature is considered constant during the whole duration of measurements.

As a final remark, it is worth noting that the exact clock period in which a selection function inside the chip is evaluated can be easily found if the adversary has sufficient knowledge of the circuit implementation of the algorithm. Even in cases where the adversary does not exactly know this clock cycle but knows that it is among a limited number of clock cycles, he/she can reiterate LPA attacks for several clock periods and perform the procedure described in next paragraph.

5.2.2 Leakage model for bit-slice structures

In this paragraph we recall the leakage model on which LPA attacks are based. Let us consider the case of a bit-slice structure, with slices of m -bit. A m -bit-slice circuit is made up of m identical replicas of the same building block; arithmetic logic units, registers, register files and bus drivers are examples of bit-slice structure [5]. Named X_i the output of the circuit, the relation between the Hamming weight $w = H(X_i)$ and $I_{leak,i}$ is expressed by Eq. 5.1:

$$I_{leak,TOT} = w \cdot I_H + (m - w) \cdot I_L = w \cdot (I_H - I_L) + m \cdot I_L \quad (5.1)$$

where I_H and I_L are the leakage current for a high level and a low level in the corresponding input bit, respectively. For instance, in the case of a CMOS inverter they are the values $I_{leak,1}$ and $I_{leak,0}$ derived in Chapter 1. According to Eq. 5.1, leakage $I_{leak,TOT}$ exhibits a linear dependence on the Hamming weight w , rather than the specific value of each input bit.

In CPA attacks, the statistical distinguisher adopted to select the correct key is the Pearson's correlation coefficient ρ ; it gives an estimation of the linear dependence between two variables, thus it can be efficiently used also in LPA attacks, where according to Eq. 5.1 a linear relation exists between static current $I_{leak,i}$ and data X_i in a bit-slice structure.

5.2.3 Standard procedure for LPA attacks on bit-slice circuits

On the basis of the leakage model recalled in previous paragraph, LPA attacks were rigorously defined along with a clear five-step procedure, which is shown in Fig. 5.2 [5] and briefly described below.

In the first step of LPA attacks, the adversary chooses an internal m -bit signal X that is physically generated within the cryptographic circuit under attack. In the second step, the adversary applies 2^m different input values I_i (with $i = 1, 2, \dots, 2^m$), and measures the corresponding leakage current $I_{leak,i}$ of the cryptographic chip at the point of time in which X is physically evaluated. In the third step, the physical value of X within the chip is estimated for each input I_i and for each possible guess k_j of the secret key (with $j = 1, 2, \dots, 2^m$), thereby generating a 2D array of possible results $X_{ij} = f(I_i, k_j)$ (where X_{ij} is the value of X for the i -th input and j -th key guess). In the fourth step, the leakage current of the block generating X is estimated by the Hamming weight $H(X)$ of X , and the adversary generates a 2D array $H_{ij} = H(X_{ij})$ (where H_{ij} is the estimated leakage for the i -th input and j -th key guess and represents the selection function of the attack). In the fifth step, the measured leakage $I_{leak,i}$ and the estimated leakage H_{ij} are compared through the evaluation of their Pearson's correlation coefficient $\rho(I_{leak,i}, H_{ij})$. The correct guess of k is that leading to the highest value of $\rho(I_{leak,i}, H_{ij})$ among all possible guesses k_j .

Similarly to CPA attacks, the Pearson's correlation coefficient cannot be exactly determined when a finite number of samples $I_{leak,i}$ and H_{ij} is considered; then, in

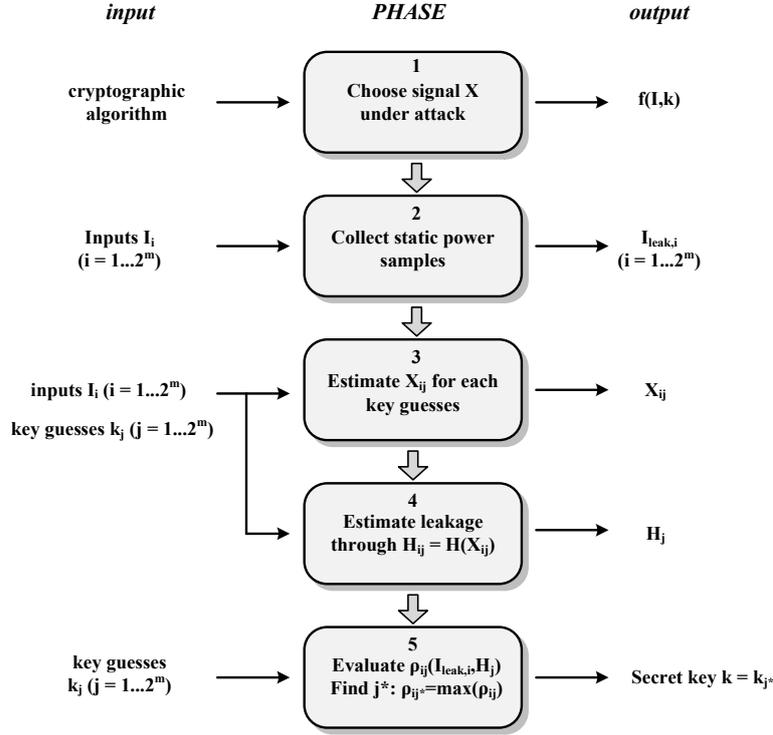


Figure 5.2. Block diagram describing the LPA procedure [5].

practical cases $\rho(I_{leak,i}, H_{ij})$ is estimated through the following estimator:

$$r_j = \frac{\sum_{i=1}^{2^m} (H_{i,j} - \bar{H}_j) \cdot (I_{leak,i} - \bar{I}_{leak})}{\sqrt{\sum_{i=1}^{2^m} (H_{i,j} - \bar{H}_j)^2 \cdot \sum_{i=1}^{2^m} (I_{leak,i} - \bar{I}_{leak})^2}} \quad (5.2)$$

5.2.4 Some considerations about noise in practical LPA attacks

An important factor for the effectiveness of LPA attacks is noise. In the context of PAAs there are two sources of noise superimposed on the exploitable signal: the electronic noise and the switching noise [72]. The switching noise represents the fraction of power consumed by the other logics on the chip which switch at different time instants and are uncorrelated to the useful signal. These logics contribute to the instantaneous power consumption through some current peaks during the elaboration. Similarly, in the context of LPA attacks two sources of noise can be distinguished: the electronic noise and the leakage noise. The leakage noise is defined as the contribution to the static power consumption due to the leakage of all the logics uncorrelated to the useful signal.

Even if the definition of noise for standard PAAs and LPA attacks is similar, however, there is an outstanding difference between dynamic and static. In general the instantaneous power consumption can be seen as the sum of a correlated and an uncorrelated component (electronic and switching noise) [72]. Standard PAAs

are based on the collection of several current samples during one elaboration; the switching noise due to the on-chip logics is distributed at different time instants, because each cell switches at different times; therefore in some time instants the correlated component is predominant with respect to the uncorrelated one, as revealed by the presence of peaks in the correlation curves for the correct key. On the contrary, LPA attacks are based on the collection of a single point for each elaboration, representing the logic state of the circuit at the steady-state condition, thus the effectiveness of LPA attacks is much more sensitive to the uncorrelated noise (leakage noise) and depends strongly on the amount of on-chip noise, which in turns depends on the complexity of a specific hardware implementation: if the uncorrelated part is predominant with respect to the useful signal, then LPA may be unsuccessful. In some sense, we can say that switching noise is distributed along a clock period and can be high or low in a specific time instant, whereas the leakage noise is a constant term.

Therefore, the amount of memory needed to perform static measurements, as well as the computational efforts to mount univariate attacks as LPA, is much less than the case of PAAs; the drawback is that static measures are much more time consuming if compared to those of dynamic power, being the fraction of exploitable signal smaller than the case of dynamic for the entire period. The presence of a dedicated setup to amplify the DC component as well as a low pass filter to reduce the HF noise may help to increase the exploitable power and reduce the time of acquisition, as concluded by authors in [85].

Another outstanding issue in practical LPA attacks is the high sensitivity to temperature of the static power consumption. Power analysis attacks require the acquisition of several current traces while the device runs for several hours (or days), and temperature is not constant during this period. A good solution is to perform experiments ensuring that the chip temperature is made not time varying, for example by using a thermoelectric cooling.

5.2.5 Security metrics to assess the vulnerability on the static power

The *Coefficient of Variation* (CV) can be used to assess the variability of the leakage of a logic circuit, but it is not sufficient to detect correlation between leakage and data for real practical attacks. From the discussion done in previous paragraph, it is necessary to define some security metrics and statistical distinguishers to assess the resistance of an implementation against LPA attacks. For this purpose, similarly to the case of standard PAAs, we use physical security metrics and actual security metrics to quantify the LPA-resistance of a logic circuit. The first metrics allow to detect physical leakage on the side-channel emission of a device when noise is taken into account, the second ones to exploit it as a matter of fact.

As physical security metric, *Signal-to-Noise Ratio* (SNR) represents still a good choice and has been used in other works on LPA [50] [89]. The SNR is simply defined as the ratio between signal and noise power. In the context of LPA attacks, signal power is defined as the fraction of the overall power consumption which contains relevant information potentially exploitable by an attacker, and can be calculated

as the variance of a leakage sample vector. Instead noise power is the sum of the electronic noise power and the leakage noise power:

$$SNR = \frac{P_{exp}}{P_{el.noise} + P_{leak.noise}} = \frac{\sigma_{exp}^2}{\sigma_{el.noise}^2 + \sigma_{leak.noise}^2} \quad (5.3)$$

The level of security strongly depends on the amount of overall noise. If the electronic noise can be averaged by enhancing the number of acquired samples, similarly to standard PAAs, leakage noise is more difficult to be eliminated because depends on the number of registers and logic gates in the circuit. More complex is the circuit, higher is $P_{leak.noise}$. This explains why leakage measures require the adoption of a dedicated setup to amplify DC signal and low pass filter the traces in order to reduce noise, as discussed in [85].

In order to understand how the static power consumption of the circuit is correlated to the input data and can be exploited by an attacker, it is useful to define an actual security metrics which makes use of a statistical evaluator. Similarly to the case of DPA attacks, the *Point Biserial Correlation (PBC)* represents a useful mean:

$$PBC = \frac{M_0 - M_1}{S_n} \sqrt{\frac{n_0 n_1}{n^2}} \quad (5.4)$$

The PBC will be calculated for each of the four bits at the output of the S-Box. Leakage samples are subset into two groups according to the value of the selected bit. M_0 (M_1) is the mean value on the leakages when bit is 0 (1), n_0 (n_1) is the number of leakage samples of the group associated to bit 0 (1), S_n is the standard deviation of the leakages distribution. Similarly to the case of standard PAAs, when PBC is applied to the static power analysis it estimates the level of correlation between the static power of a m -bit logic and a bit. In a bit-slice circuit (i.e. a m -bit register) PBC is expected to be equally high for each single bit, whereas in a non-bit-slice circuit one bit can be more or less correlated than another one, on the basis of how the gates are interconnected.

In the context of standard PAAs, *MTD* is an efficient security metric because allows to understand the efforts needed to detect the key in presence of a fixed noise level. MTD can be adopted also in the context of LPA attacks and represents the crossover point of the correlation coefficient curve of the correct key guess with respect to the curves of the other key guesses, and thus it assesses the resistance of a crypto-implementation against LPA in terms of minimum number of plaintexts required to recover the key. So, higher is MTD, higher is the resistance of the hardware implementation, noise being equal. MTD gives a direct estimation of the number of leakage measures to perform before an attacker could exploit the variability of the leakage of an implementation when a statistical distinguisher (e.g. the Pearson's correlation coefficient) is used.

5.3 Leakage current in combinational gates in CMOS 65nm

5.3.1 Evaluation of the variability of the leakage of single logic gates

In previous chapters we have studied some of the most important logic styles to counteract PAAs, the DPL styles. It has been discussed that according to the circuit primitives adopted to build a DPL style, the latter can be categorized as a logic level (standard-cell based) or a transistor level (full custom) countermeasure. In general these countermeasures aim at hiding (i.e. making constant) the dynamic power consumption for each data transition within a clock cycle, or masking it by randomizing the values of the power samples through a pre-processing operation on internal signals, by inserting redundant transistors in the logic cells with respect to standard CMOS. Thus, even if the purpose of these compounds is to equalize the dynamic power consumption, the balance of the static power consumption, which depends on transistor sizes, temperature, technology parameters and how the transistors are connected, is not ensured.

In this section we perform a preliminary investigation of the static power consumption of some combinational and sequential case study CMOS and DPL gates, i.e. WDDL, MDPL, and SABL; we have considered some basic case study cells: an input converter, an AND/NAND gate, a XOR/NXOR gate and a flip-flop. As it will be shown in next sections, these logic gates have been adopted to build a case study cryptographic primitive, which has been attacked using the procedure of LPA attacks.

As first step, we have chosen a metric to compare the variability of the static power in the DPLs cells as a function of the input data. For this purpose we use the Coefficient of Variation (CV), which has been already used in Chapter 3 to evaluate the balance of the instantaneous power consumption in the time domain of some DPL flip-flops. In the context of static power analysis, current measures are static and the leakage vector is composed of only n elements, one for each data input combination, thus CV is a single value and coincides with the *Normalized Standard Deviation (NSD)*. CV is calculated as the ratio between the standard deviation and the mean of the leakage sample I_j (with $j = 1, 2, \dots, n$); in general, larger is the coefficient, larger is the dependency of leakage on the input.

$$CV = \frac{\sigma}{m} = \frac{\sqrt{\frac{1}{n} \sum_{j=1}^n (I_j - m)^2}}{\frac{1}{n} \sum_{j=1}^n I_j} \quad (5.5)$$

5.3.2 Standard CMOS logic

The analysis of the leakage in CMOS gate is similar to the case of the inverter: the static power consumption of each gate can be measured by performing a DC analysis, given that standard logic gates are not clocked.

Each nMOS and pMOS transistor of a CMOS cell exhibits a static power consumption which contributes to the total leakage of the cell itself. This power consumption is due to the currents flowing in a single transistor when it is switched

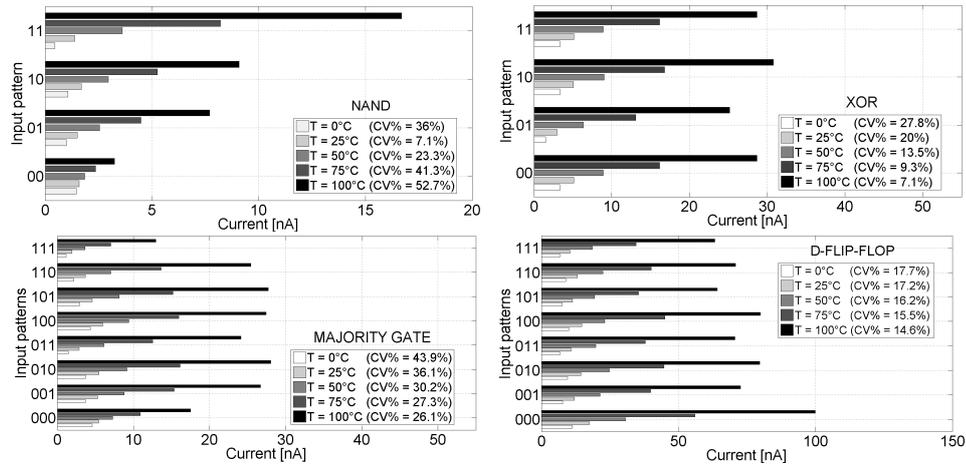


Figure 5.3. Leakage current variability for CMOS gates.

off (i.e. no dynamic transition is applied). However for each data combination there is a different value of the overall leakage current, being the threshold voltages of nMOS and pMOS different between each other. Indeed according to the topology and the number of nMOS and pMOS transistors, it is possible to correlate each value of the leakage current to a specific input data combination for each logic cell. In Fig. 5.3 simulation results for NAND, XOR, majority gates and flip-flop of the CMOS065HVTLP library (minimum fanout gates) are reported for different temperatures. The value of CV confirms a remarkable dependence of the leakage current on the input data. This analysis can be performed to any logic gate in the library.

It must be noted that for what concerns the input pattern of the flip-flop, the first bit in Fig. 5.3 is the clock, whereas the second and the third ones are data at the clock cycles i and $i - 1$ respectively. We use this convention also in next subsections, where simulations are repeated for DPLs.

5.3.3 Dual-rail Precharge Logic styles

For DPL gates a different testbench must be defined, because DPL circuits require an input stage to convert static signals from the single-rail to the dynamic dual-rail domain and may have an additional input signal routed into the combinational gate: the clock signal. As shown in Fig. 5.4, the leakage contribution is measured on the current flowing in the V_{DD} pin from the logic gate, after stopping the clock in correspondence to the first clock cycle. In the first case the clock is stopped during the precharge semi-period and corresponds to a DC analysis, in the second case in the evaluation semiperiod after the rising edge.

The DPL styles under analysis have been already recalled in previous chapters. In next paragraphs a brief description of each one is also provided.

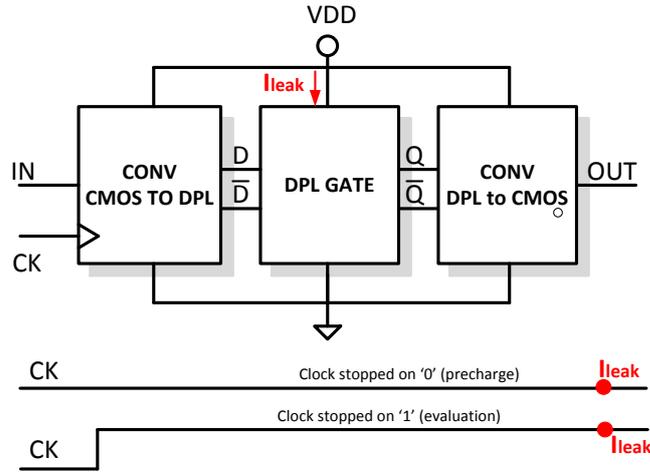


Figure 5.4. Simulation testbench to measuring the leakage of DPLs.

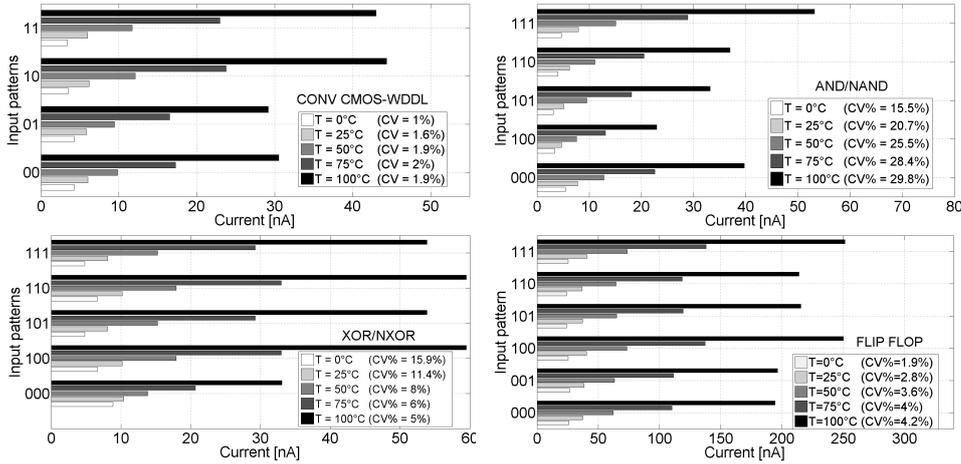


Figure 5.5. Leakage current variability for WDDL gates.

Wave Dynamic Differential Logic

Wave Dynamic Differential Logic (WDDL) is the first DPL under analysis. It is designed through CMOS standard-cells. However unlike CMOS the range of possible input patterns is limited by the input converters: for instance, during the precharge phase ($CK = 0$) dual-rail signals (D, \bar{D}) are both forced to be low irrespective of the input data. This is an important factor to be taken into account when performing static current measurements in all DPL gates.

Given that WDDL is built as compound of CMOS cells, WDDL gates are expected to exhibit an analogous dependence between leakage and input. Simulation results are reported in Fig. 5.5, and actually confirm our expectation, even though the variation is slightly reduced. Note that during the precharge, all signals are always low, and for combinational gates only the pattern '00' is admitted.

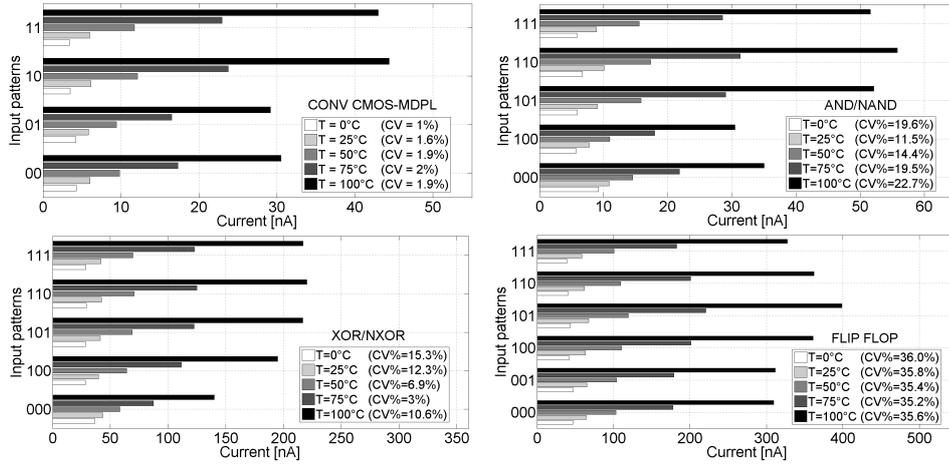


Figure 5.6. Leakage current variability for MDPL gates.

Masked Dual-rail precharge Logic

Masking is another countermeasure against DPA attacks and it is based on the randomization of the intermediate value of the cryptographic device by pre-processing input data. *Masked Dual-Rail precharge Logic (MDPL)* is a DPL style which elaborates masked values as a function of data d and a random mask value m . Similarly to WDDL, MDPL is another example of standard-cell based logic style. We consider MDPL gates where data are XORed with the mask, being logic XOR the most simple masking function:

$$d_m = f(d, m) = d \oplus m \quad (5.6)$$

Results are shown in Fig. 5.6. The values of CV confirm that also in MDPL there is a certain dependence between the original (i.e. not masked) input data and leakage current. It is worth noting that CV is calculated by averaging the values obtained by using $m=0$ and $m=1$.

We conclude that even if masking helps to de-correlate power consumption and input patterns, it is not effective to break correlation between input data and static power. This was expected, because masking logics randomize the dynamic power consumption but are still sensitive on the leakage of the devices, given that MDPL gates are built with majority gates available in the standard library [98], and the leakage of a majority gate is strongly related to the input pattern, as already shown in Fig. 5.3.

Sense Amplifier Based Logic

Full custom DPLs attempt to equalize the dynamic power consumption of a logic cell through an optimized dual-rail circuit topology and a dynamic precharge/evaluation clocking phase which forces only one logic data transition for each cycle. In a dynamic elaboration the clock period is subset into an evaluation and a precharge phase, thanks to the presence of transistors acting like charged capacitances at the different clock semi-periods. The charge and discharge of these capacitances provide

the transient response of a dynamic cell at the output nodes [101]. Unlike the case of WDDL and MDPL, in a dynamic DPL circuit pipeline also the combinational cells are clocked, thus the simulation testbench is different (Fig. 5.7). In this case an additional clock cycle is required before stopping the clock and performing static measures, in order to set up the initial precharge phase.

Extending LPA attacks to dynamic logics is not a trivial task. As recalled in the previous section, once an input data is applied to a logic gate, the current adsorbed from the power supply is well known to have a transient variation and finally settles to the steady-state value after a period depending on the technology node. But waiting for the transient to be elapsed means measuring the adsorbed current after the dynamic data encoding has occurred. The assumption that the clock period is comparable or even greater than the period required to observe the steady-state leakage in a nanometer technology [5] may be not more valid for dynamic logic families and must be verified in order to perform a LPA attack. For this reason in practical measurements it must be ensured that the steady-state condition is satisfied so that static measure the static are feasible.

Sense Amplifier Based Logic (SABL) is one of the first full custom DPA-resistant logic styles having a dynamic and differential behavior. As seen in previous chapters, SABL data encoding is based on the RTZ protocol. When $clk = 0$, differential signals are forced to be also 0; on the contrary, when $clk = 1$, one of the differential signals is 0, the other one is 1. Thus, leakage measures have been executed keeping into consideration only these possible configurations in accordance to the testbench of Fig. 5.7, as reported in Fig. 5.8. It is clear that SABL shows a remarkable robustness against leakage analysis, being CV of the leakage distribution very low. This is true in particular for symmetric cells, like flip-flop and XOR, whereas a slight dependence between input and leakage can be detected for the AND/NAND gate due to the asymmetry of the cell. This indicates that dynamic DPLs offer better performances in terms of balance on the static power, but the asymmetry of a logic function, and thus of the logic cell, highlights a (slight) data-dependance which can be still exploited.

5.4 Leakage Power Analysis attacks and lightweight cryptography

5.4.1 Leakage model for a generic non-bit-slice structure

The leakage model in Eq. 5.1 is valid under the assumption that the circuit is a bit-slice structure with m bits. The linear dependence between the input Hamming weight distribution and the leakage has been demonstrated for some registers implementations in simulation and confirmed by measurements [5]. In general, registers can be reasonably considered an outstanding leakage source also for the case of static power consumption.

However there are different parts of a chip that are typically non-bit-slice. In cryptographic circuits, more specifically in lightweight implementations, several logic blocks are built as combinational circuits, characterized by a netlist of logic cells. As

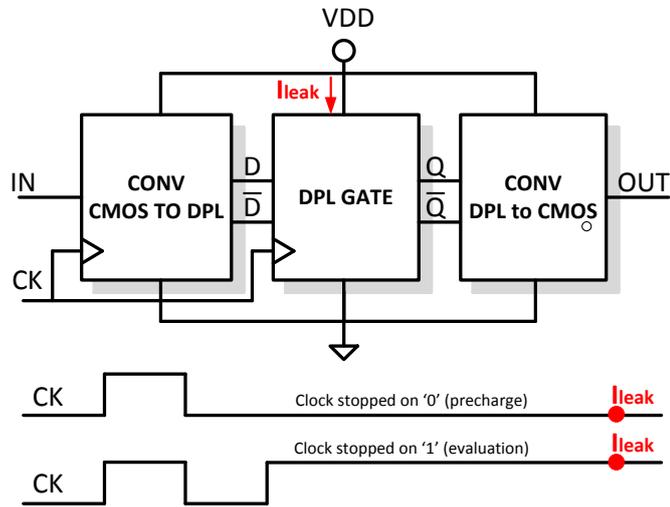


Figure 5.7. Simulation testbench to measure the leakage of dynamic DPLs.

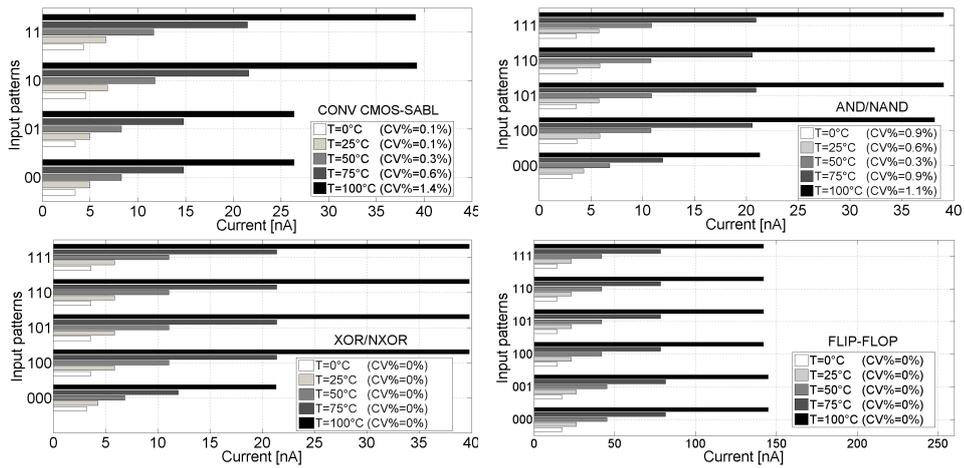


Figure 5.8. Leakage current distribution for SABL gates.

seen in previous paragraphs, combinational logic gates have also a data-dependent static power, this means that it is still possible to exploit the variability of the leakage with the input data to perform attacks. However, the validity of the linear model of Eq. 5.1, demonstrated for bit-sliced structures, is not ensured also for the case of combinational non-bit-slice circuits, and must be verified on a specific hardware implementation. Indeed the model is based on the assumption that the overall static consumption $I_{leak.tot}$ of a bit-slice circuit is given by the sum of the leakage currents of the m -bit-slices assuming that each contribution is I_H (I_L) when the corresponding input bit is 1 (0), but for non-bit-slice circuits this is not more valid because the static power strongly depends on how transistors are connected.

According to a generalized model, the overall static power consumption of a circuit is given by the sum of a component, linearly dependent on the Hamming weight of the processed datum, and a component which is a non linear function of the Hamming weight:

$$I_{leak,TOT} = K \cdot w + f(w) + C \quad (5.7)$$

The parameter C is a constant term and depends on the process and temperature variations. The non-linear function $f(w)$ takes into account the dependence between the static power consumption of the non-bit-slice blocks and the Hamming weight w of the processed word. This function is in general unknown and depends on the circuit topology of the non-bit-slice blocks.

The presence of a non linear term creates a sort of decorrelation effect: with reference to Fig. 5.9, the correlation between the words being processed through the circuit and the static power consumption is reduced along the pipeline due to the presence of the non-bit-slice block:

$$corr[I_{leak}, H(w_0)] > corr[I_{leak}, H(w_1)] > corr[I_{leak}, H(w_2)] \quad (5.8)$$

If the attacker tries to exploit the variability of the static power of the circuit by using as target word the output of a non-linear block, which is very common in PAAs applications, the leakage model to be considered is that represented by Eq. 5.7; the feasibility of linear LPA attacks is not guaranteed and should be verified using other statistical distinguishers than the Pearson's correlation coefficient.

In next paragraphs a cryptographic case study is considered; after a deep investigation of the leakage power model for this architecture, LPA attacks are mounted using the linear model and in presence of noise. It will be shown that under some assumptions, which concern the possibility to linearize the function $f(w)$, the general model of Eq. 5.7 can be approximated with the linear model of Eq. 5.1 and LPA attacks are successful even if the architecture is not rigorously bit-slice.

5.4.2 Static power analysis methodology for a lightweight cryptographic implementation

As already discussed in Chapter 1, the trend of modern lightweight cryptography is to design hardware efficient cryptographic primitives. The core of these implementations is certainly the S-Box block. Earlier cryptographic algorithms were characterized by 8x8 S-Boxes which were very efficient for software implementation, whereas in hardware they are implemented using look up tables because a combinational description would require too much area (i.e. hundreds of Boolean gates). In

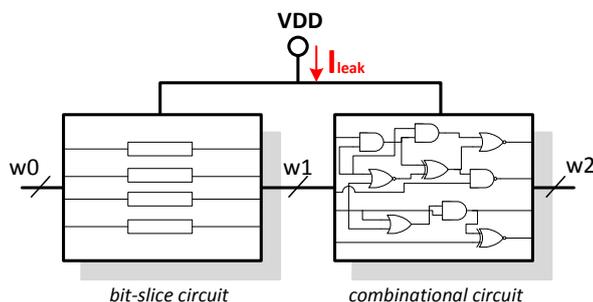


Figure 5.9. A generic circuit composed of two cascaded blocks: one bit-slice (registers) and one non-bit-slice (combinational).

lightweight applications, smaller S-Boxes, for example 4x4 bit S-Boxes, are typically used; they offer the advantage to be implemented with a very few number of Boolean gates by executing a direct synthesis of the logic function of the block. S-Boxes are cryptographic blocks whose purpose is to implement a non linear function which confers the property of diffusion to a cryptographic algorithm. Thus, cryptographic S-Boxes represent a straightforward example of non-bit-slice block.

For non-bit-slice circuits, the dependence of the static power on the Hamming weight of the output datum can be expressed by the general model of Eq. 5.7. Even if the function $f(w)$ is unknown, a relation exists between static power and word w . In this section, we propose a standard procedure to investigate this relations in order to provide a sound evaluation of the resistance of a generic lightweight cryptographic processor against attacks on the static power consumption. The methodology is applied to the case of the bit-slice implementation of Serpent block cipher depicted in Fig. 2.17, but could be extended to other lightweight block ciphers.

1. **Selection of a m -bit-slice unit of the core.** The cryptographic processor is a bit-slice implementation with n -bit ($n = 128$), thus it can be seen as the repetition of a number N of identical m -bit-slice sub-blocks ($N = 32$ and $m = 4$). The contributions of each single 4 bit-slice to the overall static power consumption can be considered independent, then it is sufficient to consider the static power consumption of one single slice, depicted in Fig. 5.10, and to recover a nibble of the key like in a *divide and conquer* strategy.
2. **Selection of the target block for attacks.** The target block represents the logic function to be attacked. In PAAs, the word at the output of the S-Box at the first stage is usually chosen as target, thus we consider the first stage of the bit-slice as target block. This block is just the SERPENT-block already described in previous chapters, whose data path is the 4 bit architecture reported for simplicity in Fig. 5.11, and the first round key is a 4 bit word.
3. **Analysis of the static power of the non-bit-slice core.** Each 4-bit-slice unit is composed of registers, XOR gates and combinational 4x4 S-Boxes. As previously described, for static power attacks the relevant block is the

non-linear S-Box which is a non-bit-slice block. The leakage model is depicted in Eq. 5.7, thus in order to assess the level of decorrelation represented by the function $f(w)$, the static power consumption of the S-Box is calculated for each input data configuration, and its statistical properties are analyzed.

4. **Analysis of the static power of the entire target block.** A similar analysis has been conducted on the SERPENT-block, and simulations are repeated for every possible key. As previously described, the PBC expresses the linear correlation between the static power consumption and one bit of the circuit. Thus, if one or more bits have a high PBC for the correct key, the combinational block can be still approximated as a bit-slice circuit for some bits, we can do the assumption that $f(w) \approx w$ and the linear model can be applied.
5. **LPA attacks against the bit-slice with a noise model.** Static power measurements on the SERPENT-block are executed with SPICE-level simulations, which are noise free. The overall static power consumption of the bit-slice is estimated as the sum of the contribution of the 8 layers composing the bit-slice (S0, S1, ... S7) and the permutation layer blocks, but the relevant leakage is given by a single layer, i.e. the SERPENT-block. We do the following assumptions: first, the contribution of the permutation layer can be neglected, because this operation can be executed directly on the round registers without the insertion of additional logic; then, the static power contributions of the single layers can be considered independent. Thus, using the first stage as target block means that the exploitable signal is just the static power consumption of this block, whereas the static power of the other layers in the bit-slice can be considered as leakage noise $I_{leak.noise}$ with a Gaussian distribution and standard deviation $\sigma_{leak.noise}$. LPA attacks with the linear model are executed by summing up different level of noise on the static power samples of the target block, and for each level of SNR the MTD is calculated for each implementation.

The target block for the attacks is the SERPENT-block widely described in previous chapters. With reference to the architecture of the bit-slice Serpent encrypting processor shown in Fig. 2.17, the SERPENT-block is just the first round of one bit-slice. The core of the implementation is the Serpent 4x4 S-Box S_0 , which represents the non-bit-slice block. The hardware implementation of the S-Box is described by a set of 57 logic instructions (see Appendix A) and the resulting netlist is represented in Fig. 4.3.

We point out that this methodology is in accordance to the fact that simulating the entire processor in Cadence would be highly time-consuming. Thus, using the *divide and conquer* strategy and consider a noise model for the leakage noise allows to reduce time efforts and have a good evaluation of the LPA-resistance of the circuit.

5.4.3 Analysis of the static power of the combinational S-Box S_0

As first step, we investigate the effect of decorrelation of the S-Box block. We point out that the function $f(w)$ assesses the dependance between static power

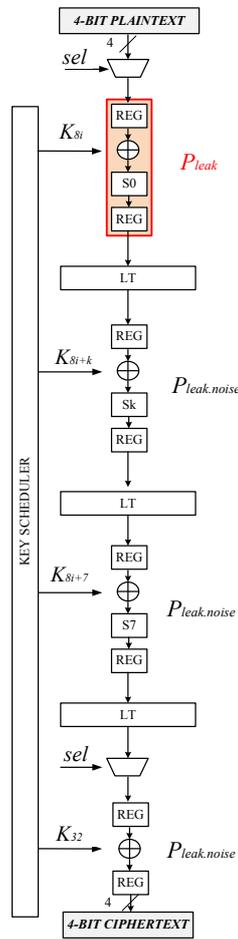


Figure 5.10. One bit-slice of the Serpent encryption processor.

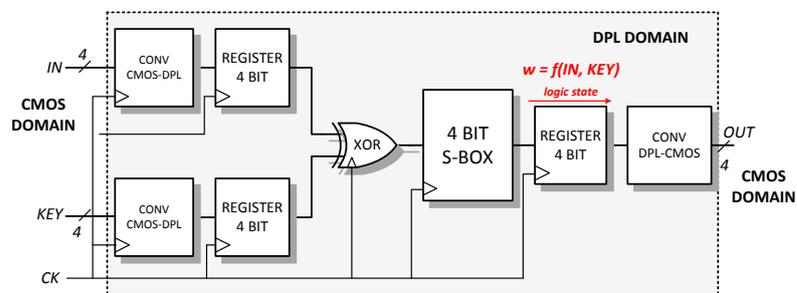


Figure 5.11. Data path of the first stage of a bit-slice (SERPENT-block) for DPLs.

Table 5.1. Correlation between static power of the S-Box and the Hamming weight of the input/output words.

Implementation	$\rho[I_{leak}, H(IN)]$	$\rho[I_{leak}, H(OUT)]$
CMOS	0.82	0.37
WDDL	0.3	0.43
MDPL	0.9	0.14
SABL	0.63	0.62

and Hamming weight of the processed word. According to the plot in Fig. 5.9, the purpose of this paragraph is to determine to what extent the S-Box decorrelates the output word (w_2) with respect to the input word (w_1). For this purpose we calculate the Pearson's correlation coefficient between $H(w)$ and I_{leak} . The S-Box has been implemented with the logic gates analyzed in the previous sections.

In Fig. 5.12, the average static power consumption as a function of the Hamming weights $H(w_1)$ and $H(w_2)$ is depicted for each implementation. Simulations have been executed at constant temperature ($T = 25^\circ C$). In Table 5.1 the correlation coefficient between the Hamming weights and the static power consumption of the S-Box are also reported. It is straightforward to note that the level of decorrelation strictly depends on the implementation: for instance, in the case of SABL the effect of decorrelation is reduced, whereas in WDDL it is even increased. This result is useful to understand the impact of the combinational netlist on the leakage model, but it is not sufficient to infer the LPA-resistance of an implementation, because it does not take into account the standard deviation of the current samples and the leakage noise.

In addition, it is useful to study the dependence of the static power on the value of the single bits of the output word w , by calculating the PBC for different temperatures, in order to understand if one or more bits are correlated to the static power irrespective of the value of the correlation with the logic word. Results of PBC for different temperatures are shown in Fig. 5.13. From these figures it is clear that the DPL S-Boxes are characterized by a different law of dependence between single bits and static power. Some information on the implementation can be deduced from this figures: for instance BIT4 of the MDPL S-Box is poorly correlated to the leakage for any temperature ($PBC < 0.1$), similarly as BIT1 of the SABL S-Box, whereas all the bits of CMOS and WDDL are well correlated to the leakage current for temperatures greater than $25^\circ C$ ($PBC \approx 0.3$) and leak more information.

We remark that the PBC is mathematically equivalent to the Pearson's correlation coefficient estimator used in the LPA procedure for bit-slice circuits, with the main difference that it indicates the correlation between a single bit and static power. As it will be shown in next paragraphs, it can be sufficient that one bit is highly correlated to the physical emission that the approximation $f(w) \approx w$ holds and linear LPA attacks can be feasible even against a non-linear block.

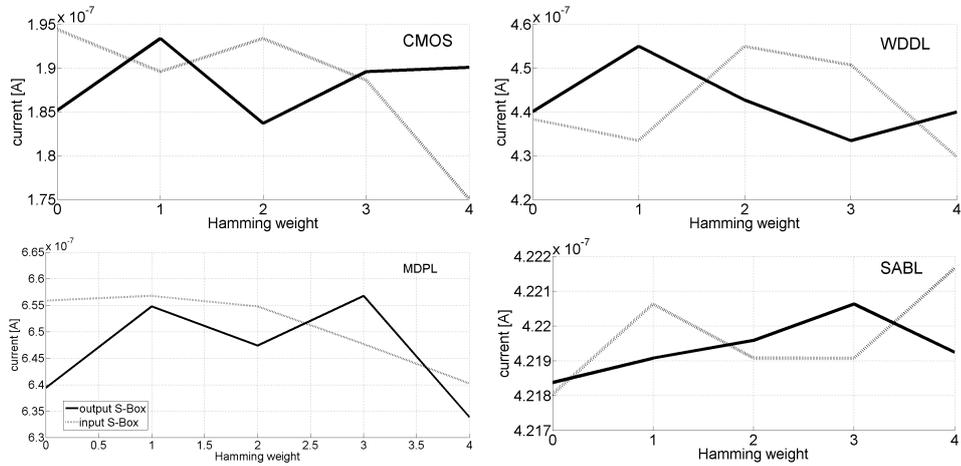


Figure 5.12. Leakage current as a function of the Hamming weight of the input (dotted line) and the output (straight line) words of the S-Box.

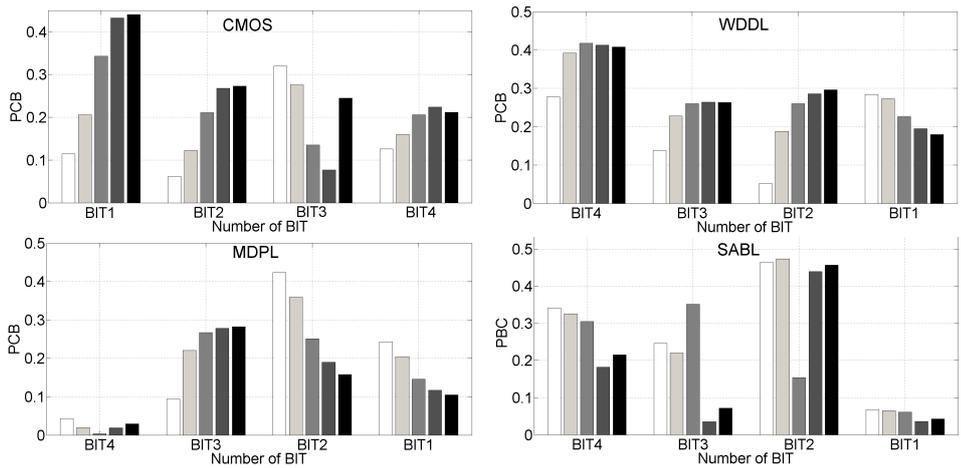


Figure 5.13. PBCs between leakage and output bits of DPL S-Boxes.

5.4.4 Analysis of the static power of the SERPENT-block

The core of the SERPENT-block is certainly the S-Box. In previous paragraph the static power consumption of the S-Box has been analyzed as a function of the Hamming weight of the input/output words and in relation to the PBC for each single output bit. In this paragraph we investigate the data-dependence of the static power of the entire SERPENT-block. The data-path of the circuit is reported in Fig. 5.11; apart from the S-Box, in the circuit there are two levels of registers, which are bit-slice blocks, two levels of converters and the XOR logic gate. The secret key stored in the key register of the crypto-core was set to $(5)_2 = (0101)$, but any other choice would lead to the same results.

The circuit has been clocked with a sufficiently low frequency signal in order to reach the steady state condition at the third clock period of the elaboration, when the output register loads data from the combinational stage. In accordance to the situation depicted in Fig. 5.11, the static measures are performed by sampling the current traces in correspondence to the logic state of the output register. In Table 5.6 the static power consumption distribution of the circuit is reported for all the DPL implementations. Words w_1 and w_2 are the input and the output words, respectively.

The variation of the leakage of the SABL implementation is limited, as expected in a dynamic DPL and as already seen during the analysis of single gates. Similarly to DPA attacks, for every DPL implementation and for each bit of the word w a prediction matrix has been built, where each row corresponds to the PBC of a key guess. In Fig. 5.14 the values of PBC have been reported for each bit, and the PBC associated to the correct key is depicted in a bold red line. It must be noted that for BIT1 the correct key is always associated to highest PBC, thus it leaks information independently of the logic style, whereas BIT4 is the less correlated, and this is in accordance to the plots of Fig. 5.13. Instead, the level of correlation of BIT2 and BIT3 depends on the implementation: for example MDPL leaks information on BIT2, whereas SABL on BIT3. In any case, at least two bits have one of the highest correlation coefficients on the correct key for each DPL circuit.

In Fig. 5.15 the trend of the static power as a function of $HW(w)$ is shown. It must be noted that these plots are almost identical to those obtained for S-Box (Fig. 5.12), because the insertion of bit-slice logics (XOR and registers) has a low impact on the level of correlation between static power and Hamming weight.

The static power analysis done in this and in previous section confirms that the combinational S-Box block reduces the linear correlation existing between the static power consumption and data processed in the pipeline. However, if the circuit is not really complex, and this is the case of 4x4 S-Boxes, the level of decorrelation is low and there are always some bits which leak information. Analogous considerations were made in the past for traditional DPA/CPA attacks on the dynamic power consumption, which were extended to combinational logic blocks in a very similar way [81]. Interestingly, in this respect LPA and DPA/CPA attacks are similar. In general, higher is the correlation between the leakage of the S-Box and the Hamming weight of the selection function, more feasible are LPA attacks, as in the case of CPA attacks. Furthermore, higher is the number of bits having high correlation, higher are the probability that static power reveals information, similarly to the

Table 5.2. Leakage currents measured for the DPL crypto-cores as a function of the input pattern.

w_1	w_2	HW(w_2)	Current [μA]			
			CMOS	WDDL	MDPL	SABL
0000	0110	2	0.5388	1.4775	2.6930	1.1209
0001	1010	2	0.5376	1.4615	2.6880	1.1210
0010	1011	3	0.5322	1.4557	2.6860	1.1209
0011	0101	2	0.5378	1.4777	2.6700	1.1206
0100	1000	1	0.5383	1.4770	2.6960	1.1211
0101	0011	2	0.5353	1.4588	2.6730	1.1212
0110	0001	1	0.5373	1.4754	2.6780	1.1210
0111	1111	4	0.5336	1.4657	2.6260	1.1211
1000	0000	0	0.5342	1.4687	2.6840	1.1207
1001	0111	3	0.5285	1.4406	2.6630	1.1210
1010	1100	2	0.5271	1.4550	2.6640	1.1208
1011	1001	2	0.5323	1.4644	2.6560	1.1206
1100	1101	3	0.5313	1.4602	2.6610	1.1211
1101	1110	3	0.5316	1.4585	2.6580	1.1212
1110	0010	1	0.5374	1.4737	2.6660	1.1208
1111	0100	1	0.5380	1.4738	2.6590	1.1208

case of DPA attacks, and the circuit can be approximated as a bit-slice structure, which can be attacked successfully with and the linear model of Eq. 5.1, as it will be shown in next paragraph.

5.4.5 LPA attacks against the crypto-processor considering noise

As final step, after having discussed the possibility to linearize the non-linear dependence between the static power consumption of a combinational logic and the Hamming weight of the processed word, LPA attacks with the linear model of Eq. 5.1 have been performed against the Serpent processor shown in Fig. 2.17. According to the bit-slice implementation of the circuit, the overall power consumption can be modeled as the sum of the contribution of the N bit-slice units composing the circuit ($N = 16$). One bit-slice is shown in Fig. 5.10. Being the bit-slices identical, the static power consumption is also approximately identical for each bit-slice. Actually this is not completely true, because the *Linear Transformation* layer mixes the bits, resulting in different contributions for each bit-slice. However the assumption that the overall static power of the circuit is the sum of N identical independent contribution is reasonable. Therefore the overall static power consumption is a multiple of the static power consumption of a single bit-slice, and by exploiting the linearity of the LPA attack procedure, Eq. 5.1 is still valid.

These assumptions allow to do a simplification on the attack procedure: given that the overall static power of a single bit-slice unit of m bits is a scaled version of the static power of the overall processor, the attack procedure can be used by considering only the static power of a single bit-slice ($m = 4$). This way it is possible

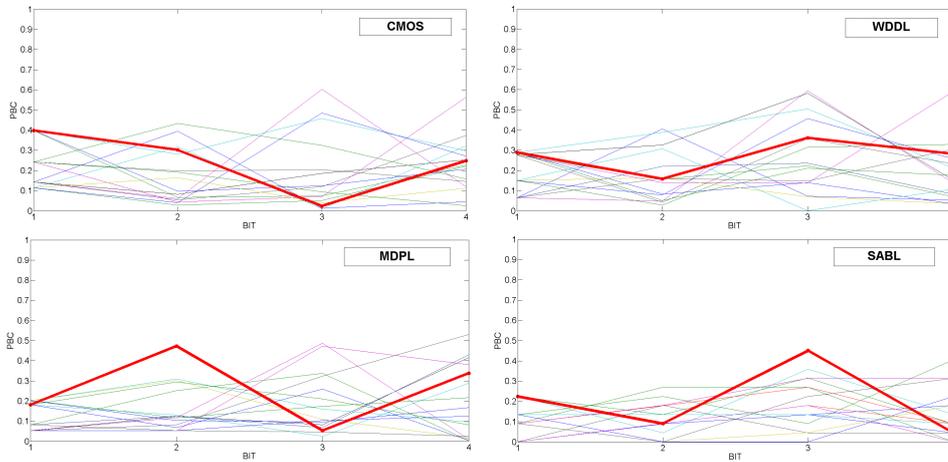


Figure 5.14. PBCs for the DPLs implementation of SERPENT-block (correct key in bold red line)

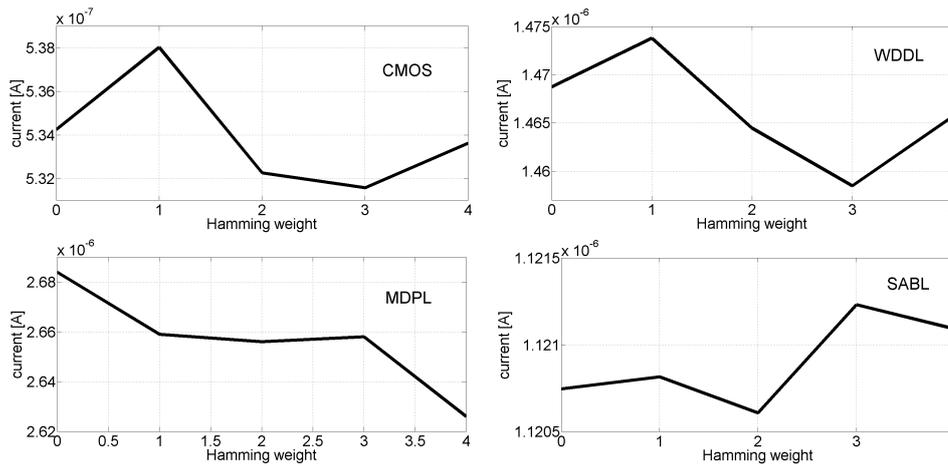


Figure 5.15. Leakage current as a function of the selection function $w = f(IN, Key)$

to recover 4 bit of the round-0 key, and by reiterating the procedure for each bit-slice it would be possible to disclose the entire key, 4 bits at a time, like in a *divide and conquer* strategy.

In the m -bit-slice unit of Fig. 5.10) the SERPENT-block represents the target circuit for the attacks (red in figure). As already discussed, the SERPENT-block is the minimal non-bit-slice unit of the circuit, being composed of combinational logics. However we have also seen that the circuit is sufficiently small to be approximated as a bit-slice circuit and can be attacked using the linear model. The target function of the attack is a 4-bit sub-word at the output of the first layer of the processor, that corresponds to the word at the output of the S-Box S_0 of the SERPENT-block.

In order to emulate real physical conditions and make LPA evaluation meaningful, noise should be considered, both electronic and leakage noise, given that SPICE-level traces are inherently noise free. With reference to the Eq. 5.3, the exploitable leakage power is just the static power consumption of the first block of the bit-slice unit (red in figure), which is represented by a vector of samples describing leakage for any input data configuration with their standard deviation σ_{exp} . The electronic noise can be considered as an uncorrelated stochastic process with a Gaussian distribution and standard deviation $\sigma_{el.noise}$, similarly to what done in Chapter 2 for the case of CPA attacks. The leakage noise is the static power consumption of the other logic instantiated on the chip which contribute to the overall static current consumption. According to the statistical analysis in [72] and [116], the overall noise of the circuit can be modeled as an additive Gaussian noise superimposed to the signal power [104]. By using this approach the overall on-chip Gaussian noise accounts both for electronic and leakage noise, thus even the leakage noise can be considered as Gaussian noise.

At this point we do the following assumption: given that the amount of leakage noise in a bit-slice depends on the implementation, the only variable which can be parametrized is the electronic noise. The leakage noise is modeled in this way. The selected bit-slice unit in Fig. 5.10 is composed of 12 flip-flops and about 40 combinational gates implementing the S-Box S_0 . It is straightforward to show that the other S-Boxes $S_1 \dots S_7$ can be also designed with on average the same number of logic gates. Thus the number of transistor required to design each bit-slice circuit is almost identical. The *Linear Transformation* block is a sequence of operations which involves circular rotations and shifts of the bit of the 32-bit words stored in the status registers, requires only a reduced number of logic gates (i.e. about additional 8 XOR gates for each round), and are the same for each round. We neglect the fraction of the power consumption due to the key scheduler, being in some cases designed off-chip or in an external protected memory. Thus, the level of exploitable signal for leakage power analysis P_{exp} can be estimated as almost one eighth of the leakage noise $P_{leak.noise}$, and the minimum amount of noise to consider for making simulated LPA attacks valid is $\sigma_{leak.noise}^2 \approx 100 \cdot \sigma_{exp}^2$. This represents the minimum amount of noise which has been considered in simulations.

Under these assumptions, the LPA procedure was repeated by summing a Gaussian vector with an increasing standard deviation on the static power samples, measured at $50^\circ C$, and using a maximum capacity of the attack equal to 150k measurements. Then, the same procedure has been reiterated for different values of noise. In Fig. 5.16 the MTD as a function of the SNR is depicted. By using

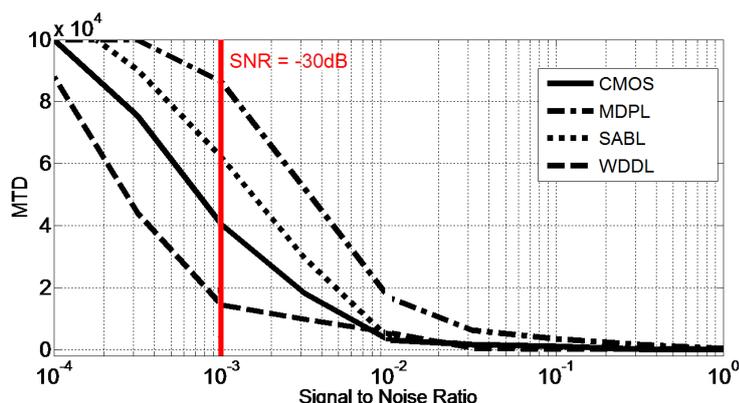


Figure 5.16. Minimum number to disclosure as a function of the SNR.

150k measurements and averaging the noisy traces in order to reduce the noise for each plaintext, the correct key was disclosed up to a SNR almost equal to nearly -40dB for all the logic styles under analysis, which corresponds to a maximum noise variance 10k times greater than the signal variance. This value is well beyond the minimum acceptable SNR of 100, and takes into account the excess of disturb due to the electronic noise. We remark that this value is valid in simulation, and the effect of all other noise sources must be taken into account in real experiments. However this is just a good estimation of the LPA resistance of the processor.

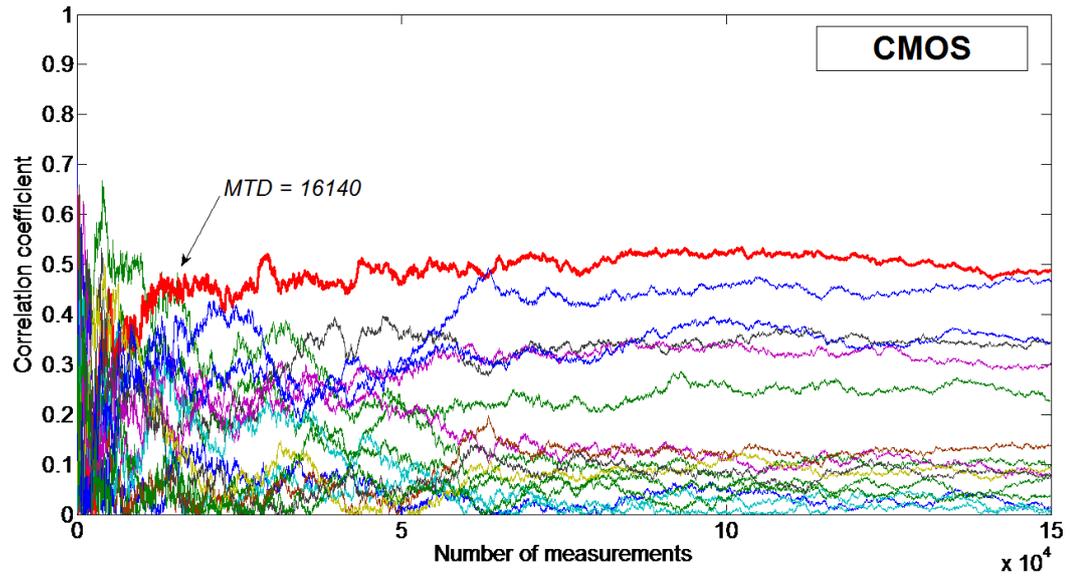
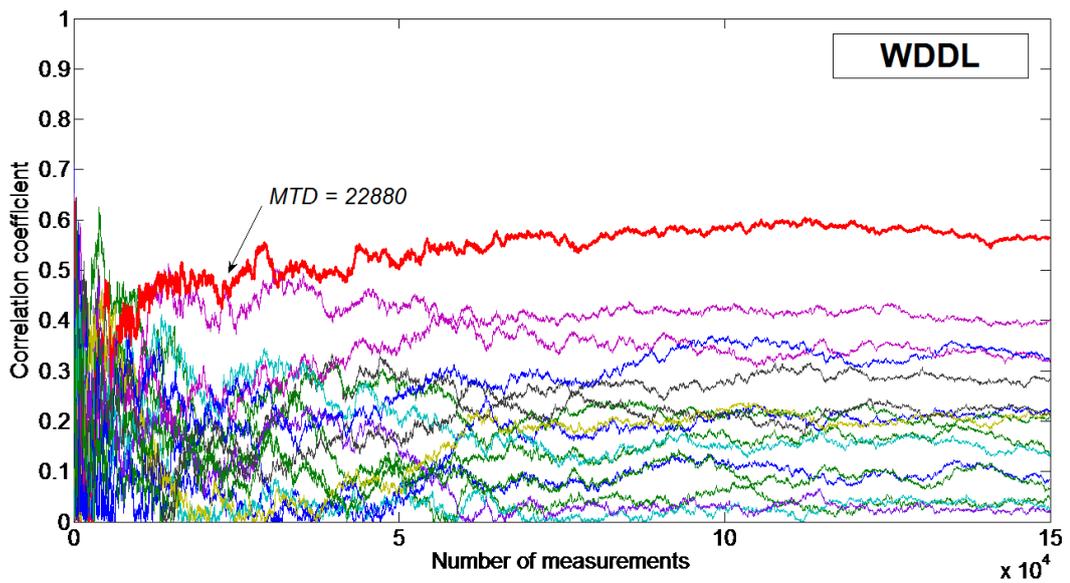
As a second set of simulations, we plotted in Figures 5.17 to 5.20 the correlation coefficient for all the correct key guesses with a fixed level of noise (i.e. $\sigma_{leak.noise}^2 = 1.000 \cdot \sigma_{exp}^2$, which corresponds to SNR = -30dB) in order to evaluate the MTD for each crypto-implementation. With level of noise, the 4-bit key can be successfully recovered for all the logic implementations. The minimum number of measurements for disclosure the key ranges from ≈ 16.000 for CMOS, which confirms to be the most vulnerable scheme, up to ≈ 50.000 for MDPL. In Table. 5.3 $\rho_{correct}$ is the value of the correlation coefficient of the correct key guess after 150k static measurements, which is a good estimation of the correlation when an unbounded attack is mounted (i.e. the value of the correlation coefficients calculated when the number of traces is high enough to obtain the convergence towards an asymptotic value [116]). Instead $max[\rho_{wrong}]$ is the maximum asymptotic value calculated among all the wrong keys. The *asymptotic gain* G is calculated as the ratio between $\rho_{correct}$ and $max[\rho_{wrong}]$ and gives an estimation of the leakage resistance of an implementation.

The results of the attacks demonstrates that hardware implementation which have a bit-slice architecture are potentially vulnerable to LPA attacks, and at least in simulation they can be successfully attacked with a some hundreds of thousands of measurements using the linear LPA procedure and considering noise. The presence of DPL styles, which are useful for enhancing the level of resistance against PAAs on the dynamic power consumption, fail at protecting the circuit against attacks on the static power.

As a final remark, we point out that leakage noise is not strictly uncorrelated to the exploitable leakage power, because the logic state of the circuits in the bit-slice have a (slight) dependence on the exploitable power.

Table 5.3. Actual security metrics for the DPL crypto-cores (SNR = -20dB).

	CMOS	WDDL	MDPL	SABL
MTD	16.140	22.880	50.820	45.280
$\rho_{correct}$	0.529	0.578	0.545	0.391
$\max[\rho_{correct}]$	0.431	0.479	0.489	0.201
$G \rightarrow \infty$	1.227	1.207	1.115	1.945

**Figure 5.17.** Correlation coefficients as a function of the number of measurements for CMOS (the curve of the correct key is in red).**Figure 5.18.** Correlation coefficients as a function of the number of measurements for WDDL (the curve of the correct key is in red).

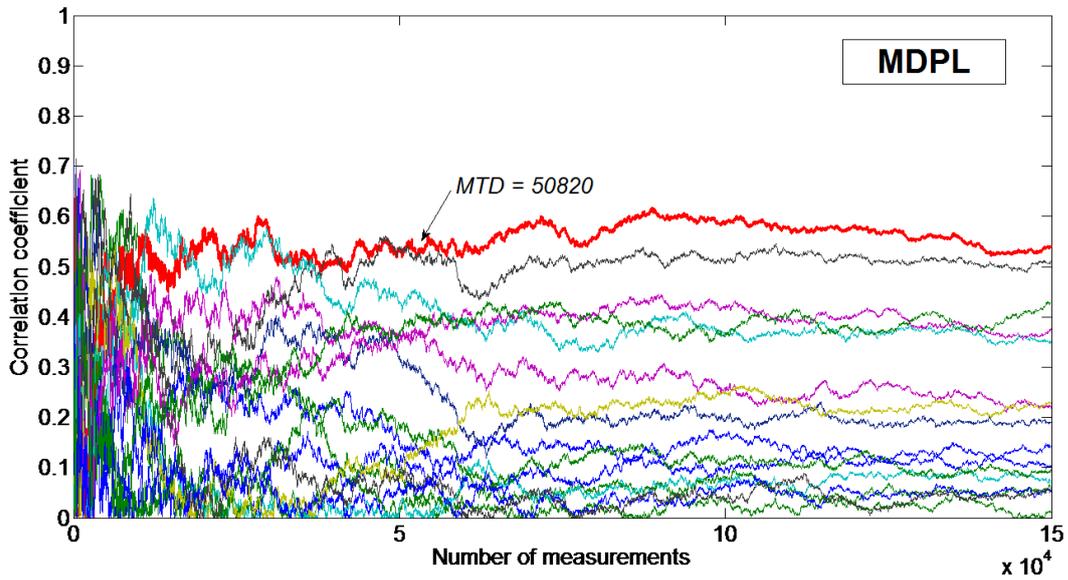


Figure 5.19. Correlation coefficients as a function of the number of measurements for MDPL (the curve of the correct key is in red).

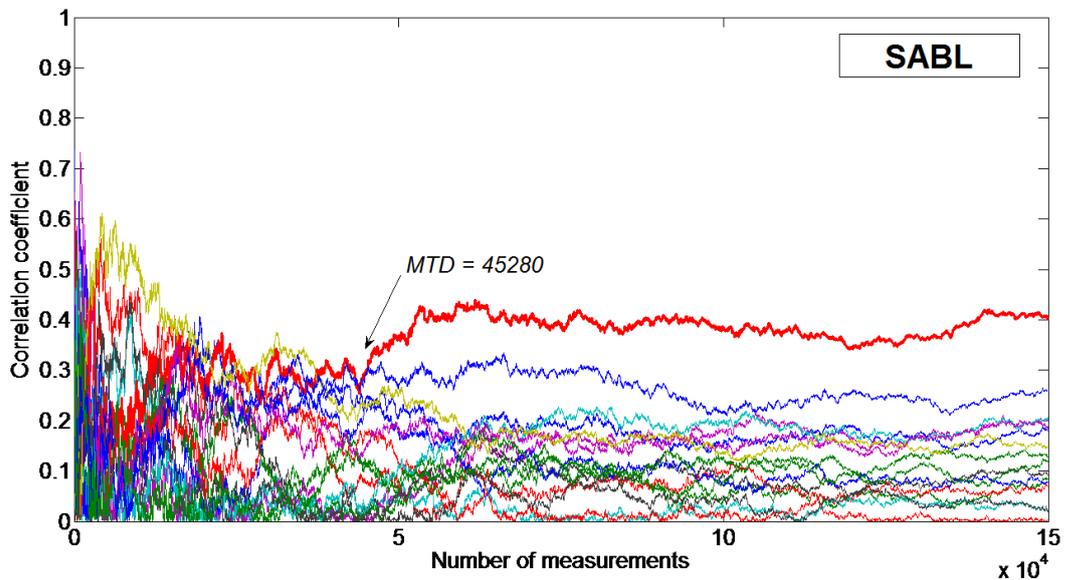


Figure 5.20. Correlation coefficients as a function of the number of measurements for SABL (the curve of the correct key is in red).

5.5 Evaluation of the LPA effectiveness with power variability issues

5.5.1 Impact of intra-die and inter-die process variations on the actual security metrics adopted in LPA

In general, VLSI circuits are subject to both inter-die and intra-die process variations [92]. Inter-die comprises not only process variations affecting all the chips on one single wafer, but also the variations on different wafers, and even on different lots. Actually, these comprise all possible variations between the standard process corners. Accordingly inter-die variations are expected to have the same effect on the value of the overall consumption of the chip, and in particular the leakage I_{leak} due to the bit-slices sub-circuits is shifted according to the same (random) factor. Therefore the correlation coefficient between I_{leak} and the Hamming weight of the input data that determines the outcome of the LPA attack is not affected by inter-die variations.

On the other hand intra-die variations represent the mismatch of devices which are located close to each other (i.e. adjacent), and which should match as closely as possible. This kind of variations affects all devices within a chip in a different manner. For this reason intra-die process variations are expected to have an impact on the outcome of LPA attacks, as they differently affect the leakage I_H (I_L) for different bit-slices (see Eq. 5.1), thereby impacting the resulting correlation coefficient both in the case of bit-slice and random logic blocks [5]. As a consequence, logic gates exhibit a different leakage consumption in presence of intra-die variations even when the same inputs are applied. Because in this case the leakage depends on the specific value of the input (not only on its Hamming weight), a slight deviation from the linear trend in Eq. 5.1 is also expected.

The effect of the intra-die process variations can be modeled as a Gaussian variable which causes an uncertainty on the value of the correlation coefficients calculated in the previous section. Namely the asymptotic value of the correlation coefficients for each key guess is also a Gaussian variable with mean ρ and a standard deviation depending on the process variations. Thus the extension of the uncertainty ranges and so their overlapping (or not) depends on the range of random variability of the technology process which should be investigated at a given technology node. Even if intra-die process variations might cause a LPA attack strategy to be ineffective, the attack is still feasible if the value ρ under the correct key guess can be distinguished from that under a wrong guess. Thus it must be ensured that the uncertainty ranges of the corresponding correlation coefficients do not overlap. If a sample circuit exhibits a correlation coefficient for the correct key guess less than the correlation coefficient for one wrong key guess, the LPA fails.

It has been demonstrated [5] that by simulating a circuit instantiation under small to moderate process variations in a CMOS 65nm technology node, the key guesses with input Hamming distances (to correct key) higher than one cannot cause an overlapping correlation coefficient distribution. So a LPA attack can identify the correct key in the most cases, or find at most a one-wrong-bit key with a high probability. Namely the standard deviation due to the process spread is higher than the uncertainty of a LPA attack in distinguishing a one wrong bit to the correct one,

but lower than the uncertainty of a LPA attack in distinguishing a two wrong bit to the guess one. Thus LPA attacks under intra-die variations are expected to be successful in current and future technology generations and in the next sections we will demonstrate these preliminary results by simulating LPA attack on the case study crypto-core.

Observe that such overlap might lead to an attack failure or not, i.e., such overlap does not convey information on the LPA effectiveness. Indeed, the presence of a correlation coefficient ρ for the wrong key $(4)_2$ that is higher than the correct key $(5)_2$ for different realizations does not indicate any attack failure (an attack fails only if ρ for key $(4)_2$ is higher than $(5)_2$ in the same realization). In other words, the statistical distribution of the correlation coefficient for different keys is not sufficient to deduce the effectiveness of a LPA attack.

According to the above considerations, the impact of variations on the effectiveness of LPA can be understood only by evaluating the percentage of realizations such that the correct key is actually associated with the highest correlation coefficient, and assessing the LPA resistance in terms of mean and standard deviation of the actual security metrics as MTD.

5.5.2 Impact of intra-die variation on the leakage model

Before performing LPA attacks on the SERPENT-block core, it is useful to investigate how the leakage model of the chip is affected by the intra-die variations. For this purpose we performed the attack on 100 samples of the circuit, each one generated by means of Monte Carlo simulations.

The 100 sample circuits generated in each experiment were all affected by mismatch variations. Note that each of 100 Monte Carlo iterations represents a realization of the circuit under test with a particular configuration of process random variables. The operating temperature is equal to 50°C . BSIM4 models with statistical parameters provided by the foundry were used [19] [117], which were previously validated on silicon over a large number of commercial integrated circuits. The distribution have been subset in five groups, according to the value of the selection function (i.e. the Hamming weight of $w = f(IN, Key)$).

In Fig. 5.21 to Fig. 5.24 the leakage current trend versus the Hamming weight is plotted, both for mean and standard deviation. As expected the linear trend is still visible for the mean leakage, and fits well the trend in nominal case plotted in Fig. 5.15. Moreover the coefficient of variation has been calculated for each Hamming weight of the DPLs and shown in Table 5.4. It stays within the range of $1 \div 3$, confirming that moderate process variations have a rather limited impact on the LPA model. For instance, for MDPL the $CV\%$ is lower than the others, so we expect that LPA in presence of process variations could be less effective in MDPL with respect to CMOS and SABL. Instead for the latter the mismatches have a stronger effect in terms of variance on the linear model.

As suggested in [104], the resistance of a crypto-core against power analysis can be assessed in the worst case by profiling and attacking the same chip. Each sample circuit exhibits a different leakage depending on the impact of process mismatches on the LPA model, according to the mean and the standard variation, as previously described. Thus we evaluate the impact of intra-die process variations by mounting

LPA against all the sample circuits and calculating the MTD and the asymptotic correlation coefficient for the correct key guess for each realization. We use the success rate as a metric to evaluate the effectiveness of LPA attacks with a maximum attack capacity N on the sample circuits, defined in this scenario as the number of sample circuits which are successfully attacked with a $MTD \leq N$:

$$SR_i = Pr\{Exp = 1\} = Pr(\rho_{corr} > \rho_{wrong}) = \begin{cases} 0, & \text{if } MTD > N \\ 1, & \text{if } MTD \leq N \end{cases} \quad (5.9)$$

$$SR \% = \sum_{i=1}^{100} SR_i \quad (5.10)$$

After having simulated static current traces from Cadence environment at an operating temperature equal to $50^\circ C$, simulated traces were post-processed by adding a Gaussian noise in order to consider also the on chip noise as done in the nominal case. According to the plot in Fig. 5.16, we considered $\sigma_{leak.noise} = 100 \cdot \sigma_{exp}$, which leads to a SNR = -20dB. As seen in previous section, this noise variance takes into account the leakage noise. LPA attacks were mounted with a maximum number of traces N equal to 100k.

The overall effect of intra-die variations can be modeled as a statistical variation in the security metrics used in previous section to assess the effectiveness of LPA attacks. For instance, if we consider the MTD, which is defined as the minimum number of measurements for which the curve of the correlation coefficient ρ of the correct key guess crosses over the curve of the maximum correlation coefficient of the wrong keys and asymptotically maintains a value higher than the others key guesses [122], it can be seen as a Gaussian variable centered on its nominal value and with a range of uncertainty around it (see Fig. 5.25). Thus it must be assured that the distribution of MTD does not overstep the boundary due to the maximum number of measurements for not increasing the complexity of the attack.

The MTD was calculated for all the sample circuits which were successfully attacked by LPA. In Fig. 5.26 the distribution of the MTD is depicted for all the logic styles.

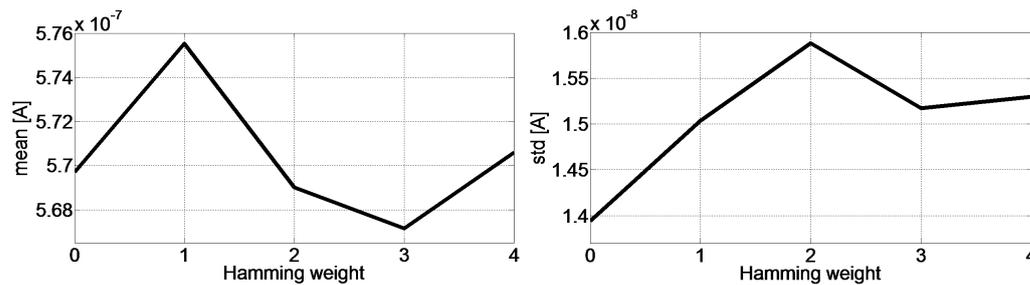
As anticipated, MTD exhibits a Gaussian distribution around a mean value which is very similar to the value calculated in the nominal case. However, the MDPL crypto-core exhibits a lower standard deviation which confirms the low impact of process variations in the leakage model for this DPL logic style. Instead SABL exhibits a higher variance which highlights a bigger impact of process variations on the leakage model, mainly due to the symmetry of the cells which aims at balancing the dynamic power consumption.

Results of LPA attacks against the crypto-cores are reported in Table 5.5. In the success rate row, the number of successfully attacked circuits when the one-bit wrong key is also considered and reported between brackets. The values of the mean asymptotic correlation coefficient for the correct key guess and the mean correlation gain agree to the nominal values reported in table.

Results confirm that LPA is still effective against the most popular cryptographic logic styles adopted to balance dynamic consumption and designed with nanoscaled devices, even when physical variability due to the on-chip noise and the process

Table 5.4. Coefficient of variation (%) for the static currents distribution measured for the 100 crypto-core sample circuits designed with DPLs.

HW	0	1	2	3	4	μ
CMOS	2.447	2.612	2.791	2.675	2.681	2.641
WDDL	1.559	1.455	1.519	1.463	1.436	1.486
MDPL	1.082	1.041	1.157	1.150	1.180	1.122
SABL	1.922	1.836	1.801	1.741	1.803	1.821

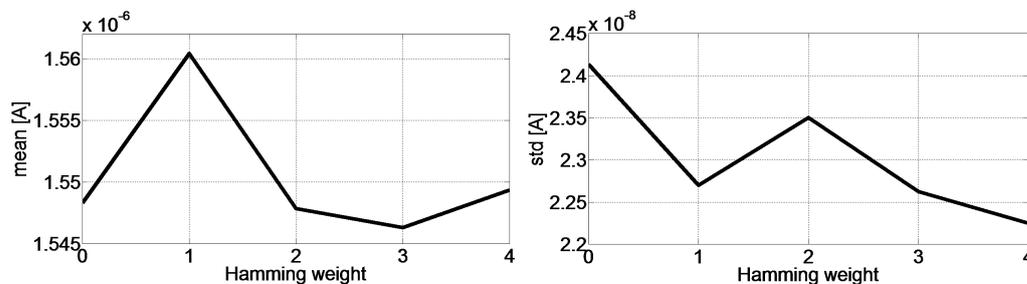
**Figure 5.21.** Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 CMOS sample circuits.

mismatches are taken into account. Process variations unavoidably impact the effectiveness of LPA depending on the specific architecture of each logic style, but the leakage model is still exploitable for all the logics under analysis.

5.6 Investigation on the static power variability in TEL circuits

5.6.1 Practical considerations on LPA attacks against TELs

In this section we investigate the feasibility of static power attack against TEL circuits. Following the considerations done in this chapter, also the static power consumption of a TEL circuit can be measured by doing some assumptions on the clock signal: in an ideal scenario in which an attacker has direct access to the clock signal of the device, he/she may want to stop the clock at a specific cycle and

**Figure 5.22.** Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 WDDL sample circuits.

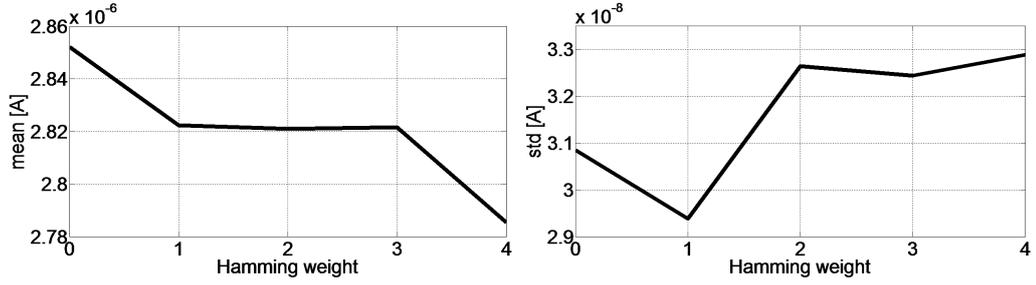


Figure 5.23. Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 MDPL sample circuits.

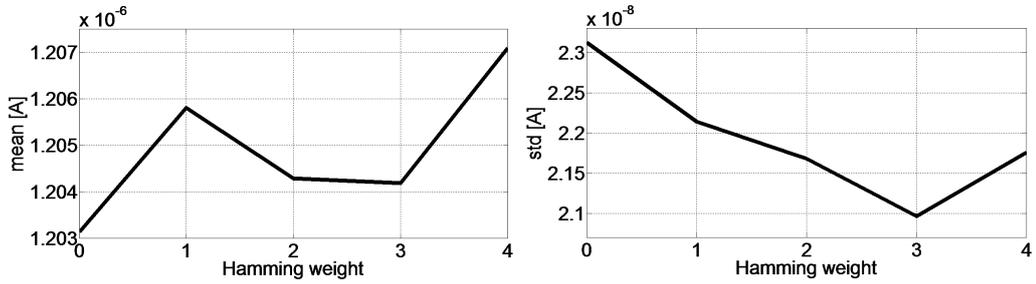


Figure 5.24. Mean and standard deviation of the leakage current distribution versus input Hamming weight over 100 SABL sample circuits.

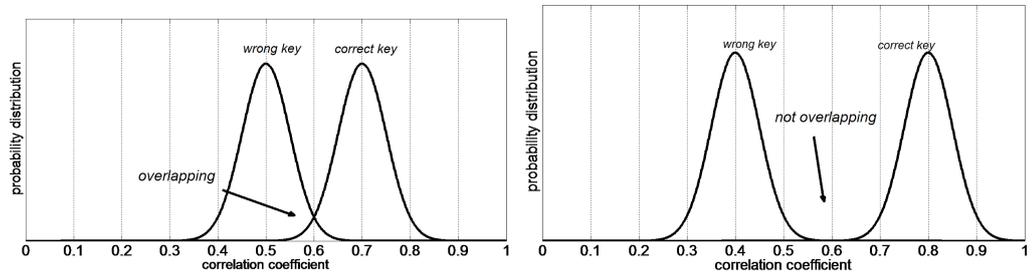


Figure 5.25. Statistical distribution of the correlation coefficient under high intra-die variations (success rate < 1) (left) and moderate intra-die variations (success rate = 1) (right).

Table 5.5. Actual security metrics for the DPL crypto-cores successfully attacked with a LPA procedure (SNR = -20dB).

	CMOS	WDDL	MDPL	SABL
SR%	26 (51)	23 (38)	49 70	17 (32)
max[MTD]	97702	78048	95200	99728
mean[MTD]	14770	17075	13876	24978
mean[ρ_{correct}]	0.522	0.529	0.584	0.495
mean[MTD]	1.299	1.296	1.255	1.276

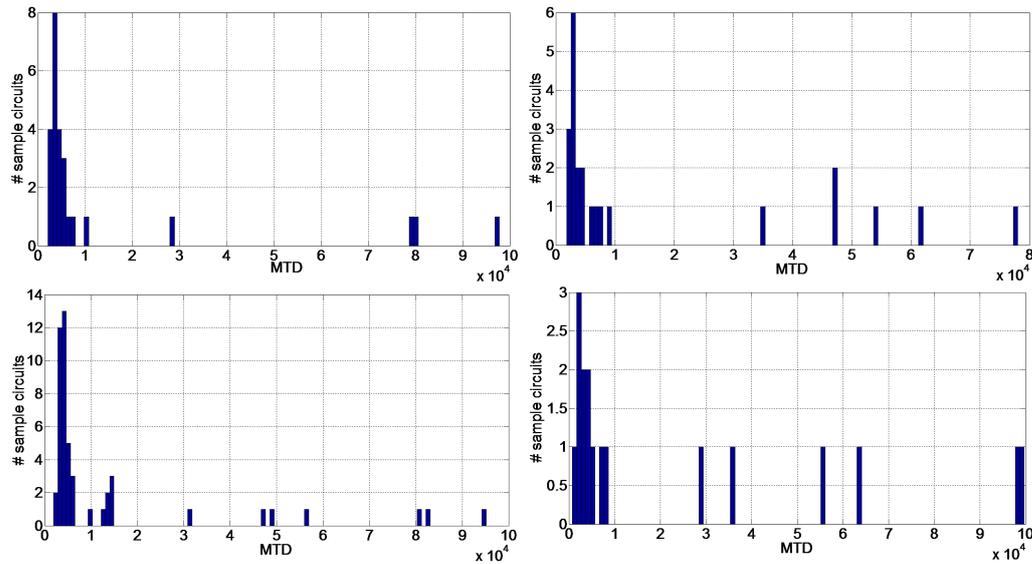


Figure 5.26. Distribution of the MTD for the successful LPA attacks against the CMOS (left upper), WDDL (right upper), MDPL (left lower), SABL (right lower) sample circuits.

measure the current samples in condition of steady-state; this condition is relaxed if we consider that the settling times of the circuit are lower than the clock period, which is reasonable in the typical working frequency range of cryptographic devices. However, the main advantage of TEL circuits is the presence of a third state in the data protocol which allows to perform a fully dynamic encoding: when the clock period is sufficiently high so to reach the steady-state condition, the output lines are always at the same values of the clock. For instance, when the clock is fixed to '0' (precharge), the output signals are also (0, 0), whereas when the clock is '1' (postcharge) the output signals are always (1, 1), and this is irrespective of the SR input datum A. This means that the instantaneous current trace reaches the steady-state condition always in the postcharge or in the precharge phase, when the signals are in an invalid logic state and have the same voltage (0 or V_{DD}).

Actually, TEL circuits can be attacked by using another strategy: if an attacker has direct access to the internal signal V_{bias} , or at least can modify the nominal value, he/she can modify the relevant period of the TEL circuit in order to highlight the leakage of the logic cells; more specifically, he/she may increase the value of V_{bias} and consequently increase the value of δ ; according to the tuning range of Fig. 3.8, some hundreds of mV are sufficient to extend δ up to $\delta_{MAX} = \frac{T_{CK}}{2}$, turning the TEL encoding into a RTZ encoding.

In general, these two attack strategies lead to two different current samples sets. The situation is reported in Fig. 5.27 for a differential signal pair of a TEL gate: when the leakage is measured on the normal activity of the circuit with a clock period sufficiently long (figure (a)), the static power samples in the two input configurations are identical, $I_{leak,0} = I_{leak,1}$; on the contrary, if the interval δ is increased up to δ_{MAX} , the TEL data encoding is turned into a RTZ data encoding and the leakage values are in general different, $I_{leak,0} \neq I_{leak,1}$, according to the

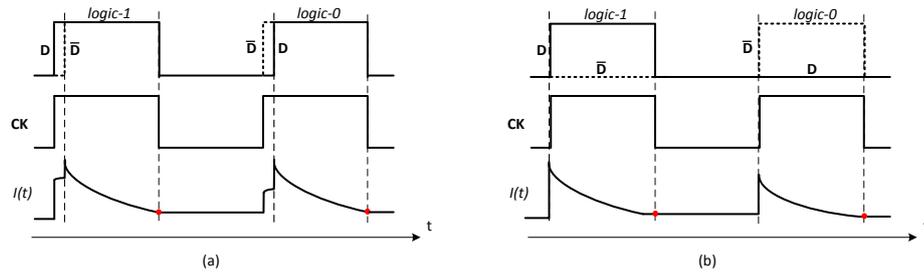


Figure 5.27. Measure of static power on the current traces of a TEL circuit when the static voltage V_{bias} is unchanged (a) and changed (b).

circuit template adopted for implementing TEL circuit.

We point out again that in order to evaluate the security of any circuit in a non-invasive attack scenario, the assumption is that the attacker cannot interact with the device and, in particular, he/she neither can modify its operating conditions nor have access to the internal signals. The consequence is that the internal voltage V_{bias} of a TEL circuit cannot be externally modified by the attacker. This means that the only feasible strategy to perform static measurements and mount LPA attacks against a TEL circuit in practical cases is to use a low clock frequency and collect the steady-state current samples, similarly to the other DPLs.

5.6.2 Leakage distribution on single iDDPL gates

The iDDPL style has been introduced as a circuit template to implement the micropipeline TEL circuit presented in Fig. 2.7, and will be used in this section as case study for evaluating LPA in a TEL circuit. According to the conclusions of last paragraph, when performing DC measures on iDDPL combinational and sequential gates, only two possible situations can be considered: when $clk = 0$, $D = \bar{D} = 0$; when $clk = 1$, $D = \bar{D} = 1$. This is not true for the input converter, where the input data are static and may have four possible configurations, leading to different values.

The distribution of the static power consumption as a function of the input is reported in Fig. 5.28. CV has been calculated for different temperatures.

5.6.3 Leakage model of an iDDPL circuit

Let us consider again the circuit describing the first stage of Serpent, reported in Fig. 5.29. The leakage model defined for the case of DPL bit-slice circuits is different in iDDPL circuits. More specifically, we consider the case in which an attacker can have access to the clock signal of the circuit and can stop the clock at a certain time instant, having knowledge of the exact data-path of circuit. This simulation setup can be unfeasible in practical cases, as discussed in previous paragraphs, but it helps to understand the behavior of the iDDPL during static power measures.

Using this setup, the most critical part during the static power measures are the converters at the input interface (Fig. 5.30): when signal clk is stopped to V_{DD} , also the signal ckd , which is an in-phase delayed replica of clk , stays at V_{DD} . This means that transistors N2-N5 and N3-N6 are always in conduction, forcing the differential

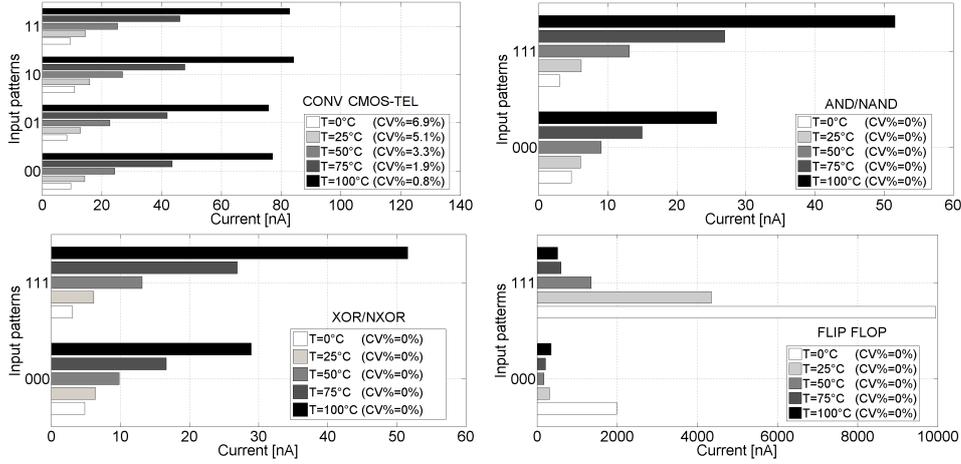


Figure 5.28. Leakage current distribution for iDDPL gates.

output signal to V_{DD} , independently from the commutations of the inputs data. Thus, the differential signals propagating along the pipeline are always in the invalid state (1, 1) (postcharge phase), and the overall static leakage is constant for each data input, according to the model of Fig. 5.27 (a).

Indeed, when $clk = clkd = 1$, the static power consumption of an input converter depends on the input signal IN . The contribution of the converters to the overall leakage can be generically represented as $I_{leak, CONV}$. As above described, when signals are in the postcharge, the static power is equal for every logic gate. This means that the overall static power of the circuit is given by the sum of a non-constant component, due to the input and the key converters, and a constant component, which comprises the contribution of the input data register, the key register, the combinational logics and the output data registers, which are completely uncorrelated from the data, given that the signals propagating along the pipeline are in the invalid state. Under the assumption that the static power of the converters is high (I_H) when $IN = 1$, and low (I_L) when $IN = 0$, the leakage power model for the iDDPL circuit is:

$$\begin{aligned}
 I_{leak, TOT} &= I_{leak, CONV} + I_{const} = \\
 &= w_{in} \cdot I_H + w_{key} \cdot I_H + (m - w_{in}) \cdot I_L + (m - w_{key}) \cdot I_L + I_{const} = \\
 &= w_{key} \cdot (I_H - I_L) + w_{in} \cdot (I_H - I_L) + 2m \cdot I_L + I_{const} = \\
 &= (w_{in} + w_{key}) \cdot (I_H - I_L) + I'_{const}
 \end{aligned} \tag{5.11}$$

The power model demonstrates that there is no possibility to select an internal word in the circuit which is correlated to the key, the input data and the logic operation (e.g. XOR or S-Box). In Table. 5.6 the single contributions to the static power of the circuit are measured for the iDDPL implementation. According to the data-path of the circuit, the clock signal is shown in Fig. 5.29; the static power samples are measured in correspondence to the red point of figure for any possible data configuration in input at the first cycle. As expected, the only variable contribution is due to the converter, which however is uncorrelated to the internal cryptographic operations and does not allow to build a sound power model.

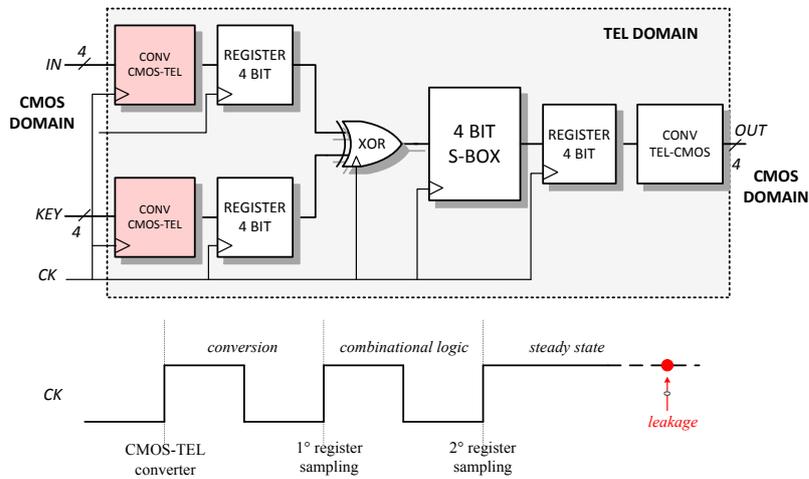


Figure 5.29. Data path of the cryptographic circuit under test and clock signal diagram during static measures.

Table 5.6. Leakage currents measured for the iDDPL crypto-core as a function of the output pattern.

IN	OUT	HW(OUT)	$I_{leak,TOT}$	$I_{leak,CONV}$	I_{const}
0000	0110	2	21.374	0.213	21.161
0001	1010	2	21.372	0.211	21.161
0010	1011	3	21.372	0.211	21.161
0011	0101	2	21.370	0.209	21.161
0100	1000	1	21.372	0.211	21.161
0101	0011	2	21.370	0.209	21.161
0110	0001	1	21.370	0.209	21.161
0111	1111	4	21.369	0.208	21.161
1000	0000	0	21.372	0.211	21.161
1001	0111	3	21.370	0.209	21.161
1010	1100	2	21.370	0.209	21.161
1011	1001	2	21.369	0.208	21.161
1100	1101	3	21.370	0.209	21.161
1101	1110	3	21.369	0.208	21.161
1110	0010	1	21.369	0.208	21.161
1111	0100	1	21.367	0.208	21.161

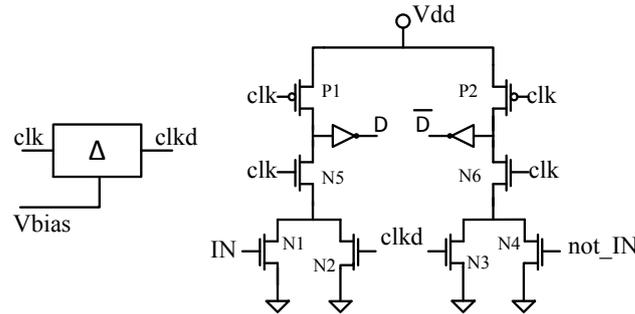


Figure 5.30. Circuit for the conversion of a CMOS signal into the TEL domain for the iDDPL style.

5.6.4 LPA attacks

From Eq 5.11 it is clear that the static power consumption of the circuit linearly depends on the sum of the Hamming weights of key and input, but is uncorrelated to the Hamming weight of the target function, irrespective of the chosen internal operation (e.g. XOR or S-Box). More in general, it is not possible to detect a sound leakage model for the circuit because TEL signals are in the invalid state when static measures are executed and there is no possibility to detect information by the correlation between actual and predicted power. Furthermore, the correlated component $I_{leak,CONV}$ is only a very low fraction of the overall static power consumption of the chip (less than 1%) and is too low to allow to exploit a sound leakage power model ($CV = 0.008\%$).

The impossibility to find a valid leakage model and predict the leakage consumption of the circuit inside the chip makes actually impossible to distinguish which part of the leakage depends on the correct key. Moreover it is possible to design LPA-resistant multiplexers in the CMOS-TEL converters so that also the leakage current $I_{leak,CONV}$ can be completely de-correlated from the key.

On the basis of these considerations, we mounted LPA attack against the iDDPL implementation, in a similar way as done for the other DPLs. In Fig. 5.31 the results of the attacks against the output of the S-Box are shown. LPA was still performed with an on-chip noise with a variance $\sigma_N^2 = 100 \cdot \sigma_S^2$, and at a temperature equal to $50^\circ C$. The plot of the correlation coefficients in Fig. 5.31 confirms that LPA attack is not effective in the iDDPL test chip, leading to high correlation coefficients for the wrong key guesses, whereas the correlation coefficient of the correct one is almost zero, revealing that no correlation exists between key and the measured overall static consumption in our implementation. LPA attacks were also mounted against a different selection function: the output of the XOR logic, which is a bit-slice block. Nevertheless (see Fig. 5.32) LPA is still unsuccessful and leads to the wrong key guess $(0)_2$.

We would like to point out that the results obtained for iDDPL do not depend on the statistical distinguisher. In this simulations the Pearson's correlation coefficient has been used, but the point is that there is no physical correlation between correct key and physical emission. In the DPLs the precharge and the evaluation phases alternate within the clock semi-periods, thus when the clock signal is stopped

DPL behaves similarly to the static logic. Instead in iDDPL no correlation exists between input data, hardwired key and static power, because the information is encoded during the dynamic time interval δ and it is lost when the steady state is reached. Therefore by using this measurement setup to evaluate the leakage static consumption of the test chips, the information leakage has already elapsed after the dynamic time interval δ .

5.7 Conclusion

The topic of this chapter has been the investigation of Leakage Power Analysis attacks against cryptographic devices in submicron circuits. These attacks are becoming very important in the cryptography community because with the technology scaling the leakage of the devices (i.e. the power consumption when no signals are processed in the circuit) is increasing constantly. Actually, static power measurements of a cryptographic circuit are characterized by several practical difficulties, concerning noise, time, and measure setup, and this makes LPA attacks not trivial in practice.

The simulation results presented in this chapter demonstrate that the LPA procedure defined in [5] for simple case study circuits with an intrinsic bit-slice structure (e.g. flip-flops, registers, bus, etc.) can be extended for more complex hardware implementations, composed with a mixture of bit-slice and not-bit-slice circuits (e.g. S-Boxes) under some assumptions, which must be verified through a novel leakage model and a statistical analysis. The trend of modern cryptography is to design ever smaller cryptographic primitives in order to optimize hardware sources. Reducing the dimensions of the circuits has two important consequences: on a physical point of view, there is an increase of the leakage consumption of the device which is strongly related to the processed data; on the other side, the complexity of the circuits also decreases and an entire processing unit, as the case of the bit-slice Serpent encryption processor analyzed in this chapter, is more vulnerable to linear PAAs, as for example LPA attacks.

Results of LPA attacks demonstrate that the transistor level and logic level countermeasures (DPL and masking), developed to overcome DPA/CPA in cryptographic circuits and implemented to balance the instantaneous power consumption, fail at balancing the static power consumption, which keeps on exhibiting a dependence on the data input. The case study cryptographic circuit has been implemented using different DPL styles and has been successfully attacked using the linear LPA procedure. After having provided a model of the on-chip noise, which must be also taken into account to estimate the amount of physical leakage and at the same time validate the resistance of the device, in all the aforementioned DPL implementation the key has been recovered with a maximum number of input plaintexts equal to 150k. We used actual security metrics introduced for DPA/CPA to assess the LPA resistance of each implementation, such as the Point Biserial Correlation (PBC) coefficient and the minimum number of Measurements for Disclosure the key (MTD). The MTD for LPA attacks is very low for the DPLs under analysis: MTD ranges from ≈ 16.000 for CMOS, up to ≈ 50.000 for MDPL, when a SNR equal to -30dB is considered. Obviously, the number of traces increases in practical measures, where

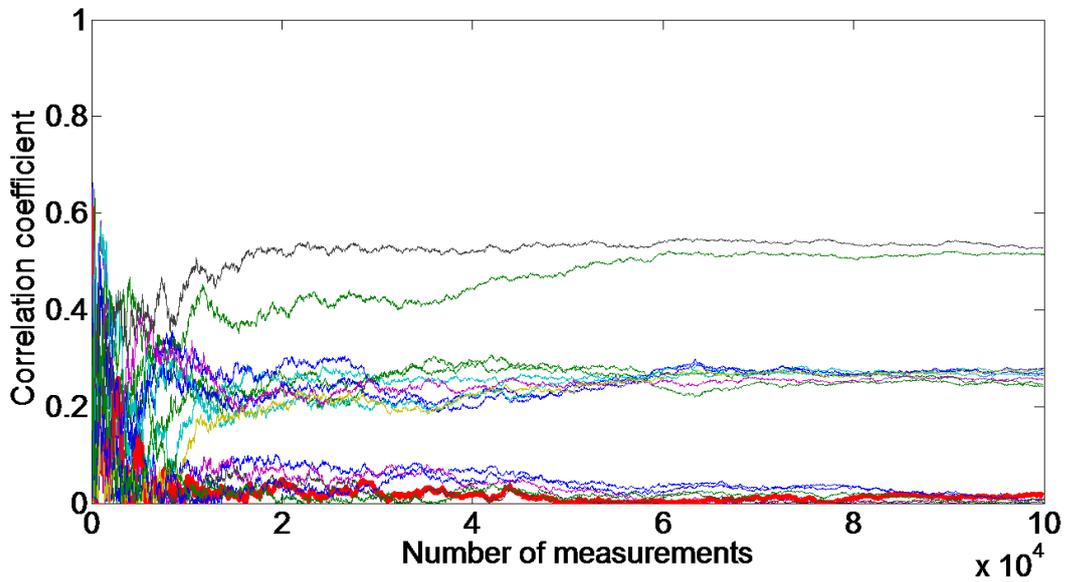


Figure 5.31. Correlation coefficients as a function of the number of measurements for DDPL (the curve of the correct key is in red).

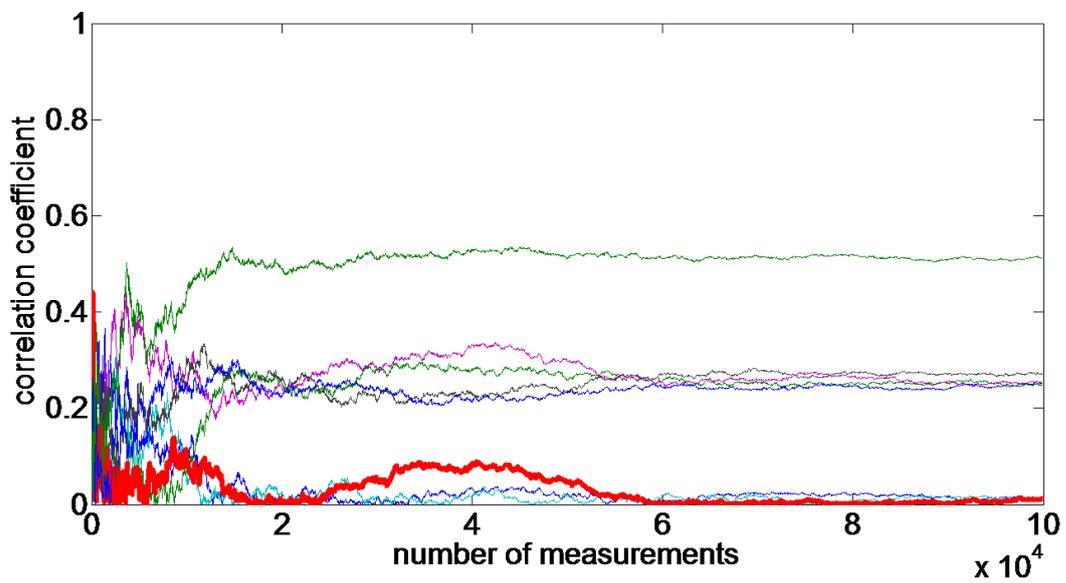


Figure 5.32. Correlation coefficients as a function of the number of measurements for DDPL using the output of the XOR as selection function (the curve of the correct key is in red).

noise is much higher and several additional noise sources are present.

LPA have been also repeated by considering another potential issue which impacts leakage measures in deep submicron circuit: the physical variability due to the mismatches of the devices, which are typical in a real scenario. Simulations results after extensive Monte Carlo simulations showed that in real mismatches circuits LPA attacks can be even more effective. Results showed that intra-die variations have an influence on the outcome of LPA attacks. For standard cells based DPLs, process mismatches do not reduce the effectiveness of LPA, leading to a success rate equal to 1 for an intolerably high percentage of sample circuits (i.e. $23 \div 50\%$) when a SNR equal to -20dB has been considered. Furthermore if we consider the one-bit wrong key guess in a second order attack strategy, the percentage increases up to 70%. Instead for a full custom logic as SABL, the static power consumption is more sensitive to process variations with respect to the nominal case, mainly due to the symmetry of the logic gates; in this case the success rate is slightly lower (i.e. 17%), demonstrating that also anti-DPA full custom logic styles are potentially vulnerable to LPA attacks. As an additional contribution, we have evaluated the static power variability in the iDDPL style, presented in previous chapter as a possible TEL circuit template. A novel leakage model has been introduced for iDDPL, which still exploits the linearity of the static power in bit-slice structures. This model explains why using a dynamic data encoding as TEL helps to decorrelate the static power of the circuit from the logic properties of the processed word and the internal logic operations. As a consequence, TEL circuits, and in particular the iDDPL style which is one implementation, are a promising solution to counteract LPA attacks in submicron circuits.

In conclusion, this work demonstrates the effectiveness of LPA attacks against real cryptographic implementations, highlighting the need of reducing the side-channel due to the data-dependence of the static power. In the future, new countermeasures must be designed in order to be secure both against DPA/CPA and LPA attacks. More in general, the design of transistor and gate level countermeasures for power analysis must take into account also the need of breaking correlation between the static power, and the input data. To the best of our knowledge, no specific countermeasures against LPA have been presented in the literature and validated on case studies. Even if the anti-DPA countermeasures prevent power analysis attacks on the differential power consumption, many logic styles which are considered as "secure" against DPA can be violated using LPA, in some cases also exploiting process mismatches. This means that the body of work focused on countermeasures to DPA attacks does not solve the security issues arising with power analysis attacks and research efforts must address toward anti-LPA countermeasures. The aim of this study is then to promote the design of novel countermeasures which are secured both against DPA/CPA and LPA attacks, and the adoption of a dynamic data encoding as TEL can certainly help. Hence, in the future a significant research effort will be required to devise appropriate solutions and countermeasures against LPA that keeps the security level to the current standards, provided that different leakage models can also be investigated by eventually exploiting a non-linear dependence between input data and static power, and other statistical distinguishers can be adopted, such as *Mutual Information Analysis* [44] and *Template Attacks* [28].

Chapter 6

Practical evaluation of PAAs against cryptographic circuits

6.1 Introduction

This chapter is focused on the description of a practical activity executed during the Ph.D. course. Practical measurements represent an important activity in the context of physical security and cryptography because allow to understand which are the main issues regarding the implementation of SCAa, thus they deserve a specific description. More specifically, in this chapter some experiments executed in the laboratory of the Ministry of Economic Development of Italy will be described for a project named *SESAMO*.

The activity in the Ministry of Economic Development has been executed in the measurement attack setup located in the Smart-Card Laboratory of the Ministry, which in this chapter will be called for simplicity with the acronym *SCLab*.

6.2 *SESAMO* project: evaluation of the hardware security of cryptographic devices

6.2.1 Objective of this activity

SESAMO was a project in collaboration between the "Istituto Superiore delle Comunicazioni e delle Tecnologie dell'Informazione" (ISCOM) of the Ministry of Economic Development of Italy and the Foundation "Ugo Bordonini" (FUB), whose purpose was the analysis of the security of mobile payments systems based on smart-cards and other cryptographic embedded devices. The project consisted of several activities, ranging from the analysis of the state of the art of the field of mobile payments up to the setup of an embryonic laboratory where mounting implementation attacks on real devices. More specifically, among them a very important role is played by SCAs, which can seriously put on risk the security of a smart-card and any other cryptographic device. Therefore, a relevant issue faced in the context of this project was to determine to what extent the most common hardware devices are secure and can be really adopted to support novel applications in the field of mobile payment, for example using novel technologies as RFID and

NFC. Further information about SESAMO can be found for example in [54].

The purpose of the activity presented in this section has been the implementation of a measurements setup to mount PAAs, which among SCAs represent the most effective and dangerous threat for the security of such cryptographic devices. as widely described in previous chapters. Basically the experimental activity consisted of two main phases: first, a measurement workstation to execute PAAs has been set up, using the devices found in the Smart-Card laboratory; then, some practical experiments have been conducted on a target cryptographic device as case study, with the purpose of evaluating the effectiveness and the potentialities of the workstation for further applications, and at the same time assessing the level of security of some crypto-cores pre-designed in our department.

At the end of the activity, the measurement workstation is assembled and correctly operating in the Smart-Card laboratory in order to be used for future activities.

6.2.2 Description of the measurement setup *SCLab*

Setup of a measurement workstation for PAAs

The first step has been the definition of the characteristics of *SCLab*, in accordance to the specifications of a standard PAAs measurement workstation, with the constraints on the sources and the budget available for the activity. Under the perspective of building a prototype workstation with a medium cost budget (in the order of some tenths of thousands of euros), we assumed that the PAAs workstation must be able of performing at least the most known attack methodologies: DPA and CPA attacks.

Basically, a standard measurement setup for PAAs [72] is composed of a device under test (DUA), on which perform the measurements on the physical emissions, some devices to provide the I/O signals and the voltage supply, an oscilloscope for the data-acquisition provided with a suitable measurement circuit and a probe, and a post-processing calculator to collect and post-process the sampled data.

A general description of the block diagram of a standard PAAs measurement setup is depicted in Fig 6.1.

In figure, the principal blocks of the setup are indicated, together with the interconnections between the components. More specifically, the arrows indicate the direction of the data flowing between two blocks, according to the order defined by the numbers. In the first phase (1), the DUA is turned on by providing the power supply voltage through a measurement circuit and the clock for the digital logic inside. Then (2), the oscilloscope is initialized by the PC. The following step is the management of the I/O flow between PC and DUA, which is executed through an I/O interface: the PC sends the data into the interface, which in turns sends them into the DUA (4); at the same time, the interface sends the trigger signal to the oscilloscope, so that all the operations are perfectly synchronized. During the elaboration, the current traces are acquired by the oscilloscope (5). Finally, the samples are sent from the oscilloscope to the PC where are stored (6).

Once defined the principal devices and their connection, the following step has been the choice of the cryptographic device. As it will be clear in the following, the choice of a specific DUA poses a constraint on the choice of the other components and

on the way to connect them. For example, in the case of a commercial smart-card, an adequately modified smart-card reader must be used as IO interface; furthermore, also the design of the measurement circuit strongly depends on the chosen DUA.

Description of the cryptographic device under test

As described in the introduction to the *SESAMO* project, the purpose of this activity has been to build up a measurement workstation for PAAs against cryptographic hardware. The block diagram depicted in Fig 6.1 refers to a standard PAA workstation, therefore once the setup has been built, it can be easily adopted with slight modifications for any cryptographic device.

In our activity we have decided to use a specific cryptographic device for the workstation *SCLab*: a FPGA board of our department. The reason for using a FPGA is that it offers a great flexibility for testing different cryptographic cores, which would not have been possible with a specific smart-card; furthermore, we have chosen a known architecture because it is already provided with an interface on-board circuitry as well as power measurements test-points, given that it has been already used for cryptographic applications in several experiments executed in our laboratory in last years [77] [127].

Indeed, the FPGA board was developed several years ago in our department with the purpose of attacking software crypto-implementations on microcontrollers, in the context of an European project called SCARD [77]. The board was designed with multiple sockets under the perspective of being reused for future applications. The main parts of the board are:

- A Cypress EZ-USB FX2 USB microcontroller (CY7C68013).
- A Microchip serial EEPROM (24LC64).
- An Altera Cyclone FPGA (EP1C12Q240C6).
- A clock generator IC also from Cypress (CY22393FC).
- A USB Interface (implemented by the Cypress microcontroller).
- A RS232 Interface (the MAX 3232 level shifter).

The integrated circuit Cypress EZ-USB FX2 USB microcontroller integrates a USB 2.0 transceiver, a serial interface circuitry, an enhanced 8051 microcontroller, and a programmable peripheral interface. This device provides a data transfer rates of 56 MBytes per second, that is the maximum allowable USB 2.0 bandwidth. The microcontroller supports also I^2C -compatible bus as a master, but only at 100/400 kbps. The pin *SCL* and *SDA* have open-drain outputs and hysteresis inputs. The address and data buses of the 8051 are mapped both on the FPGA IO pins and connectors in order to allow memory capacity enhancement. The firmware is stored in a standard 64K I^2C CMOS Serial EEPROM (24LC64) of the microcontroller, and can be updated via I^2C using the USB Control Panel or by means of a standard programmer, however it is possible to download the firmware from the PC via USB, doing without the flash memory. The Altera Cyclone EP1C12 FPGA placed on the board offers 12.060 logic elements, 52 4K RAM blocks and 2 phase-locked loops

(PLLs) which can be used to customize the measurement execution and generate clock signals. The FPGA can be re-configured via the USB connection. The digital test board has connectors in which are mapped more than 150 IO pins of the FPGA.

The auxiliary clock generator IC (Cypress CY22393) implements fractional PLLs and allows to provide low jitter programmable clock signals with high output-driving capability. In particular this IC provides three integrated PLLs, ultra-wide divide counters (8-bit Q, 11-bit P, and 7-bit post divide), improved linear crystal load capacitors and flash programming both by external programmers and via I2C. The power supply of all components of the card, with the exception of the FPGA chip, is supplied directly from the PC, via the USB port while the FPGA is powered by a separate power supply line, with a voltage of 1.5 volt. The latter is just the externally accessible power supply line where the current probe has been connected in order to measure the adsorbed current.

For the experiments, the serial interface was implemented directly in the FPGA, with an external adapter, based on the MAX 3232 level shifter, mounted on a separate strip-board and connected to two IO pins, instead of using the serial interface implemented by the Cypress microcontroller.

A final remark must be done in regard of the practical implementation of the experiments. In order to improve the resolution of the measured traces, we have decided to disconnect the decoupling capacitors from the FPGA power supply network. It must be pointed out that removing on-chip capacitors doesn't invalidate the concept of "non-invasive" attack, because this principle is in accordance to the concept of "reversibility of tampering"; namely, after the experiments the on-chip capacitors can be reconnected without leaving traces of tampering, and the operation is completely reversible. The adversary model is in accordance to the situation depicted in left part of Fig. 2.1, where the attacker removes the off-chip capacitance in order to perform attacks in the best conditions. This situation is also in accordance to the case of attack against a smart-card, where in practical application no decoupling capacitors are considered, being the chip only connected to a read-write apparatus (i.e. a card reader) via the contacts of which is equipped. Therefore, this setup could be also re-arranged using a smart-card reader modified by an attacker. On this basis, the FPGA board perfectly simulates a smart-card under attack.

The board is depicted in Fig. 6.2. Note that the flash memory chip is not present on the board, thus the firmware must be loaded through a Cypress software (EZ-USB control panel) running on the PC before loading the configuration code on the FPGA. The latter FPGA has a volatile memory, therefore it is necessary to load the code every time it is turned on.

Further information about the board can be found in [77] and [127].

Description of the devices found in the laboratory

The measurement workstation *SCLab* has been implemented in the laboratory of the Ministry of Economic Development. In the laboratory, the following devices were available:

- A digital oscilloscope Tektronix DPO5104.

- A BNC adapter for the oscilloscope Tektronix DPO5104 (standard Tektronix proprietary).
- A digital multimeter Tektronix DMM4050.
- A digital multimeter TTI 1906-GP.
- A power supply with 4 outputs GWINSTEK GPS-4303.
- A personal computer with Microsoft Windows 7.
- An acquisition board GAGE COB-021-000.
- A spectrum analyzer TTI PSA1301T.
- An arbitrary waveform generator TGA 12104.
- An Eprom programmer Dataman 48 pro.
- A current inductive probe Tektronix TCP202.
- A differential preamplifier ADA400A.
- A current inductive probe Tektronix CT-1 (not supplied).

Actually, with reference to the situation of standard PAAs depicted in Fig 6.1, this set of instruments is oversized for the application, and only some of them have been used: the oscilloscope, the PC, the power supply, and the probes. All the other devices found in the laboratory have been tested and studied through their data-sheet, but for the set of PAAs experiments presented in this chapter they have not been used. In general, these devices could be adopted for improved PAAs, such as for example the spectrum analyzer for PAAs in the frequency domain [42] [48].

The basic element of the setup is the oscilloscope. A digital scope is certainly the most common used device for signals acquisition. It must be provided with a port for PC connections, usually a USB port, but old devices has only the GPIB (General Purpose Interface Bus, standard IEEE 488) port, and must have a good frequency response. The Tektronix DMM4050 is a modern digital device, which can be adopted for a very large field of applications, among them also SCAs. The oscilloscope is provided with a BNC adapter Tektronix DPO5104, which allows to connect the current inductive probe. The oscilloscope has a bandwidth equal to 1GHz, which is the minimum required for PAAs, a rising time of 350ps, a vertical resolution of 8 bit, a maximum sampling frequency of 5GS/s on each channel and of 10GS/s on a single channel. The oscilloscope is also equipped with an operating system (Windows 7), and works just as a personal computer; anyway, even if it could be used as a stand-alone workstation, it is preferable that the device for acquisition and for storage of the data are separated. The connections between PC and scope are handles by a Labview script.

Under the perspective of doing current measurements, another important device is the power supply, which must supply a regulated direct voltage with the minimum ripple. Cryptographic devices work at low voltages (in the order of few volts) and require low currents (in the order of milliAmpere); therefore, the most important

property required by the power supply is a good stability of the output voltage. The GPS-4303 is a linear-mode power supply, with a very small ripple (1mV rms with a fixed voltage). It has 4 separated outputs, an adjustable output voltage in the range 0 - 30V, and an adjustable output current in the range 0 - 3A; the regulation capability is less than 0.01% both in presence of input voltage and load variations.

For what concerns the PC, a calculator with a modern operating system (Windows 7) and a high data storage capability (in the order of 1TB) is enough.

We have used two current inductive probes for the measurements setup, one found in the original equipment of the laboratory, and one of our department. They are described more in detail in next paragraphs.

Notes on the implementation of the power measurement circuit

The most critical part in the diagram of Fig 6.1 is the implementation of the measurement circuit. In literature several methods are suggested to measure the current adsorbed by a cryptographic device, which differ according to the application and poses a constraint on the design of the measurement circuit and the choice of the probe.

A first simple method is to insert a sensor resistor (typically in the order of a tenth ohms) in series between the power supply network and the power pin of the DUA, in order to measure the current through the voltage potential across it [4]. Another method is to use a specific electronic circuit which provides the DUA with the power supply and at the same time generates a signal proportional to the adsorbed current [25]. A third method is to use an inductive probe, which measures the adsorbed current by exploiting the phenomenon of electromagnetic induction. For the measurement setup *SCLab* we have just chosen an inductive probe.

Basically, an inductive probe is constituted by a coil which is wound around a toroidal core. The current flows in a conductor passing through the toroidal inductor, and the electromagnetic field arising from this current generates a current flowing in the coil (see Fig. 6.3). This method has the advantage of avoiding any voltage drop in the power supply of the cryptographic device, as happens using a series sense resistance, and guarantees a high bandwidth, typically several hundreds MHz, in order to capture the rapid variations which typically occur in the power trace. The absence of voltage drop on the voltage supply is an outstanding advantage, because some cryptographic devices are equipped with power drops detecting circuitry as a countermeasure to stop the encoding (or decoding) operations if a voltage drop is sensed.

The probe Tektronix CT-1 has very good performances which makes it suitable for PAAs applications: the bandwidth is between 25kHz and 1GHz, the rising time is 350ps, the sensibility is 5mV/mA, with an accuracy of $\pm 3\%$. A very important property is that the probe blocks the DC component of the signals, having a bandwidth greater than 25kHz; this represents an outstanding issue in the experiments.

On the contrary, the Tektronix TCP202 probe has a bandwidth of 50MHz, which is quite poor for PAAs applications; it can be used for example for mounting LPA attacks against a smart-card working at less than 4MHz, but in the case of standard

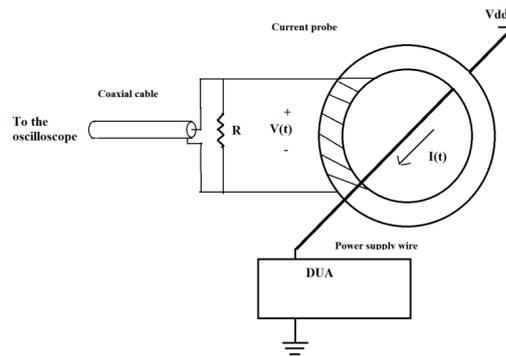


Figure 6.3. Working principle of an inductive current probe.

PAAs it must be preferred a probe with a higher bandwidth. Furthermore, the probe doesn't allow an AC coupling, therefore it represents a crucial drawback during the experiments.

The experiments presented in next paragraphs have been conducted with both the probes; the performances and usability of the probes will be described and compared in next paragraphs.

The *SCLab* workstation

After having described all the components of the measurement setup, in Fig. 6.4 the block diagram of *SCLab* is depicted. A photograph of *SCLab* is reported in Fig. 6.5. The experiments described in next paragraphs have been executed using this setup.

According to the fact that the choice of the instruments and the way how they are interconnected depend on the DUA, the board is connected to the PC with a USB port, through which the programming code is loaded, and with a serial port RS-232, which is used to exchange plaintexts and ciphertexts, as well as the start signal of the algorithm.

The setup has been used to validate the resistance of some AES encryption units, that have been synthesized on the FPGA and will be described in the following paragraphs. Note that the adoption of the FPGA allowed to implement different cryptographic cores with a great flexibility. The working principle of the AES block cipher has been already described in Chapter 1.

Description of the software procedure adopted in PAAs experiments

As a final remark, we briefly describe the software procedure to control all the phases of the PAAs. For this purpose we decided to use three different programs instead of using a stand-alone program:

1. The phase of acquisition is controlled by a Labview script.
2. The algorithm of attack is executed through a proprietary software developed in our department, *Gemini*.

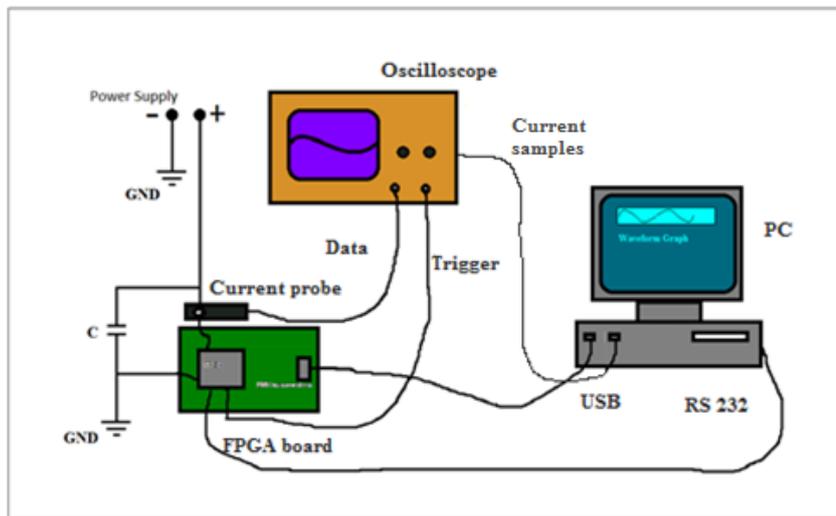


Figure 6.4. Block diagram of the *SCLab* workstation in the laboratory.

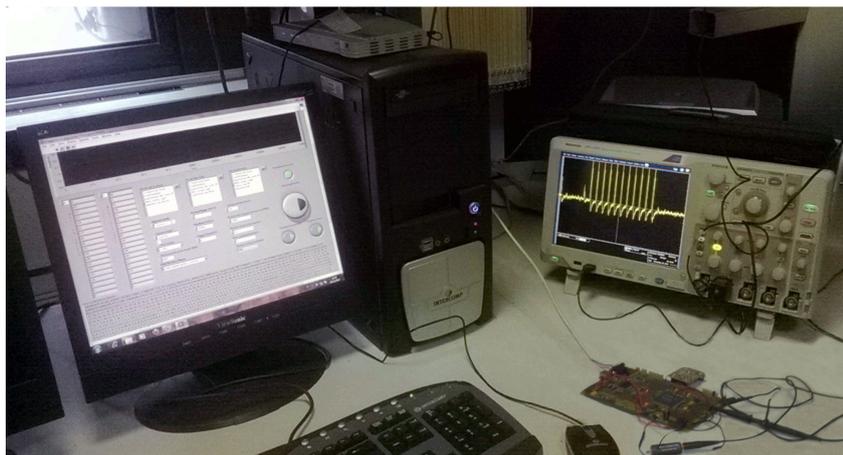


Figure 6.5. Photograph of the *SCLab* workstation in the laboratory.

3. The representation of the correlation trace is done by Matlab.

Using separated programs makes the procedure less automatic, because the user must manually sets the programs for the execution of the single phases. On the other side, this allows to reuse pre-designed software modules with very slight modifications of the scripts, therefore it represents the best choice from our perspective. Furthermore, using a stand-alone program could put on risk the reliability and the length of the experiments: for example, any error in the sampling phase would cause the failure of the attack, whereas separating the sampling and the attack phases allows to check if any error occurs in the single phases.

The program for the acquisition of the traces is written in G language in Labview environment. Labview is very intuitive and simple to be used, and is widely used to control an oscilloscope from a PC. Basically, the program executes these operations:

- Generation of the 128-bit random plaintext words to be sent to the AES encoder.
- Generation of the *.txt* files containing such plaintexts.
- Acquisition and storage of the ciphertext words from the AES encoder.
- Acquisition and storage of traces from the oscilloscope.

An example of current traces monitored by Labview is shown in Fig. 6.6. Data acquired from the oscilloscope are then sent to the PC and stored in form of *.txt* files, and each sample is represented as a signed two-digit number (the voltage value of each sample is multiplied by 1000 and then truncated); for each elaboration of the cryptographic algorithm, the vector containing the current samples, the plaintext represented in the hexadecimal format, and the encrypted data also in hexadecimal. At the end of the elaboration, if the dimension of each vector is of L samples and the number of elaboration N (with a duration T for each one), the number of stored samples is $N \cdot L$.

The attack phase is executed by a software called *Gemini*. *Gemini* is an executable file which takes as input the *.txt* files in the PC and executes the PAAs according to a specific statistical methodology. It has been developed for the European project SCARD in the context of side-channels [77]. The program offers also the possibility of using other procedures. A screenshot of the graphic user interface of the program is shown in Fig. 6.7, where all the possible options for selecting the statistical procedure of PAAs, as well as the target function of the AES core and the power model, are depicted.

The program is independent from the architecture of the algorithm, therefore it can be used to evaluate different AES implementations. An in-depth description of the working principle of *Gemini* can be found for example in [127]. In our experiments we have used a Correlation Power Analysis attack strategy based on the Hamming weight model, and we have used as target function of the attack the data word at the output of the *Substitution* phase of round 1 or the data word at the output of the *AddRoundKey* phase of round 0. The choice will be specified for each experiment.

Gemini stores the correlation coefficient traces for all the key guesses in a specific

file, which can be opened by any graphic processor. For this purpose, we used Matlab and produced a specific script for elaborating the traces. An example of correlation trace for a successful attack is shown in Fig. 6.8. The correlation curve associated to a byte of the correct key is shown in green; similar curves will be plotted for each byte of the key.

6.2.3 Description of the cryptographic cores used in the experiments

AES cores

In this set of experiments we have tested three different architectures. The first architecture is the basic AES encryption unit already described in Chapter 4, and depicted in Fig. 4.21 (AES-0) [75]. The second architecture is the XOR-series countermeasured core already described in Chapter 4, and depicted in Fig. 4.24 (AES-3) [78] [79]. The security of both these architectures have been also evaluated in the workstation *SCLab* before being inserted in the SERPAES chip.

The third architecture has not been inserted in the SERPAES-chip. We will refer to this core as AES-5 [108]. This core implements a different data-path for the AES encoder. Similarly to AES-0, the core AES-5 is based on an iterative structure; but unlike AES-0, there is only one combinational logic implementing all the layers of the algorithm, and only one accumulator register at the output of the combinational logic. Thus, the data-path is not based on an inner-round pipeline as AES-0 and takes one clock cycle for processing one round.

Basically the architecture is a two levels pipeline: in the first clock cycle, the plaintext is combined to the round key through the XOR gates, which implement the *AddRoundKey* of round 0, and the datum is stored in a register; then, the combinational logic takes 10 clock cycles for processing all the rounds. Considering also the first two clock cycles, the encoder takes 12 clock cycles for encrypting a 128 data block, which corresponds to a throughput higher than AES-0.

The cryptographic cores are reported in Fig. 6.9, 6.10, and 6.11, respectively.

Notes on the implementation on the FPGA

The aforementioned AES cores have been synthesized using the Altera Quartus II software, starting from their VHDL descriptions, and loaded on the Altera Cyclone FPGA. At the beginning of this activity, the cores have been already designed and evaluated in our department with a less advanced attack workstation [127]. In the FPGA, but outside the coprocessor itself, some additional functional blocks were added; these blocks have the purpose of managing the IO operations as well as the start and trigger signals. More specifically, the following components are synthesized on the FPGA:

- A control unit, in the form of a finite state machine, for managing IO operations. This unit also generates both the start signal for the cryptographic coprocessor and the trigger signal for the oscilloscope.
- An asynchronous serial communication interface according to the RS232 standard.

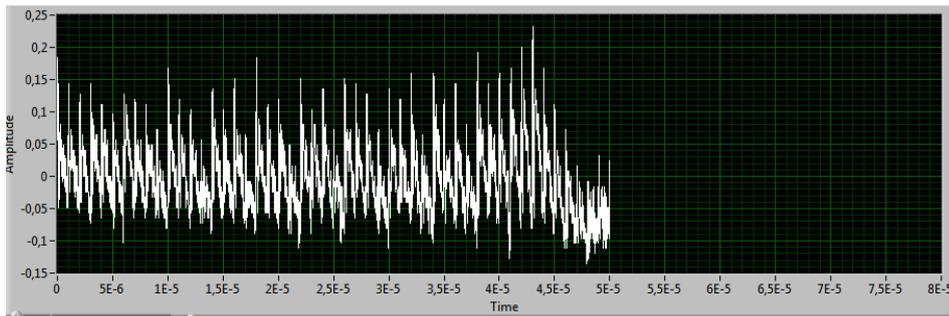


Figure 6.6. Current trace of the FPGA correctly displayed in Labview.

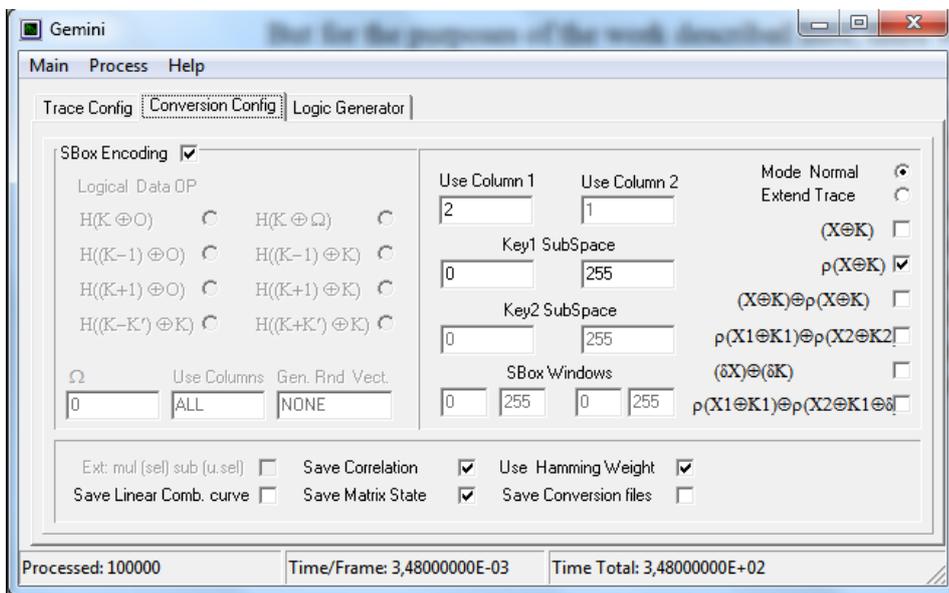


Figure 6.7. Screenshot of the GUI of Gemini for PAAs against an AES core.

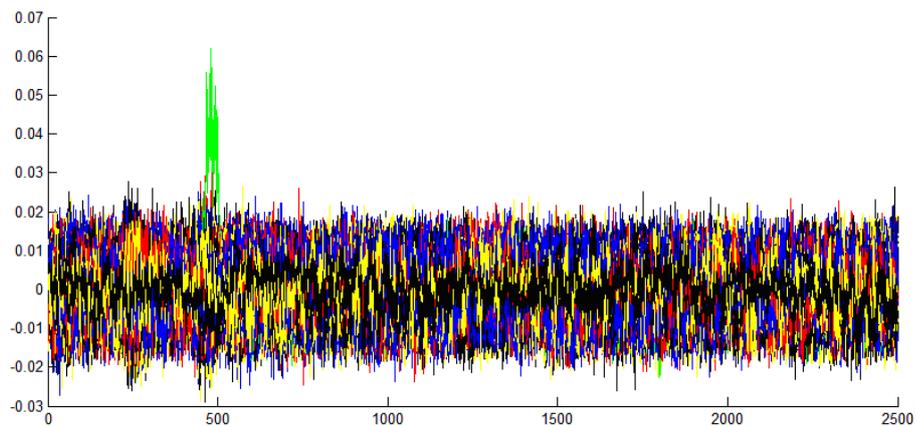


Figure 6.8. Correlation coefficient curves displayed by Matlab for a single byte of the key (case of successful attack).

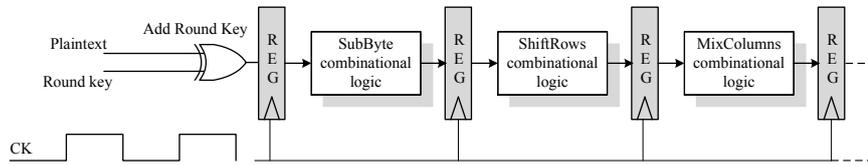


Figure 6.9. Block diagram and synchronization signals of AES-0.

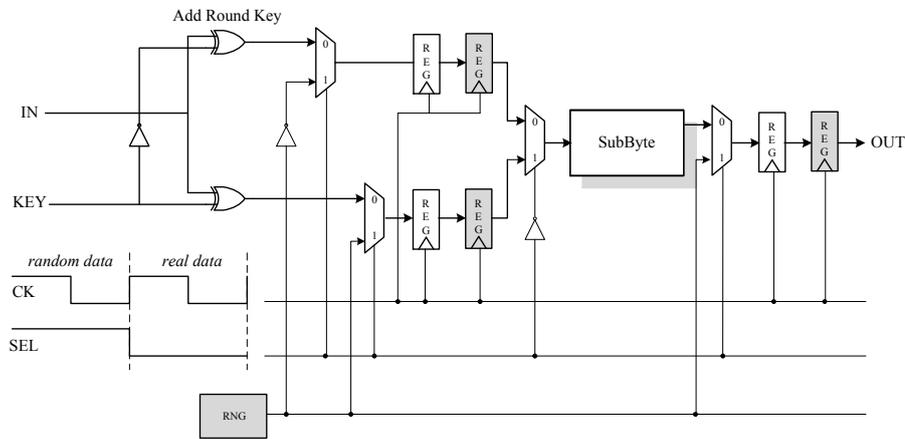


Figure 6.10. Block diagram and synchronization signals of AES-3.

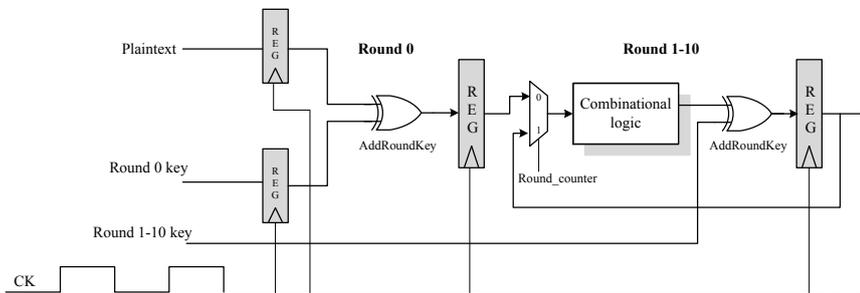


Figure 6.11. Block diagram and synchronization signals of AES-5.

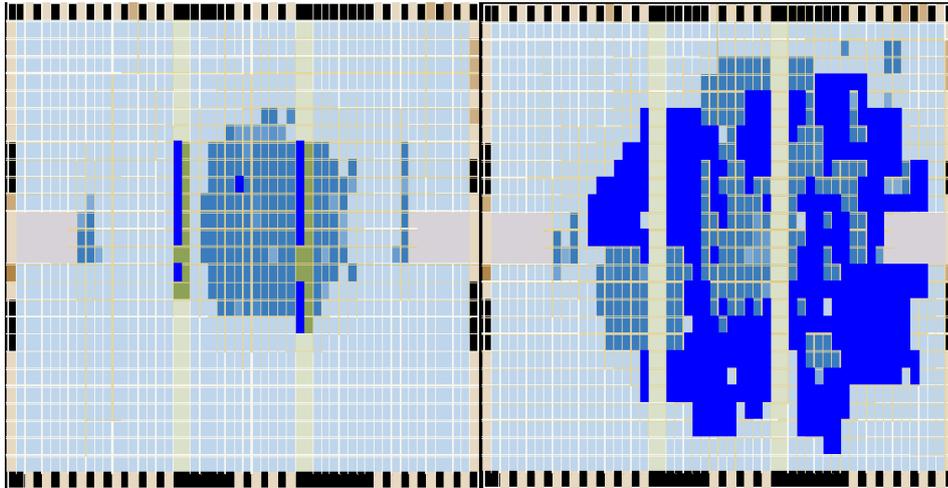


Figure 6.12. Floorplan of AES-0 on the FPGA, in the case of synthesis with RAM cells (left) and without RAM cells (right).

These functional blocks may vary according to the synthesized AES core, since each architecture has its own peculiarities regarding IO bus, control signals, and so on. More in general, the synthesis varies according to the settings of Quartus. These settings may impact timing (e.g. propagation times, maximum clock frequency, setup and hold time) and the usage of hardware sources of the FPGA. The original version of the asynchronous serial interface has been found on the internet as VHDL code [51] and modified to fit the needs of this particular application. Further details about control unit and interface can be found in [127].

In regard of the timing of the circuits, each experiment has been executed by using the same settings. Experiments differ according to the hardware sources used by the synthesis: more specifically, a synthesized core may use the RAM cells on the FPGA or not. Using RAM cells allows to reduce the area occupation with respect to a full combinational synthesis, as in the floorplans shown in Fig. 6.12.

According to Fig. 6.12 on the left, the floorplan refers to the synthesis of AES-0 executed using RAM cells; the S-Boxes of the *SubstitutionBox*, depicted in blue in figure, are almost entirely implemented using RAM cells. On the contrary, the floorplan on the right is synthesized without using RAM cells, leading to a bigger area occupation. It must be pointed out that this may impact the security of the core, as it will be shown in next paragraphs. Experiments on AES-0 and AES-3 will be repeated in both cases, in order to understand in which way the security of the core is affected.

6.2.4 Description of the experiments

General considerations

In this chapter we describe the experiments that have been executed on the previously described FPGA board using the *SCLab*.

The PAAs procedure has been executed by supposing to have a full knowledge

of the architecture of the cores, which indeed have been widely studied in last years in our department. Accordingly, we already know the level of security of each core synthesized in Quartus: for example, AES-0 is well known to be vulnerable for PAAs against the *SubstitutionBox*, whereas AES-5 is more vulnerable for PAAs against the *AddRoundKey* layer. The core AES-3, which has been implemented for enhancing the security level of AES-0 both on the *SubstitutionBox* layer as well as the *AddRoundKey*, has been synthesized and evaluated with two different FPGA synthesis, one with and one without RAM cells.

As it will be shown in the following paragraphs, even if the target function is an operation of round 0 or round 1 and the correlation peaks should be detected in correspondence to the first clock cycles of the elaboration, in some cases the peaks can be detected in correspondence to the last part of the elaboration. This depends on the specific implementation on the core: for instance, in the pipeline of AES-0 the plaintext block is re-sent as input and stored in the input register until the following block is loaded, with a latency of several clock cycles. This property has been kept also in the design of AES-3.

As a final remark, we provide some details about the experiments. In the attacks we have used as target function the output word of the *SubstitutionBox* of round 1, or the output word of the *AddRoundKey* of round 0. The success (or the failure) of the attacks depends on the number of acquired traces, and therefore on the time of elaboration. In the experiments, we have considered an acquisition time of maximum 48 hours, which leads to a number of acquired traces equal to 1.2M, which represents the maximum capacity of the attacks. With *SCLab* we managed to obtain a constant acquisition speed of 10 traces per second for the first 800k, for an overall duration of 23 hours of acquisition, thanks to the very good performances of the oscilloscope. After this values, the acquisition starts to slow down, mainly due to the high number of data stored in the PC.

For each PAA experiment the correlation coefficient plot of the key guesses for each byte of the key is reported, from the most significant bit (byte 1) to the least significant (byte 16), according to the convention adopted by software Gemini. In the experiments, we have used a fixed value for the key, which is equal to the following hexadecimal string: '9DAC5F7D56E8BE7B655A48912120BA77'. In the correlation coefficient curves, the correct key will be shown in green, and the successful or unsuccessful of the attack is assessed according to the amplitude of the correlation coefficient (visual inspection).

Finally, we point out that the experiments have been executed using the two inductive probes of the laboratory, in order to have a fair comparison between them. Experiments from 1 to 6 have been done with the probe Tektronix CT-1, whereas experiments 7 and 8 with the probe Tektronix TCP-202, using lower clock frequencies (down to 100kHz). We point out that smart-cards work at frequencies in the order of MHz (usually 3.57 or 4.91 MHz), therefore the working frequency during the elaborations of the FPGA, typically equal to 1MHz, is compatible with this specification.

Experiment $n^{\circ}1$

Experiment 1 has been executed on AES-0. The synthesis of AES-0 in the FPGA has been done mostly using the RAM cells of the device, obtaining the floorplan shown in the left part of Fig. 6.12. Synthesis is automatically performed by Quartus, then the choice of the blocks of the data-path to be synthesized with RAM cells is not done by the user. The clock frequency of the elaboration is set to 1MHz, and the number of samples for each trace is equal to 2.5k.

The number of acquired traces is equal to 750k, with a total duration of the acquisition of about 22 hours. The results of the attacks against the *SubstitutionBox* of round 1 are shown in Fig. 6.13 for a number of traces equal to 100k. The figures show that this core is strongly vulnerable on the *SubstitutionBox*, being recovered 13 bytes on the 16 of the correct key with high correlation peaks only using 100k traces. By enhancing the number of traces, also the remaining bytes 2,5, and 11 can be easily recovered, as shown in Fig. 6.14.

Instead, the core has proved to be resistant against PAAs if the output word of the *AddRoundKey* block of round 0 has been used, as expected; in this case, no correlation has been detected for the bytes of the key with 750k, as shown in Fig. 6.15 for the case of one byte; the plots of the other bytes show similar results and are not reported. Note that the correlation curves for the *AddRoundKey* are symmetric due to the linearity property of the XOR function.

Experiment $n^{\circ}2$

Experiment 2 has been executed again on the core AES-0. In this case the synthesis have been done without using RAM cells: the combinational logics have been then synthesized using the logic elements of the FPGA, obtaining the floorplan shown in the right part of Fig. 6.12. The operating frequency is equal to 1MHz, and the number of samples is 2.5k, which allows to have a fair comparison to the results of experiment 1.

The number of acquired traces is equal to 800k, which led to an acquisition duration of about 23 hours. Also in this case the attack against the *SubstitutionBox* can be considered successful, as can be seen by the plots reported in Fig. 6.16; actually, some bytes of the correct key (5 and 11) cannot be distinguished, then this can be considered as an improvement with respect to the case of a RAM cells based synthesis. Instead, attacks against the *AddRoundKey* layer of round 0 are still unsuccessful, as depicted in Fig. 6.17, where the correlation curves of a byte are shown as an example; for the other bytes the plots are similar and are not reported.

By comparing the results of experiment 1 and 2, we conclude that core AES-0 is strongly vulnerable on the S-Boxes irrespective of the synthesis. But if the data-path is synthesized using only combinational logic elements instead of the RAM cells of the FPGA, a slight improvement is obtained, and the number of traces needed to distinguish the peaks of the key must be increased up to at least 1M.

Experiment $n^{\circ}3$

After having evaluated the PAA resistance of AES-0, experiment 3 is conceived for the assessment of the resistance of the countermeasure implemented in AES-3.

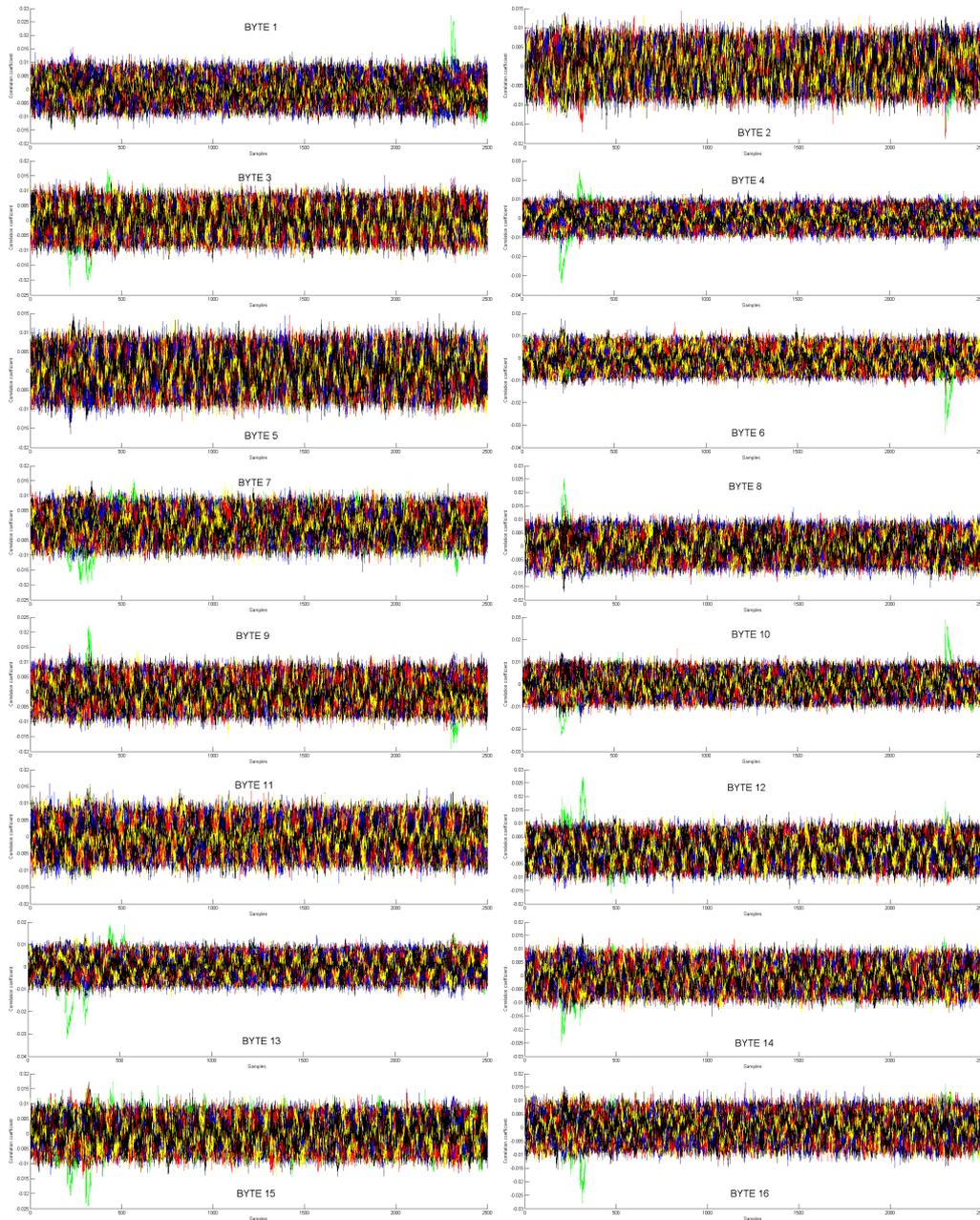


Figure 6.13. Experiment 1: correlation coefficient curves for the 16 bytes of the word at the output of the *SubstitutionBox* layer of round 1 of AES-0 (100k plaintexts).

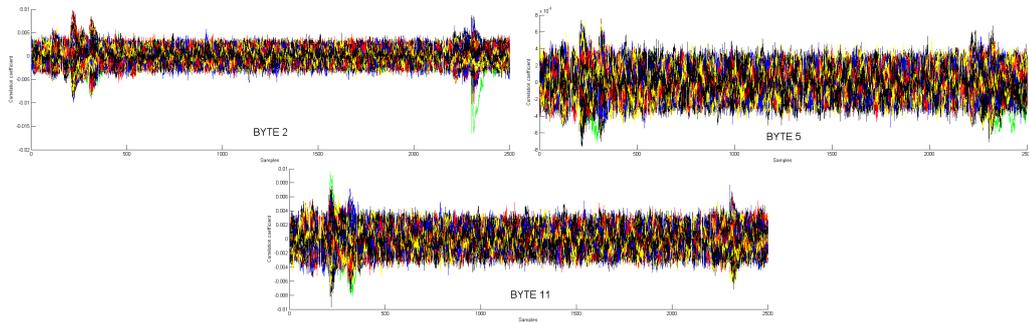


Figure 6.14. Experiment 1: correlation coefficient curves for the bytes 2, 5, 11 of the word at the output of the *SubstitutionBox* layer of round 1 of AES-0 (750k plaintexts).

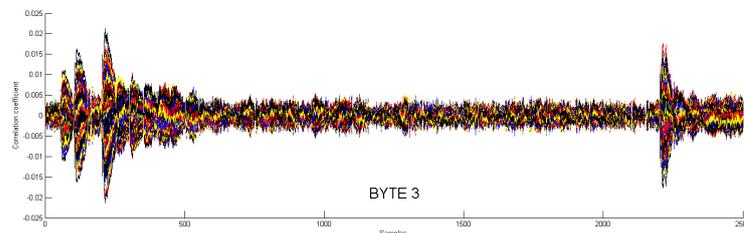


Figure 6.15. Experiment 1: correlation coefficient curves for one byte of the word at the output of the *AddRoundKey* layer of round 1 of AES-0 (750k plaintexts).

In this case the synthesis has been executed without using RAM cells, similarly to the case of the experiment 2. In order to have a fair comparison, we have used a clock frequency of 1MHz and a number of samples equal to 2.5k for each trace. We have increased the number of acquired traces up to 1.2M, with a total acquisition duration of almost 48 hours.

The attacks against the *SubstitutionBox* layer of round 1 can be considered failed, as visible in the plots shown in Fig. 6.18. Indeed, only two bytes are visible (byte 4 and 12), and a slight peak can also be detected in the curves related to byte 1. Then, we repeated the procedure on the *AddRoundKey* layer of round 0; in this case correlation can be detected for a higher number of bytes (2,3,5,6, and 15), as visible in the plots of Fig. 6.19, for other bytes (9, 10, and 11) the correlation curve of the correct key exhibits high correlation peaks.

Therefore, we can conclude that the countermeasure implemented in AES-3 improves the PAA resistance on the *SubstitutionBox* layer with respect to the case of AES-0, but we suppose that with a higher number of traces (more than 1.2M) also for this core the correct key can be completely disclosed if the *AddRoundKey* operation is attacked. With *SCLab*, the acquisition would require some days, which is still a reasonable time.

Experiment $n^{\circ}4$

Experiment 4 has been also conducted against AES-3, but with different settings during the synthesis. In this case, we have used RAM cells to implement the S-Boxes of the *SubstitutionBox* layer, similarly to the case of experiment 1 for the core AES-0.

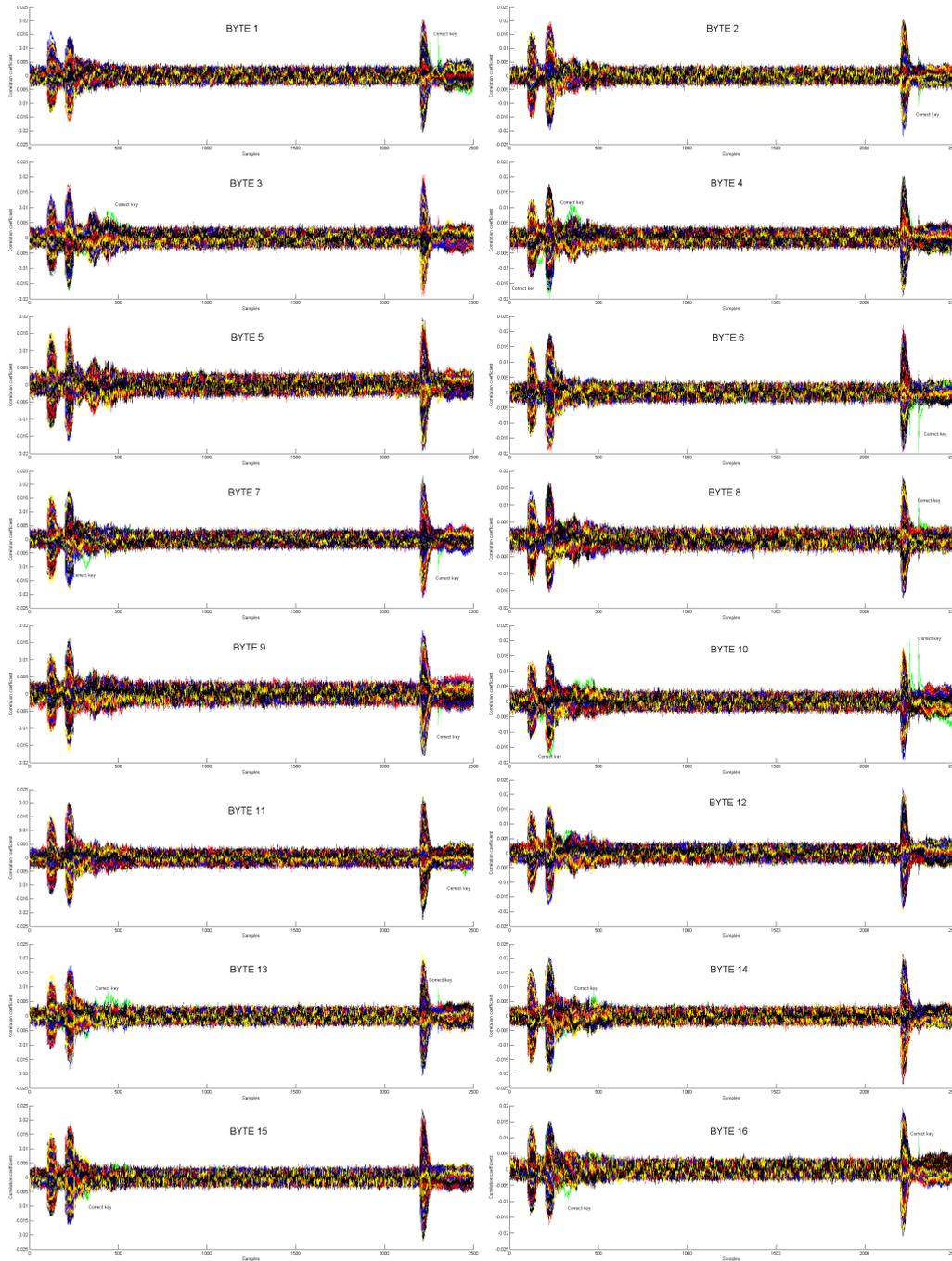


Figure 6.16. Experiment 2: correlation coefficient curves for the 16 bytes of the word at the output of the *AddRoundKey* layer of round 1 of AES-3 (800k plaintexts).

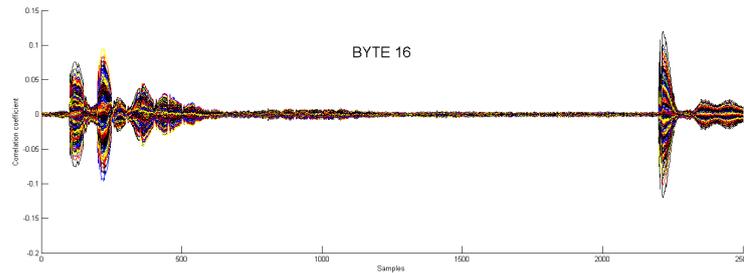


Figure 6.17. Experiment 2: correlation coefficient curves for one byte of the word at the output of the *AddRoundKey* layer of round 0 of AES-0 (800k plaintexts).

We have collected 750k traces, with a number of samples equal to 2.5k for each one. Again, we have used an operating frequency equal to 1MHz.

As already seen in the case of experiment 3, PAAs against the *SubstitutionBox* failed for each byte and this countermeasure on the S-Boxes is effective. The plot related to one byte of the key is reported in Fig 6.20. Attacks have been repeated using the *AddRoundKey* of round 0 as target function, and also in this case the countermeasure proved to be resistant (Fig. 6.21). In the figures only the plot of one byte is reported as an example; the correlation curves of the other bytes are similar and are not reported.

The results of experiments 3 and 4 proved that the countermeasure XOR series implemented in core AES-3 really increases the resistance of the basic AES unit implemented in AES-0. The strength of this architecture has been validated for different synthesis options, with and without RAM cells, and attacking two different parts of the data-path (*AddRoundKey* and *SubstitutionBox*). The core implemented with RAM cells proved to be more resistant, since PAAs with a maximum number of traces equal to 1.2M failed both on the *AddRoundKey* and on the *SubstitutionBox*.

Experiment $n^{\circ}5$

Experiment 5 has been conducted against the core AES-5. Also in this case we chose an operating frequency equal to 1MHz. In this case, we slightly reduced the number of samples for trace (2k samples). The aim of this experiment has been to prove the vulnerability of AES-5 against PAAs on the *AddRoundKey* layer of round 0. For this purpose, only a limited number of traces has been collected, about 30k. Despite that, the PAAs were successful for all the bytes of the key, as depicted in Fig. 6.22.

The reason for this strong vulnerability of the *AddRoundKey* block is due to the particular data-path of the core: indeed, the XOR operation of round 0 is executed before any other elaboration, when all the other logics are not switching; thus, the switching noise is very low and the PAAs are effective. This aspect is highlighted by the current trace of one elaboration cycle, shown in Fig. 6.23, where it is visible that combinational logics of round 1 to 10 start to process after $5\mu s$ from the beginning of the encoding. The current adsorption before $5\mu s$ depends only on the XOR gates of round 0, as it can be clearly seen by superimposing the current pattern to the correlation curves.

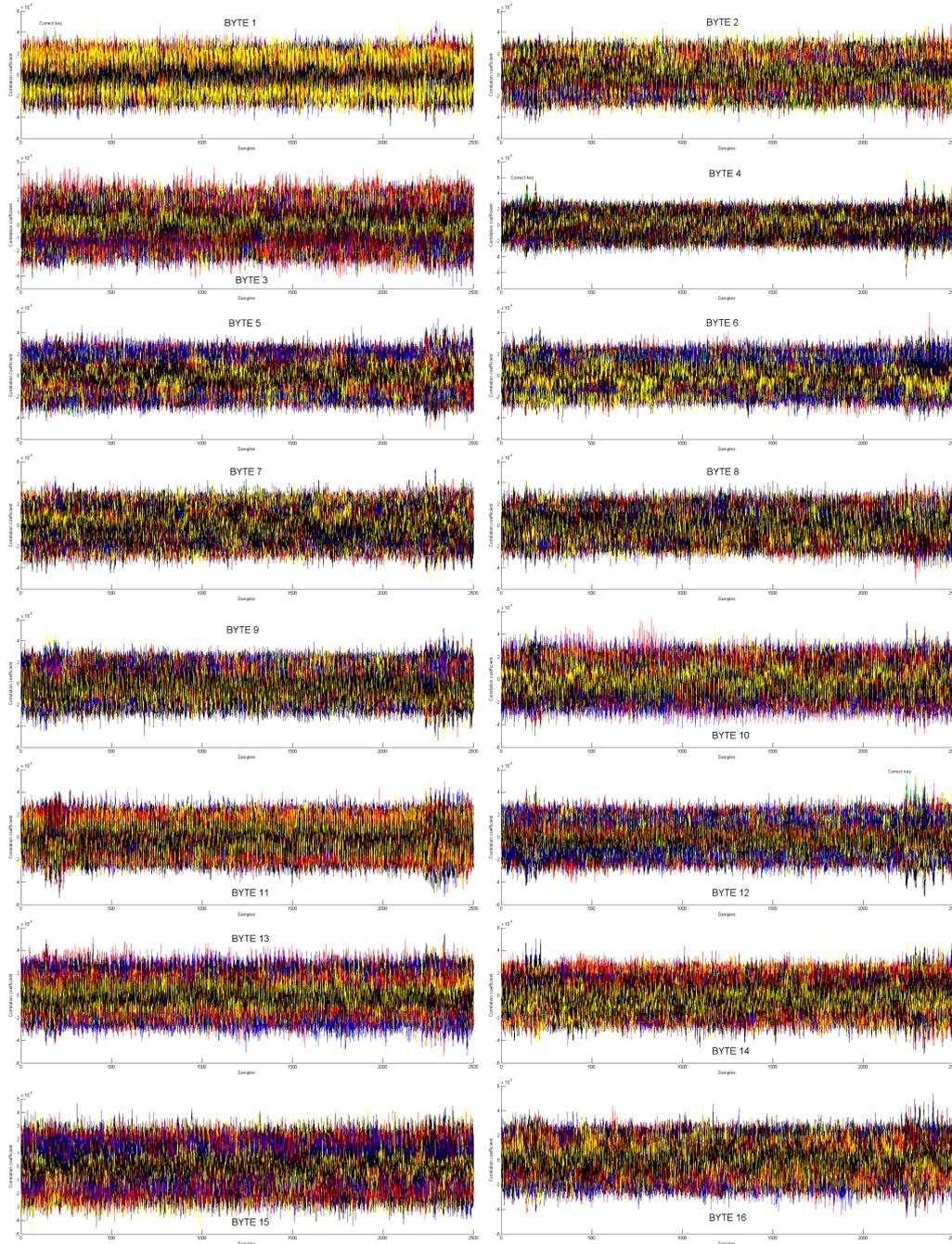


Figure 6.18. Experiment 3: correlation coefficient curves for the 16 bytes of the word at the output of the *SubstitutionBox* layer of round 1 of AES-3 (1.2M plaintexts).

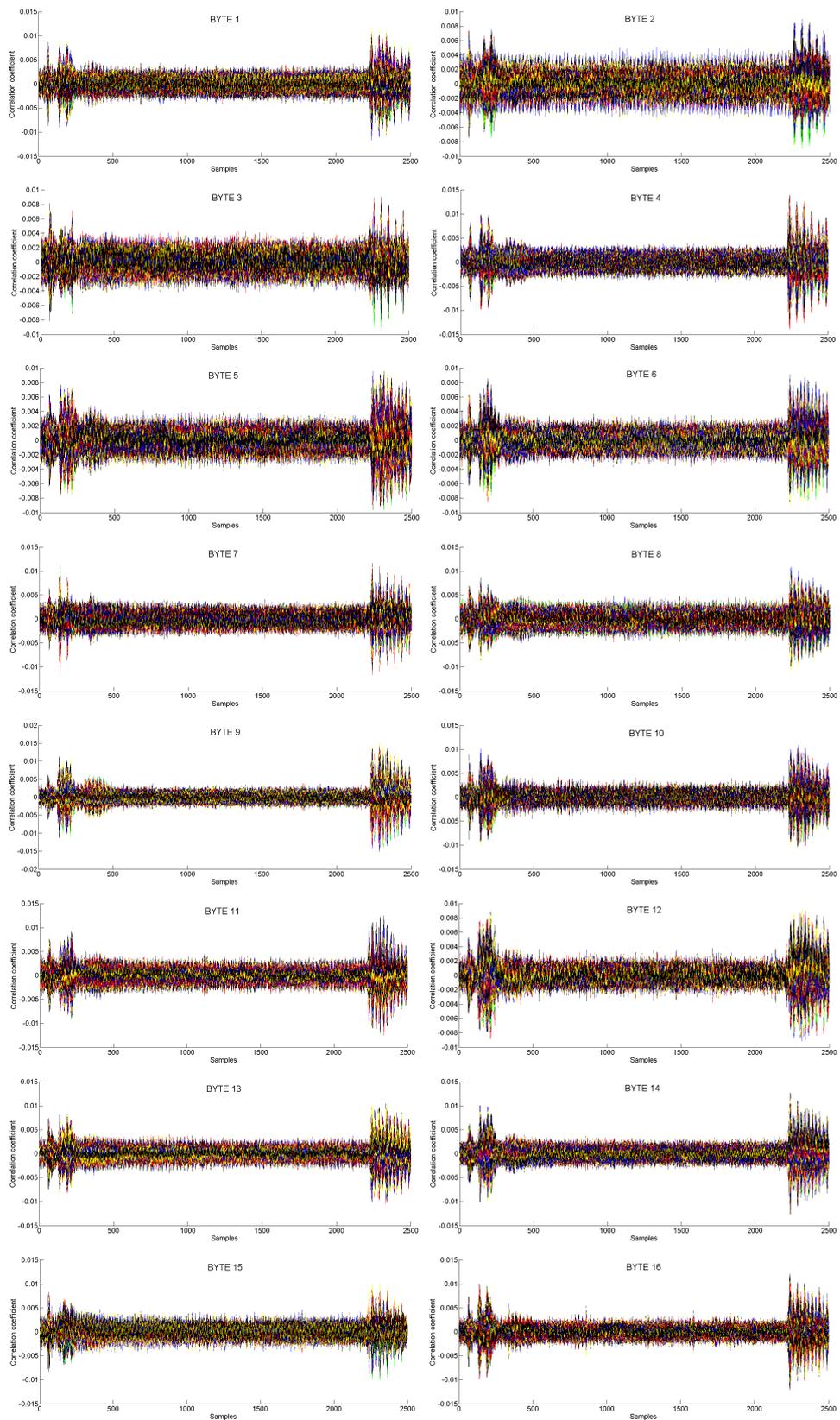


Figure 6.19. Experiment 3: correlation coefficient curves for the 16 bytes of the word at the output of the *AddRoundKey* layer of round 1 of AES-3 (1.2M plaintexts).

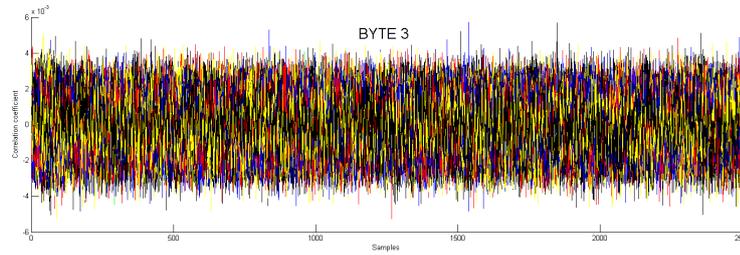


Figure 6.20. Experiment 4: correlation coefficient curves for one byte of the word at the output of the *SubstitutionBox* layer of round 1 of AES-3 (750k plaintexts).

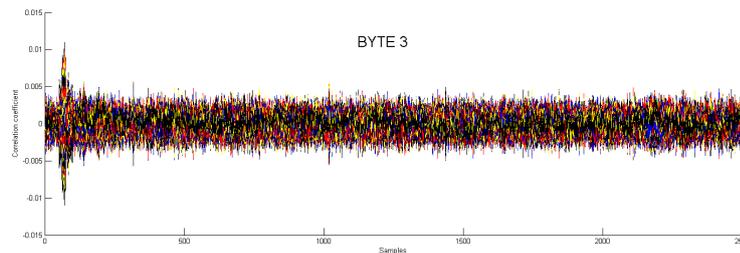


Figure 6.21. Experiment 4: correlation coefficient curves for one byte of the word at the output of the *AddRoundKey* layer of round 0 of AES-3 (750k plaintexts).

PAA has been repeated against the *SubstitutionBox*, but given the low number of traces no correlation has been detected (Fig. 6.24 for the case of a byte).

From the results highlighted by these experiments we deduce that core AES-5 is strongly vulnerable to PAAs, in particular on the *AddRoundKey* layer of round 0. Therefore the data-path of this core should be adequately protected.

Experiment $n^{\circ}6$

The last experiment using the probe Tektronix CT-1 has been again executed against core AES-5. In this case, we have decided to use an operating frequency of 100kHz, with a number of samples for trace equal to 10k. Once we have defined the level of vulnerability of the core, as shown in experiment 5, the reason for using this low frequency is to compare the performances of the probes CT-1 and TCP202 on a vulnerable core, as it will be done in next experiments. For this experiment, the number of acquired traces is 1M.

If PAAs failed when the *SubstitutionBox* layer is attacked (Fig. 6.25) in agreement with the results of experiment 5, on the contrary the core confirmed to be vulnerable on the *AddRoundKey* operation also with a lower clock frequency, as visible in Fig. 6.26 for the case of one byte reported as an example; note that at least 100k traces must be used to detect the correct key, which is more than the case of experiment 5, where only 30k were used for successful PAAs.

Contrary to what expected, attacking a core operating at frequencies lower than 1MHz (in the range of 100kHz), increased the number of required traces. This may be due to the fact that by enhancing the clock period, the internal capacitances of the device are completely discharged before the arrival of the data, therefore during the sampling phase several noise samples are stored in the current trace, which

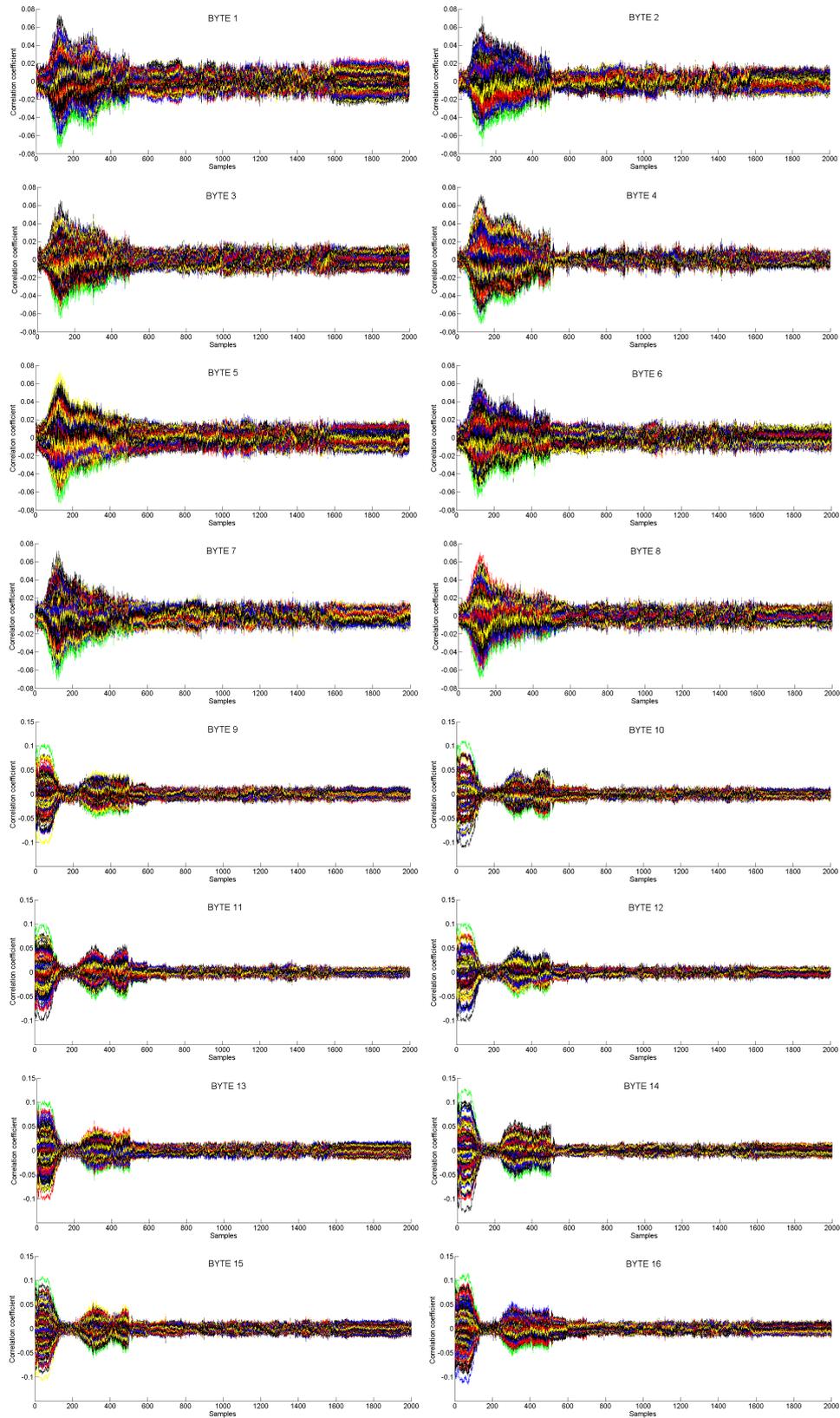


Figure 6.22. Experiment 5: correlation coefficient curves for the 16 bytes of the word at the output of the *AddRoundKey* layer of round 0 of AES-5 (30k plaintexts).



Figure 6.23. Experiment 5: current trace of AES-5 during one elaboration.

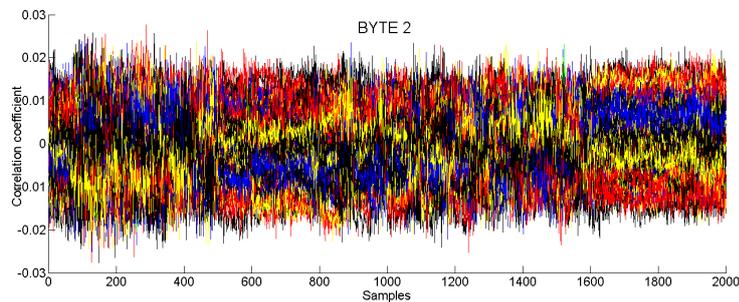


Figure 6.24. Experiment 5: correlation coefficient curves for one byte of the word at the output of the *SubstitutionBox* layer of round 1 of AES-5 (30k plaintexts).

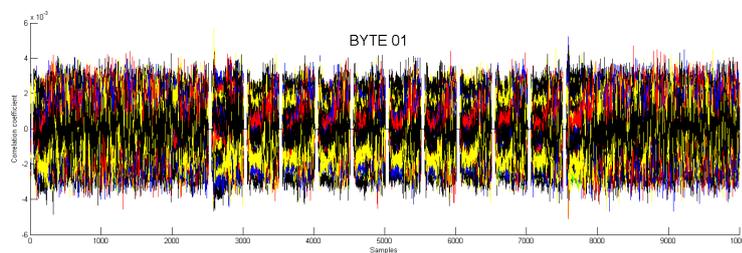


Figure 6.25. Experiment 6: correlation coefficient curves for one byte of the word at the output of the *SubstitutionByte* layer of round 1 of AES-5 (1M plaintexts).

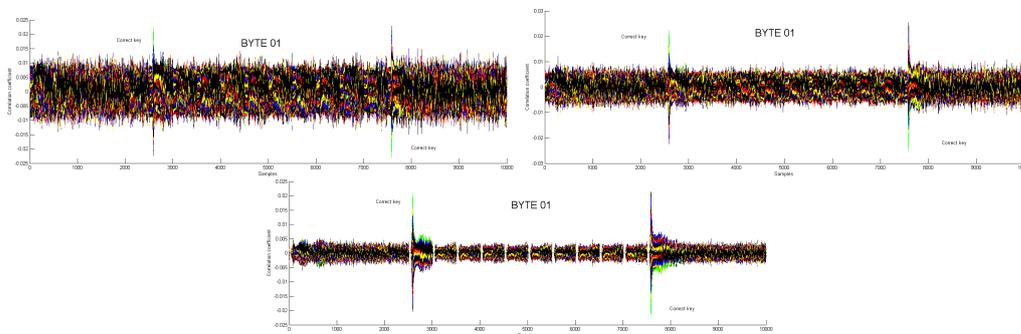


Figure 6.26. Experiment 6: correlation coefficient curves for one byte of the word at the output of the *AddRoundKey* layer of round 0 of AES-5 for 100k, 200k and 1M plaintexts.

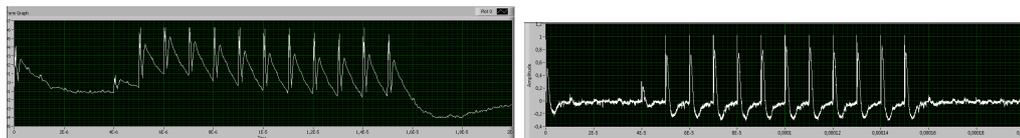


Figure 6.27. Experiment 6: current trace of AES-5 during the elaboration for a clock frequency equal to 1MHz (left) and 100kHz (right).

are intrinsically uncorrelated to the internal key. For this reason, the number of traces required to detect any peak in the correlation coefficient plot must be higher. Obviously this result depends on the FPGA; in more modern devices, we expect that leakage currents are dominant and, according to the results of Chapter 5, if the clock frequency is reduced also the data-dependence of the static power can be exploited for mounting successful PAAs.

In Fig. 6.27 the current traces for the case of clock frequency of 1MHz (experiment 5) and 100kHz (experiment 6) are shown. It is clear that during the interval when the XOR gates are expected to be very correlated to the current pattern, that is before $5\mu s$ as argued in previous paragraph, a lot of noise can be detected on the current pattern. Therefore, we had to acquire a higher number of samples in order to collect a sufficient number of relevant samples; the drawback is that also a lot of noise samples are collected, reducing the strength of the attack.

The results of experiment 5 and 6 demonstrate that the vulnerability of this core can exhibit a certain dependence with the operating frequency. In next experiments we try to attack AES-5 with the probe Tektronix TCP202, which has a limited bandwidth of 50MHz, using lower clock frequencies, as done in experiment 6.

Experiment $n^{\circ}7$

Experiment 7 has been executed on AES-5, using the probe Tektronix TCP-202. In this case we have used a clock frequency of 1MHz, similarly to experiment 5, with a number of samples equal to 2k. We have collected a number of traces equal to 1M.

PAAs have been mounted on the *AddRoundKey* layer of round 0, which has been proved to be the most critical part of the core in terms of security.

The experiments executed with the TCP202 showed to have some limitations, due to the performances of this probe. Indeed, the probe is provided with a control knob, which must be used to do the manual DC compensation during the acquisition. The reason is that for a long period of working, the setup tends to unavoidably change its operation conditions (e.g. temperature), and therefore this must be adequately compensated. Usually, oscilloscopes offer the possibility of changing the coupling settings of the connected probe, or alternatively they automatically set the AC or the DC coupling according to the probe properties (e.g. the bandwidth). Indeed, in the case of the TCP202, the coupling is automatically set to DC as default by the scope, without any possibility of setting the AC coupling option. Therefore, a user should be present during the entire phase of acquisition to manually compensate the DC values measured by the probe. Obviously, this represents a strong limitation of the probe.

Table 6.1. Summary of the results of the experiments conducted on the AES cores implemented on the Cyclone FPGA in the workstation *SCLab*.

Exp	Core	Synthesis	Probe	f _{CK}	Target function	
					XOR	S-Box
1	AES-0	RAM + Logic Elements	CT-1	1MHz	>750k	100k
2	AES-0	Only Logic Elements	CT-1	1MHz	>800k	800k
3	AES-3	Only Logic Elements	CT-1	1MHz	1.2M	>1.2M
4	AES-3	RAM + Logic Elements	CT-1	1MHz	>750k	>750k
5	AES-5	RAM + Logic Elements	CT-1	1MHz	30k	>30k
6	AES-5	RAM + Logic Elements	CT-1	100kHz	100k	>100k
7	AES-5	RAM + Logic Elements	TCP202	1MHz	<i>failed</i>	<i>failed</i>
8	AES-5	RAM + Logic Elements	TCP202	100kHz	<i>failed</i>	<i>failed</i>

This issue makes the probe inadequate for PAAs applications, where the acquisition may take several days and an attacker cannot control the DC compensation of the probe. The phenomenon is depicted in Fig. 6.28, where a drift of the DC value of the sampled traces is clearly visible. After a certain number of acquired traces, about beyond 100k, the DC value of the traces considerably increases, until the current pattern is almost saturated when the number of traces is beyond 300k.

The value of the correlation coefficient strongly depends on the DC value of the traces, as visible in Fig. 6.29 for the case of 100k, 300k, and 1M traces. When the DC values increases, traces are less correlated because the DC variations can be considered as a superimposed noise; for a very high number of traces, they saturate and correlation is zero, making the attack ineffective.

PAAs aren't successful even with 100k traces, when the saturation is still low, which represents the minimum number of traces required for attacking AES-5, as proved in experiment 6. Therefore, the limited bandwidth of the probe as well as the impossibility of AC coupling it to the scope make the TCP202 inappropriate for PAAs applications.

Experiment $n^{\circ}8$

Experiment 8 is an attempt of using the probe TCP202 with a lower clock frequency (100kHz) on AES-5.

Results are the same as the case of experiment 7. In Fig. 6.30 the correlation coefficient curves of the *AddRoundKey* of the round 0 with a number of traces equal to 300k are plotted. Again, no correlation can be detected using the probe TCP202.

A summary of the results of all the experiments is reported in Table 6.1.

6.3 Conclusions

In this chapter we have described some PAAs experiments executed in a real workstation, named *SCLab*, where we have mounted PAAs against some cryptographic cores on a FPGA board. The results of the experiments confirm that all the countermeasures developed in our department enhance the level of security of basic non-countermeasured implementations of the cryptographic algorithms under test.

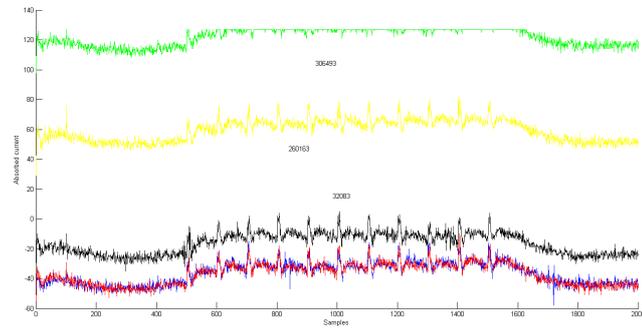


Figure 6.28. Experiment 7: drift effect of the current traces after an increasing number of elaborations for AES-5, when the probe TCP202 is used.

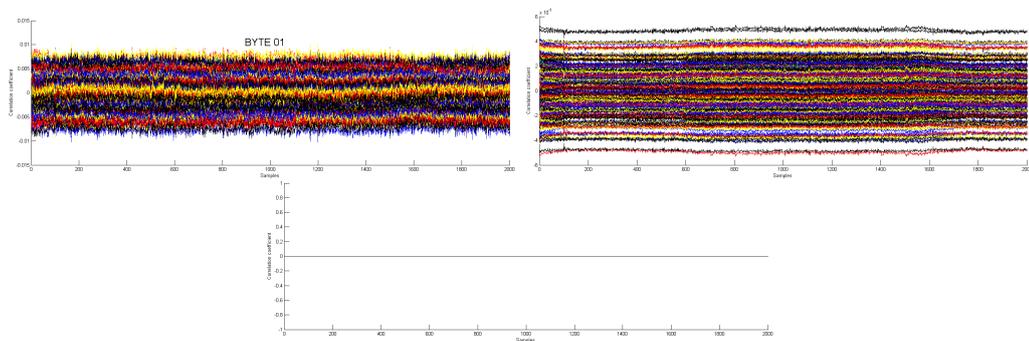


Figure 6.29. Experiment 7: correlation coefficient curves for one byte of the word at the output of the *AddRoundKey* layer of round 0 of AES-5 with 100k, 300k and 1M plaintexts, using the probe TCP202.

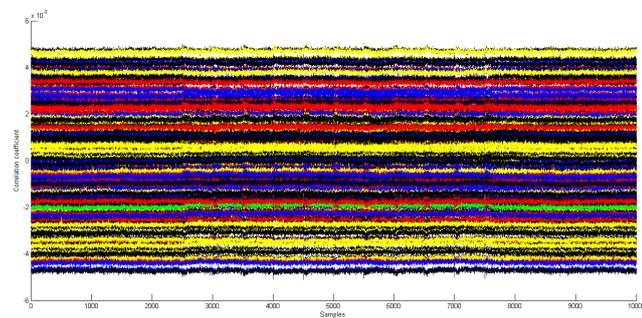


Figure 6.30. Experiment 8: correlation coefficient curves for one byte of the word at the output of the *AddRoundKey* layer of round 0 of AES-5 with 300k plaintexts, using the probe TCP202.

Chapter 7

Conclusions and final remarks

7.1 Relevance of full-chip simulations and design-time evaluation in chip-card manufacturing

As seen in this thesis work, physical observable cryptography is a sub-part of cryptography focused on the analysis of the security of cryptographic devices against physical attacks. Given that each physical emission is strongly related to the technology with which these devices are built, this discipline is strongly permeated with electronics. The design of cryptographic devices should not abstract from physical security issues, which must be assessed already during the design steps in order to prevent the manufacturing of a weak product.

In last years, several manufacturing companies had to face with the new issues introduced by Side-Channels Attacks and had to change they approach towards the design and the production of commercial devices. In this regard, the case of MIFARE DESFire MF3ICD40 represents a prime example. MIFARE is the NXP Semiconductors-owned trademark of a series of chips widely used in contactless smart-cards and proximity cards in thw world, and represents a reference in this field [94]. The MIFARE DESFire MF3ICD40 was introduced in 2002. It was based on a core similar to the previous family product SmartMX, based on microcontroller, with more hardware and software security features than its predecessor MIFARE Classic. Mifare DESFire products were equipped with co-processors running Triple-DES and AES algorithms. In April 2011 researchers of Ruhr University Bochum, led by Professor Paar, announced to NXP that they had broken the security of MIFARE DESFire (MF3ICD40) through Differential Power Analysis attacks against the implementation and published the attack results and details at the annual Workshop on Cryptographic Hardware and Embedded Systems (CHES) [96]. In October NXP had to acknowledge the successful of the attack, announced the discontinuation of the MIFARE DESFire (MF3ICD40) and the introduction of its successor MIFARE DESFire EV1 late 2008, by informing their direct customers and eco-system partners on the crack of the system. From that day on, manufacturer companies realized that putting on the market imperfect products can compromise their reputation and can cause them to waste a lot of money and time; usually, if a card is stolen or lost, it is trivial for the company to deactivate the device and preserve the secret data of their customer; but if their customer cannot detect the fraud and backlist the card as in

the case of implementation attacks, the damages could be considerably high, both for the customer and the card manufacturer.

In order to address these new security issues, manufacturers realized that the problem should be efficiently managed by investing money and time in research in the field of side-channels and physical security. Designers and cryptographers should exchange constantly their role in order to immerse one in the point of view of the other. Nowadays, final products have to pass several strict certification procedures before being put on market, both during the design steps and also after the chip manufacturing. Design-time tests are crucial, because if a device reveals some vulnerabilities before being manufactured, several money and time can be saved. Under this perspective, full chip simulations are important because allow to patch any possible vulnerabilities in the most critical parts of a secure co-processor.

In this context, this thesis work is an attempt to analyze and describe these issues on a specific case study, with the purpose of proposing novel design strategies and metrics in order to overcome the problems which can be encountered during the design of nanometer cryptographic devices and prevent the feasibility of Power Analysis Attacks.

7.2 Summary of the research contribution

The physical security issues treated in this work oscillate between cryptography and electronics. The body of the work has been the analysis of novel countermeasures against PAAs, with a specific focus on transistor level countermeasures, which are considered of main interest for cryptography community because aim at preventing PAAs directly at physical level.

The research contribution of this work can be summarized in the following points:

- Study and analysis of the state of the art on *Dual-rail Precharge Logic* styles (*DPL*) used to design secure co-processors in order to counteract PAAs at circuit level, and focus on the most important leakage sources arising in submicron circuits, such as the electrical mismatches due to the capacitive unbalance and timing mismatch (e.g. early evaluation errors), which are expected to increase with the technology scaling.
- Design of a novel hardware countermeasure against PAAs, named *Time Enclosed Logic*, adoptable for a secure semi-custom digital design flow and based on a two dimensional protection: a new data encoding, based on hiding the instantaneous power consumption in the time domain, has the purpose of balancing at logic level the energy dissipated on a cycle of operation, and a layout optimization based on the insertion of decoupling capacitances on the internal power supply wires of the chip, aims at eliminating the high frequencies data dependent components of the instantaneous power consumption.
- Development of a novel design time metric, based on the study of the variability of the current traces of a circuit in the frequency domain, named *Frequency Energy Deviation*, and proposal of a new design methodology that can be integrable in a secure semi-custom flow. This methodology is based on an

analytical calculation of the energy variability of TEL circuits in the frequency domain, through which the variable data-dependent components are proved to be located at high frequencies and can be efficiently filtered off through the insertion of the on-chip capacitances, always present in the design of digital circuits. The amount of additional capacitance needed for security issues is expected to decrease with the technology scaling.

- Development and discussion of a secure digital semi-custom flow to design cryptographic circuits, based on the adoption of the TEL principle. Under the assumption of meeting some design constraints, a technology library of TEL-compatible gates can be built and adopted to design cryptographic circuits without requiring any additional constraint on the routing of the logic gates, overcoming conventional DPL styles (e.g. SABL), which on the contrary are strongly sensitive on the electrical mismatches and require a precise (but suboptimal) back-end optimization.
- Design of a prototype digital library of logic gates designed for full custom application, named *improved Delay-based Dual-rail Precharge Logic*, which is fully compatible as circuit template to implement TELs. This library has been designed in a CMOS 65nm technology and has been optimized down to the layout level with the purpose of preventing electrical mismatches. The library is composed of some basic combinational gates, input/output converters, and a master-slave flip-flop. The balance of the instantaneous power consumption of these cells has been assessed by using standard physical security metrics as NED, NSD, and CV, in presence of electrical mismatches due to early evaluation mismatches, output load unbalance, process mismatches.
- The iDDPL library has been used to implement a cryptographic case study, the SERPENT-block, which represents the first stage of a Serpent encryption unit. Two different implementations of the circuit have been designed, one with iDDPL gates and one with SABL gates. The design has been executed without doing any assumption on the routing of the differential wires, resulting in an intentionally unbalanced netlist. Extensive PAAs have been mounted against the circuit, using specific software scripts and conventional actual security metrics like PBC and MTD considering Gaussian noise. Under the assumption to consider a limited resolution and bandwidth of a hypothetical acquisition setup, the iDDPL has been proved to strongly overcome the SABL style in terms of PAAs resistance.
- Design of a cryptographic ASIC in CMOS 65nm, the *SERPAES* chip, on which different countermeasures against PAAs developed in our department have been implemented as independent cores. Among them, the SERPENT-block has been integrated as a full custom macro-block. An evaluation testboard has been assembled to host the chip, and an interface circuitry has been designed in order to test the functionality of the ASIC. The SERPAES chip is correctly working and prompt to be analyzed through PAAs measurements using laboratory instrumentation of our department.

- In-depth analysis of *Leakage Power Analysis (LPA)* attacks against submicron circuits. This class of attacks has been proposed for the first time in our department, but research in this field is still at early stages. LPA attacks have been mounted in simulation on a real cryptographic circuit implemented with different DPL styles, and with some assumptions on the clock signal of the circuit and the on-chip noise. Even if the standard LPA procedure has been proposed for linear bit-slice circuits, simulation results prove that it can be also extended to the general case of non-bit slice cores, under the assumption that the leakage power model can be linearized. This an outstanding security requirement, because the trend of cryptography is to design small hardware-oriented cryptographic ciphers for lightweight applications which behaves like bit-slice parallelized circuits and are potentially vulnerable if the linear power model defined in LPA is used. Furthermore, LPA attacks have been mounted against a TEL implementation, which on the contrary exhibited an enhanced level of resistance with respect to the other DPLs, showing that the side-channel arising from the static power can be efficiently counteracted at logic level by adopting a fully dynamic data encoding.
- As additional activity, a measurement setup has been designed in the laboratories of the Ministry of Economic Development in a project in collaboration with the Foundation Ugo Bordoni. The security of some of the countermeasures integrated in the SERPAES chip have been evaluated on a FPGA board.

7.3 Future directions and improvements

The main result of this work has been of course the design of a cryptographic ASIC in CMOS 65nm using a semi-custom design flow, which has been manufactured during the activity, tested with success on functionality and prompt to be validated through extensive measures during the next months in order to validate the SCA resistance of each single core. Among them, the TEL style proposed in this work plays an important role, because it represents an innovative attempt to overcome the electrical mismatches arising in submicron technologies, which are the physical reasons of the success of the attacks against commercial devices, as for example the Mifare DESFire, by using standard automatized CAD tools with no additional constraints on the design flow.

Anyway, several open points and possible future improvements remain to be discussed. First of all, in accordance to the description of the practical activity in Chapter 6, a similar measurement testbench will be built in order to perform real PAAs against the SERPAES chip. More specifically, an adequate attack routine must be implemented using high level languages (e.g. Labview, C, SystemC) and modern equipment. In the future, the security of TEL circuits should be validated using physical security metrics, based on information theory metrics (e.g. conditional entropy and mutual information [71] [116]) recently introduced in the literature on SCAs. At the same time, more advanced statistical distinguishers should be used, which are certainly more efficient and strong then the Pearson's correlation coefficient adopted in standard CPA (e.g. Mutual Information Analysis [44]).

Furthermore, given that several novel side-channels are continuously discovered,

it would be interesting to test the security of TEL circuits on the side-channels arising from the electromagnetic emissions. EMAs is a topic of primary interest in cryptography community and are becoming as important as PAAs, to the extent that several research laboratories are starting to be provided with more expensive and advanced instruments to perform EM measurements.

As an improvement of TEL circuits, the efficiency of this countermeasure should be validated also for FPGA applications, by designing novel circuit templates based on the adoption of CMOS standard cells. In this thesis we have presented and evaluated a full custom style, the iDDPL, but TEL principles could be exported also to CMOS templates. In last years, specific FPGA boards have been developed to perform SCAs, which use more advanced technologies (sub 45nm) and high clock frequencies (more than 100MHz) [52]. Several novel hardware-oriented block ciphers, such as PRESENT [15], have been presented. Implementing these cryptographic cores in TEL style for ASIC as well as FPGA applications would be certainly useful to have a complete evaluation of its strength.

Finally, in Chapter 5 the relevance of Leakage Power Analysis attacks has been discussed. Nowadays LPA attacks represent a debatable argument in SCA community. Even if an efficient practical procedure of attack has not been yet standardized for real experiments, successfully LPA attacks on some cryptographic architectures have been already published. It is known that static power is expected to increase with the technology scaling, leading to the need of developing novel countermeasures to counteract both standard PAAs and LPA attacks. Simulation results presented in this thesis are a step forward in the field of LPA and provide some future directions in the development of new countermeasures to build hardware-oriented cryptographic primitives.

Appendix A

Implementation of 4x4 S-Boxes

In this section the Boolean equations describing the 4x4 S-Box S_0 of Serpent and the 4x4 S-Box of PRESENT are reported.

The S-Box S_0 has been implemented in two ways: the first set of equations is composed of 57 logic instructions; the second set of equations is a lightweight version of S_0 , obtained through an iterative algorithm implemented in software [46] which allows to obtain an optimized instruction set of 16 logic operations.

The implementation of PRESENT, executed using Karnaugh maps and composed of 48 logic instructions, has not been used in our work, but can be studied for further implementations.

Table A.1. Boolean equations describing the 4x4 S-Box S_0 of Serpent and the 4x4 S-Box of PRESENT.

Serpent S_0		PRESENT		
Full	Lightweight			
$n_1 = \neg c \ \& \ \neg d$	$n_{19} = \neg b \ \uparrow \ \neg c$	$m_1 = a \ \wedge \ d$	$p_1 = \neg a \ \wedge \ b$	$p_{19} = a \ \& \ c$
$n_2 = a \ \& \ n_1$	$n_{20} = n_{18} \ \& \ n_{19}$	$m_2 = a \ \& \ d$	$p_2 = b \ \& \ c$	$p_{20} = \neg c \ \& \ p_{16}$
$n_3 = b \ \uparrow \ \neg d$	$n_{21} = n_{20} \ \uparrow \ \neg n_{14}$	$m_3 = c \ \wedge \ m_1$	$p_3 = \neg c \ \& \ d$	$p_{21} = p_{17} \ \& \ p_{18}$
$n_4 = n_3 \ \& \ \neg n_2$	$n_{22} = \neg a \ \& \ b$	$m_4 = b \ \wedge \ m_3$	$p_4 = a \ \mid \ b$	$p_{22} = d \ \& \ p_{19}$
$n_5 = \neg c \ \& \ d$	$n_{23} = n_{22} \ \uparrow \ \neg n_7$	$m_5 = b \ \& \ m_1$	$p_5 = p_1 \ \mid \ p_2$	$p_{23} = p_{20} \ \mid \ p_{21}$
$n_6 = a \ \& \ n_5$	$n_{24} = n_{21} \ \uparrow \ d$	$m_6 = a \ \wedge \ m_6$	$p_6 = p_3 \ \& \ p_4$	$p_{24} = b \ \mid \ \neg c$
$n_7 = c \ \& \ d$	$n_{25} = n_{23} \ \& \ n_{24}$	$m_7 = c \ \mid \ m_6$	$p_7 = \neg d \ \& \ p_5$	$p_{25} = a \ \wedge \ d$
$n_8 = \neg a \ \uparrow \ n_7$	$n_{26} = n_{12} \ \uparrow \ \neg b$	$m_8 = m_3 \ \wedge \ m_6$	$p_8 = \neg c \ \wedge \ d$	$p_{26} = \neg b \ \& \ c$
$n_9 = \neg b \ \uparrow \ n_6$	$n_{27} = c \ \wedge \ b$	$m_9 = m_2 \ \wedge \ m_4$	$p_9 = a \ \wedge \ b$	$p_{27} = \neg a \ \wedge \ d$
$n_{10} = n_8 \ \& \ n_9$	$n_{28} = \neg a \ \uparrow \ \neg d$	$m_{10} = m_8 \ \& \ m_9$	$p_{10} = a \ \& \ b$	$p_{28} = p_{24} \ \& \ p_{25}$
$n_{11} = c \ \& \ \neg d$	$n_{29} = n_{11} \ \uparrow \ n_{14}$	$e = \neg m_6 \ \wedge \ m_{10}$	$p_{11} = c \ \& \ \neg d$	$p_{29} = p_{26} \ \& \ p_{27}$
$n_{12} = a \ \& \ n_{11}$	$n_{30} = n_{27} \ \uparrow \ n_{28}$	$f = \neg m_3 \ \wedge \ m_{10}$	$p_{12} = \neg B \ \& \ p_8$	$e = p_{28} \ \mid \ p_{29}$
$n_{13} = b \ \uparrow \ n_6$	$n_{31} = n_{29} \ \& \ n_{30}$	$g = m_4 \ \wedge \ m_7$	$p_{13} = \neg c \ \& \ p_9$	$f = p_{22} \ \mid \ p_{23}$
$n_{14} = \neg a \ \& \ \neg b$	$n_{32} = b \ \uparrow \ n_2$	$h = m_2 \ \wedge \ m_4$	$p_{14} = p_{10} \ \& \ p_{11}$	$g = p_{14} \ \mid \ p_{15}$
$n_{15} = \neg n_2 \ \& \ \neg n_{12}$	$e = n_4 \ \uparrow \ n_{10}$		$p_{15} = p_{12} \ \mid \ p_{13}$	$h = p_6 \ \mid \ p_7$
$n_{16} = n_{13} \ \& \ \neg n_{14}$	$f = n_{15} \ \uparrow \ n_{16}$		$p_{16} = b \ \wedge \ d$	
$n_{17} = b \ \& \ c$	$g = n_{25} \ \uparrow \ n_{26}$		$p_{17} = \neg a \ \& \ b$	
$n_{18} = a \ \uparrow \ n_{17}$	$h = n_{31} \ \uparrow \ n_{32}$		$p_{18} = \neg c \ \mid \ \neg d$	
# of instructions	57	16	48	

Appendix B

Description of the Matlab scripts for PAAs attacks

```

% Script to post-process current traces exported by Cadence and perform CPA attacks
% In order to perform LPA attacks, this script has been slightly modified

traces = VDD_LK([20001:2600000],1); % read Cadence traces and eliminates first cycles
input = IN([3:260],:); % read data matrix and eliminates the first two nibbles
NK = 16; % set number of possible keys for a 4-bit architecture
points = 100; % set number of samples in a clock cycles
n_traces = 3876; % set number of plaintexts to generate (with ND = 258 to have 1M)

OFFSET= 10 + round(20*rand); % generate a random offset for decimation

T = 10e-8; % set the clock period of the elaboration
TC = 10e-12; % set the sampling period of current traces exported by Cadence
cycles = 3; % set the elaboration latency
D = round((T/TC)/points); % set the scale factor for traces decimation

[NT, n] = size(traces); % set the number of points exported by Cadence
[ND, n] = size(input); % set the number of input plaintexts

NT = round(NT/ND); % set effective number of points for each trace
ND = ND - cycles + 1; % set the effective number of input plaintexts

v = [1:D:(cycles*10000)]; % decimate the current traces with scale D

sigma = std(traces); % calculate standard deviation of the samples
var_sig = sigma^2; % calculate the variance of the samples (signal power)

disp('Predicting the instantaneous power consumption ...'); % start to predict leakage
                                                    power model

leak_model = zeros(ND, NK);

for k=1:NK % calculate the leakage model using the Hamming weight of the word at the
            output of the S-Box
    for i=1:ND
        target = 8*input(i,1) + 4*input(i,2) + 2*input(i,3) + input(i,4);
        leak_model(i, k) = H(SBOX(bitxor(target, k-1)));
    end
end
end

```

```

disp('Simulating acquisition and elaboration of samples ...');

clear key_traces;

for z = 2:12 % in this cycle increasing levels of Gaussian noise on the Cadence samples
             are summed on the traces and averaged

    std_noise = 10^(-z/2);
    var_noise = std_noise^2;
    level = std_noise/sigma;
    NSR = level^(-2); % set different level of noise-to-signal ratio

    for i = 1:n_traces % in this cycle we consider also the jitter of the oscilloscope

        jitter = round(10*rand(1,cycles*100)-5);
        if jitter(1) < 0
            jitter(1) = 0;
        end
        jitter = 0;

        noisy = quant_noise(traces, sigma); % sum quantization noise
        quant_noisy = el_noise(traces, sigma, level); % sum electronic noise
        quantized = QUANT(el_noisy); % traces quantization
        splitted = split(quantized, cycles, ND); % function 'split' orders traces on the
                                                rows of a matrix ND x NT

        [ND, NT] = size(splitted);

        sampled_points = v + jitter; % sum random jitter
        sampled = splitted(:, sampled_points); % sample traces with scale D
        [ND, NT] = size(sampled);
        if (i == 1)
            sum_traces = zeros(ND, NT);
        end

        fprintf('Generating correlation traces of the first %d plaintexts...\n', i*ND);

        sum_traces = sum_traces + sampled;
        leak_traces = sum_traces ./ i; % noisy traces are averaged in order to low pass
                                     filter noise

        for k = 1:NK % in this cycle the absolute value of the correlation coefficient is
                    calculated for each key guess
            C = abs(corrcoef(leak_model(:,k), leak_traces(:,t)));
            for t = 1:NT
                key_traces(k, t) = C(1,2);
            end
        end

        for k = 1:NK
            corr_curves(k,i) = max(abs(key_traces(k, [(round(2*NT/3 + 1)):NT])));
        end

        MTD = CPA_MTD(corr_curves', ND); % calculate the MTD for each key guess

    end
end

```

```
% Generation of the figures

% Plot the correlation curves of each key as a function of the number of traces
(for a specific value of noise)
X = [1:256:256*n_traces];
MAP = [0.6 0.6 0.6];
figure; plot(X, corr_curves', 'linewidth', 2);
hold on; plot(X,corr_curves(12,:)', 'k', 'linewidth', 5);
set(gca,'Ylim',[0 1]);
set(gca,'Ytick',[0:0.2:1]);
set(gca, 'FontSize', 16);
ylabel ('Correlation coefficient');
xlabel ('Input vectors');
grid on;

% Plot the correlation curve of each key as a function of the time
(for a specific value of noise)
figure; plot(key_traces', 'color', MAP, 'linewidth', 4);
hold on; plot(key_traces(12,:)', 'k', 'linewidth', 4);
set(gca,'Ylim',[0 1]);
set(gca,'Ytick',[0:0.2:1]);
set(gca, 'FontSize', 16);
ylabel ('Correlation coefficient');
xlabel ('Time samples');
grid on;

% Calculates the MTD on the correct key for a specific level of noise
rumore(z-1) = std_noise;
metrica(z-1) = MTD;

end

% Plot the MTD as a function of noise
figure; semilogx(rumore, metrica, 'k', 'linewidth', 4);
set(gca, 'FontSize', 16);
ylabel ('MTD');
xlabel ('std noise');
grid on;
```


Bibliography

- [1] Abdollahi, A., Fallah, F., & Pedram, M. (2004). Leakage current reduction in CMOS VLSI circuits by input vector control. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(2), 140-154.
- [2] AES, N. I. S. T. (2001). Advanced encryption standard. Federal Information Processing Standard, FIPS-197, 12.
- [3] Agrawal, D., Archambeault, B., Rao, J. R., & Rohatgi, P. (2003). The EM side—channel (s). In *Cryptographic Hardware and Embedded Systems-CHES 2002* (pp. 29-45). Springer Berlin Heidelberg.
- [4] Aigner, M., & Oswald, E. (2000). Power analysis tutorial. In *Institute for Applied Information Processing and Communication, University of Technology Graz-Seminar*, Tech. Rep.
- [5] Alioto, M., Giancane, L., Scotti, G., & Trifiletti, A. (2010). Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(2), 355-367.
- [6] Alioto, M., Giancane, L., Scotti, G., & Trifiletti, A. (2009, December). Leakage Power Analysis attacks: Well-defined procedure and first experimental results. In *Microelectronics (ICM), 2009 International Conference on* (pp. 46-49). IEEE.
- [7] Alioto, M., Giancane, L., Scotti, G., & Trifiletti, A. (2009, December). Leakage Power Analysis attacks: Theoretical analysis and impact of variations. In *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on* (pp. 85-88). IEEE.
- [8] Anderson, R., Biham, E., & Knudsen, L. (1998). Serpent: A proposal for the advanced encryption standard. NIST AES Proposal, 174.
- [9] Athas, W. C., Svensson, L. J., Koller, J. G., Tzartzanis, N., & Ying-Chin Chou, E. (1994). Low-power digital systems based on adiabatic-switching principles. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2(4), 398-407.
- [10] Baddam, K., & Zwolinski, M. (2008). Divided backend duplication methodology for balanced dual rail routing. In *Cryptographic Hardware and Embedded Systems-CHES 2008* (pp. 396-410). Springer Berlin Heidelberg.

- [11] Barengi, A., Pelosi, G., & Regazzoni, F. (2014). Simulation-Time Security Margin Assessment against Power-Based Side Channel Attacks. IACR Cryptology ePrint Archive, 2014, 307.
- [12] Bhasin, S., Guilley, S., Flament, F., Selmane, N., & Danger, J. L. (2010, October). Countering early evaluation: an approach towards robust dual-rail precharge logic. In Proceedings of the 5th Workshop on Embedded Systems Security (p. 6). ACM.
- [13] Bhatnagar, H. (2001). Advanced ASIC chip synthesis. Springer Science & Business Media.
- [14] Biryukov, A., & Perrin, L (2014). State of the Art in Lightweight Cryptography. http://cryptolux.org/index.php/Lightweight_Cryptography
- [15] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., ... & Vikkelsoe, C. (2007). PRESENT: An ultra-lightweight block cipher (pp. 450-466). Springer Berlin Heidelberg.
- [16] Bouesse, G. F., Sicard, G., Baixas, A., & Renaudin, M. (2004, March). Quasi delay insensitive asynchronous circuits for low EMI. In Proc. 4th International Workshop on Electro-Magnetic Compatibility of Integrated Circuits (pp. 27-31).
- [17] Brier, E., Clavier, C., & Olivier, F. (2004). Correlation power analysis with a leakage model. In Cryptographic Hardware and Embedded Systems-CHES 2004 (pp. 16-29). Springer Berlin Heidelberg.
- [18] Brown, C. E. (1998). Coefficient of variation. In Applied Multivariate Statistics in Geohydrology and Related Sciences (pp. 155-157). Springer Berlin Heidelberg.
- [19] BSIM Group, UC Berkeley Device Group (2013 update). <http://www-device.eecs.berkeley.edu/~bsim3/bsim4.html>
- [20] Bucci, M., Giancane, L., Luzzi, R., Scotti, G., & Trifiletti, A. (2011). Delay-based dual-rail precharge logic. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 19(7), 1147-1153.
- [21] Bucci, M., Giancane, L., Luzzi, R., & Trifiletti, A. (2006). Three-phase dual-rail pre-charge logic. In Cryptographic Hardware and Embedded Systems-CHES 2006 (pp. 232-241). Springer Berlin Heidelberg.
- [22] Bucci, M., Giancane, L., Luzzi, R., & Trifiletti, A. (2012). A flip-flop for the DPA resistant three-phase dual-rail pre-charge logic family. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 20(11), 2128-2132.
- [23] Bucci, M., Guglielmo, M., Luzzi, R., & Trifiletti, A. (2004). A power consumption randomization countermeasure for DPA-resistant cryptographic processors. In Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation (pp. 481-490). Springer Berlin Heidelberg.

- [24] Bucci, M., Giancane, L., Luzzi, R., Varanomuovo, M., & Trifiletti, A. (2006, May). A novel concept for stateless random bit generators in cryptographic applications. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on* (pp. 4-pp). IEEE.
- [25] Bucci, M., Giancane, L., Luzzi, R., Marino, M., Scotti, G., & Trifiletti, A. (2008). Enhancing power analysis attacks against cryptographic devices. *IET circuits, devices & systems*, 2(3), 298-305.
- [26] Cadence Design System, Inc. *Encounter Digital Implementation System User Guide, Product Version 9.1.2* (2010, July).
- [27] Chari, S., Jutla, C. S., Rao, J. R., & Rohatgi, P. (1999, January). Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology—CRYPTO'99* (pp. 398-412). Springer Berlin Heidelberg.
- [28] Chari, S., Rao, J. R., & Rohatgi, P. (2003). Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002* (pp. 13-28). Springer Berlin Heidelberg.
- [29] Charvet, X., & Pelletier, H. (2005). Improving the DPA attack using Wavelet transform. In *NIST Physical Security Testing Workshop* (Vol. 46).
- [30] Chen, Z., & Zhou, Y. (2006). Dual-rail random switching logic: a countermeasure to reduce side channel leakage. In *Cryptographic Hardware and Embedded Systems-CHES 2006* (pp. 242-254). Springer Berlin Heidelberg.
- [31] Chen, H. M., Huang, C. L., & Chang, R. C. (2006, May). A new temperature-compensated CMOS bandgap reference circuit for portable applications. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on* (pp. 4-pp). IEEE.
- [32] Daemen, J., & Rijmen, V. (1999). AES proposal: Rijndael.
- [33] Daemen, J., & Rijmen, V. (2002). *The design of Rijndael: AES-the advanced encryption standard*. Springer.
- [34] Danger, J. L., Guilley, S., Bhasin, S., & Nassar, M. (2009, November). Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors. In *Signals, Circuits and Systems (SCS), 2009 3rd International Conference on* (pp. 1-8). IEEE.
- [35] Diffie, W., & Hellman, M. E. (1977). Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6), 74-84.
- [36] Djukanovic, M., Giancane, L., Scotti, G., & Trifiletti, A. (2009, December). Impact of process variations on LPA attacks effectiveness. In *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on* (Vol. 1, pp. 102-106). IEEE.

- [37] Djukanovic, M., Giancane, L., Scotti, G., Trifiletti, A., & Alioto, M. (2011, May). Leakage Power Analysis attacks: Effectiveness on DPA resistant logic styles under process variations. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on* (pp. 2043-2046). IEEE.
- [38] Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., & Uhsadel, L. (2007). A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24(6), 522-533.
- [39] Fant, K. M., & Brandt, S. A. (1996, August). NULL Convention Logic TM: a complete and consistent logic for asynchronous digital circuit synthesis. In *Application Specific Systems, Architectures and Processors, 1996. ASAP 96. Proceedings of International Conference on* (pp. 261-273). IEEE.
- [40] Ferguson, N., Schneier, B., & Kohno, T. (2012). *Cryptography engineering: design principles and practical applications*. John Wiley & Sons.
- [41] Gandolfi, K., Mourtel, C., & Olivier, F. (2001, January). Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems—CHES 2001* (pp. 251-261). Springer Berlin Heidelberg.
- [42] Gebotys, C. H., Ho, S., & Tiu, C. C. (2005). EM Analysis of Rijndael and ECC on a Wireless Java-based PDA. In *Cryptographic Hardware and Embedded Systems—CHES 2005* (pp. 250-264). Springer Berlin Heidelberg.
- [43] Gierlichs, B. (2007). *DPA-Resistance Without Routing Constraints?* (pp. 107-120). Springer Berlin Heidelberg.
- [44] Gierlichs, B., Batina, L., Tuyls, P., & Preneel, B. (2008). Mutual information analysis. In *Cryptographic Hardware and Embedded Systems—CHES 2008* (pp. 426-442). Springer Berlin Heidelberg.
- [45] Giorgetti, J., Scotti, G., Simonetti, A., & Trifiletti, A. (2007, March). Analysis of data dependence of leakage current in CMOS cryptographic hardware. In *Proceedings of the 17th ACM Great Lakes symposium on VLSI* (pp. 78-83). ACM.
- [46] Gladman, B.R., & Simpson, S. Partially optimized Serpent S-Boxes boolean functions.(1998). On the website http://brgladman.org/oldsite/cryptography_technology/aesr1/f_box.h
- [47] Guilley, S., Hoogvorst, P., Mathieu, Y., & Pacalet, R. (2005). The “backend duplication” method. In *Cryptographic Hardware and Embedded Systems—CHES 2005* (pp. 383-397). Springer Berlin Heidelberg.
- [48] Guilley, S., Sauvage, L., Hoogvorst, P., Pacalet, R., Bertoni, G. M., & Chaudhuri, S. (2008). Security evaluation of WDDL and SecLib countermeasures against power attacks. *Computers, IEEE Transactions on*, 57(11), 1482-1497.
- [49] Guilley, S., Flament, F., Hoogvorst, P., Pacalet, R., & Mathieu, Y. (2007). Secured cad back-end flow for power-analysis-resistant cryptoprocessors. *Design & Test of Computers, IEEE*, 24(6), 546-555.

- [50] Halak, B., Murphy, J. P., & Yakovlev, A. (2013). Power Balanced Circuits for Leakage-Power-Attacks Resilient Design. IACR Cryptology ePrint Archive, 2013, 48.
- [51] Hayashida Filho, A. VHDL code downloaded from the website <http://www.opencores.org>. (2009).
- [52] Hori, Y., Katashita, T., Sasaki, A., & Satoh, A. (2012, October). Sasebo-giii: A hardware security evaluation board equipped with a 28-nm fpga. In Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on (pp. 657-660). IEEE.
- [53] Hwang, D. D., Tiri, K., Hodjat, A., Lai, B. C., Yang, S., Schaumont, P., & Verbauwhede, I. (2006). AES-Based Security Coprocessor IC in 0.18-CMOS With Resistance to Differential Power Analysis Side-Channel Attacks. *Solid-State Circuits, IEEE Journal of*, 41(4), 781-792.
- [54] Inzerilli T., & Riccardi A. (2012). Il Progetto Sesamo: le minacce alla sicurezza dei pagamenti mobili. *La Comunicazione - Note, Recensioni & Notizie*, Pubblicazione dell'Istituto Superiore delle Comunicazioni e delle Tecnologie dell'Informazione (ISCOM), Ministero dello Sviluppo Economico, Numero Unico, Anno 2012, Vol. LVIII, pp. 87-9.
- [55] Iokibe, K., Amano, T., Okamoto, K., & Toyota, Y. (2013). Equivalent circuit modeling of cryptographic integrated circuit for information security design. *Electromagnetic Compatibility, IEEE Transactions on*, 55(3), 581-588.
- [56] ITRS, International Technology Roadmap for Semiconductors (2013 Update). Website: <http://public.itrs.net/>.
- [57] Jovanović, G. S., & Stojčev, M. (2005). Linear Current Starved Delay Element. *Proc. of ICEST*
- [58] Jovanovic, G., Stojcev, M., & Stamenkovic, Z. (2010). A CMOS voltage controlled ring oscillator with improved frequency stability.
- [59] Kamel, D., Renauld, M., Flandre, D., & Standaert, F. X. (2014). Understanding the limitations and improving the relevance of SPICE simulations in side-channel security evaluations. *Journal of Cryptographic Engineering*, 1-9.
- [60] Kobenge, S. B., & Yang, H. (2009). Digitally controllable delay element using switched-current mirror. *WSEAS Transaction on Circuit and Systems*, (7), 599-608.
- [61] Kocher, P. C. (1996, January). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO'96* (pp. 104-113). Springer Berlin Heidelberg.
- [62] Kocher, P., Jaffe, J., & Jun, B. (1999, January). Differential power analysis. In *Advances in Cryptology—CRYPTO'99* (pp. 388-397). Springer Berlin Heidelberg.
- [63] Koç, C. K. (2009). *About Cryptographic Engineering* (pp. 1-4). Springer US.

- [64] Kulikowski, K. J., Karpovsky, M. G., & Taubin, A. (2006, July). Power Attacks on Secure Hardware Based on Early Propagation of Data. In *IOLTS* (Vol. 6, pp. 131-138).
- [65] Kulikowski, K. J., Venkataraman, V., Wang, Z., Taubin, A., & Karpovsky, M. (2008, May). Asynchronous balanced gates tolerant to interconnect variability. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on* (pp. 3190-3193). IEEE.
- [66] Kulikowski, K. J., Su, M., Smirnov, A., Taubin, A., Karpovsky, M. G., & MacDonald, D. (2005, March). Delay insensitive encoding and power analysis: a balancing act [cryptographic hardware protection]. In *Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on* (pp. 116-125). IEEE.
- [67] Leander, G., & Poschmann, A. (2007). On the Classification of 4 Bit S-boxes. In *Arithmetic of Finite Fields* (pp. 159-176). Springer Berlin Heidelberg.
- [68] Lin, L., & Burleson, W. (2008, May). Leakage-based differential power analysis (LDPA) on sub-90nm CMOS cryptosystems. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on* (pp. 252-255). IEEE.
- [69] Maymandi-Nejad, M., & Sachdev, M. (2003). A digitally programmable delay element: design and analysis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(5), 871-878.
- [70] MacDonald, D. J., Karpovsky, M. G., & Hirbbard, A. E. (2005). A Balanced-Power Domino-Style Standard Cell Library for Fine-Grain Asynchronous Pipelined Design to Resist Differential Power Analysis Attacks. In *Master of Science Thesis*. 2005.
- [71] Macé, F., Standaert, F. X., & Quisquater, J. J. (2007). Information theoretic evaluation of side-channel resistant logic styles (pp. 427-442). Springer Berlin Heidelberg.
- [72] Mangard, S., Oswald, E., & Popp, T. (2008). *Power analysis attacks: Revealing the secrets of smart cards* (Vol. 31). Springer.
- [73] Mateos, E., & Gebotys, C. H. (2010, October). A new correlation frequency analysis of the side channel. In *Proceedings of the 5th Workshop on Embedded Systems Security* (p. 4). ACM.
- [74] Matsui, M., Hara, H., Uetani, Y., Kim, L. S., Nagamatsu, T., Watanabe, Y., ... & Sakurai, T. (1994). A 200 MHz 13 mm² 2-D DCT macrocell using sense-amplifying pipeline flip-flop scheme. *Solid-State Circuits, IEEE Journal of*, 29(12), 1482-1490.
- [75] McCoy, M. C., VHDL code downloaded from the website `inmcm-hdl.googlecode.com`, (2010)

- [76] McEvoy, R. P., Murphy, C. C., Marnane, W. P., & Tunstall, M. (2009). Isolated WDDL: a hiding countermeasure for differential power analysis on FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETTS)*, 2(1), 3.
- [77] Menicocci, R., Simonetti, A., Scotti, G., & Trifiletti, A. (2010). On practical second-order power analysis attacks for block ciphers. In *Information and Communications Security* (pp. 155-170). Springer Berlin Heidelberg.
- [78] Menicocci, R., Trifiletti, A., & Trotta, F. (2013, June). Random interleaved pipeline countermeasure against power analysis attacks. In *Ph. D. Research in Microelectronics and Electronics (PRIME), 2013 9th Conference on* (pp. 145-148). IEEE.
- [79] Menicocci, R., Trifiletti, A., & Trotta, F. (2013, June). A logic level countermeasure against CPA side channel attacks on AES. In *Mixed Design of Integrated Circuits and Systems (MIXDES), 2013 Proceedings of the 20th International Conference* (pp. 403-407). IEEE.
- [80] Merkle, R. C. (1978). Secure communications over insecure channels. *Communications of the ACM*, 21(4), 294-299.
- [81] Messerges, T. S., Dabbish, E. A., & Sloan, R. H. (2002). Examining smart-card security under the threat of power analysis attacks. *Computers, IEEE Transactions on*, 51(5), 541-552.
- [82] Micali, S., & Reyzin, L. (2004). Physically observable cryptography. In *Theory of Cryptography* (pp. 278-296). Springer Berlin Heidelberg.
- [83] Moon, Y., Choi, J., Lee, K., Jeong, D. K., & Kim, M. K. (2000). An all-analog multiphase delay-locked loop using a replica delay line for wide-range operation and low-jitter performance. *Solid-State Circuits, IEEE Journal of*, 35(3), 377-384.
- [84] Moore, G. E. (1965). Cramming more components onto integrated circuits.
- [85] Moradi, A. (2014) Side-Channel Leakage through Static Power - Should We Care about in Practice? *IACR Cryptology ePrint Archive*, 2014, 25.
- [86] Moradi, A., & Poschmann, A. (2010). Lightweight cryptography and DPA countermeasures: a survey. In *Financial Cryptography and Data Security* (pp. 68-79). Springer Berlin Heidelberg.
- [87] Moradi, A., Khatir, M., Salmasizadeh, M., & Manzuri Shalmani, M. T. (2009, March). Charge recovery logic as a side channel attack countermeasure. In *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design* (pp. 686-691). IEEE.
- [88] Moradi, A., Eisenbarth, T., Poschmann, A., & Paar, C. (2010). Power analysis of single-rail storage elements as used in MDPL. In *Information, Security and Cryptology-ICISC 2009* (pp. 146-160). Springer Berlin Heidelberg.

- [89] Del Pozo, S. M., Standaert, F. X., Kamel, D., & Moradi, A. Side-Channel Attacks from Static Power: When Should we Care?. On the website <http://perso.uclouvain.be/fstandae/PUBLIS/152.pdf>
- [90] Muresan, R., & Gregori, S. (2008). Protection circuit against differential power analysis attacks for smart cards. *Computers, IEEE Transactions on*, 57(11), 1540-1549.
- [91] Nassar, M., Bhasin, S., Danger, J. L., Duc, G., & Guilley, S. (2010, March). BCDL: a high speed balanced DPL for FPGA with global precharge and no early evaluation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010 (pp. 849-854). IEEE.
- [92] Nassif, S. R. (2000). Modeling and forecasting of manufacturing variations. In *Statistical Metrology, 2000 5th International Workshop on* (pp. 2-10). IEEE.
- [93] Nikolic, B., Oklobdzija, V. G., Stojanovic, V., Jia, W., Chiu, J. K. S., & Ming-Tak Leung, M. (2000). Improved sense-amplifier-based flip-flop: Design and measurements. *Solid-State Circuits, IEEE Journal of*, 35(6), 876-884.
- [94] NXP (2011). MIFARE Classic 1K, product data sheet - Rev. 3.1 - 21 February 2011. Online: http://www.nxp.com/documents/data_sheet/MF1S503x.pdf
- [95] Olsson, U., Drasgow, F., & Dorans, N. J. (1982). The polyserial correlation coefficient. *Psychometrika*, 47(3), 337-347.
- [96] Oswald, D., & Paar, C. (2011). Breaking mifare desfire mf3icd40: Power analysis and templates in the real world. In *Cryptographic Hardware and Embedded Systems—CHES 2011* (pp. 207-222). Springer Berlin Heidelberg.
- [97] Peeters, E., Standaert, F. X., & Quisquater, J. J. (2007). Power and electromagnetic analysis: Improved model, consequences and comparisons. *Integration, the VLSI journal*, 40(1), 52-60.
- [98] Popp, T., & Mangard, S. (2005). Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *CHES 2005* (pp. 172-186). Springer Berlin Heidelberg.
- [99] Poschmann, A. Y. (2009). Lightweight cryptography: cryptographic engineering for a pervasive world. In Ph. D. Thesis.
- [100] Quisquater, J. J., & Samyde, D. (2001). Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security* (pp. 200-210). Springer Berlin Heidelberg.
- [101] Rabaey, J. M., Chandrakasan, A. P., & Nikolic, B. (2002). *Digital integrated circuits (Vol. 2)*. Englewood Cliffs: Prentice hall.
- [102] Rakers, P., Connell, L., Collins, T., & Russell, D. (2001). Secure contactless smartcard ASIC with DPA protection. *Solid-State Circuits, IEEE Journal of*, 36(3), 559-565.

- [103] Ratanpal, G. B., Williams, R. D., & Blalock, T. N. (2004). An on-chip signal suppression countermeasure to power analysis attacks. *Dependable and Secure Computing, Trans. on*, 1(3), 179-189.
- [104] Renauld, M., Standaert, F. X., Veyrat-Charvillon, N., Kamel, D., & Flandre, D. (2011). A formal study of power variability issues and side-channel attacks for nanoscale devices. In *Advances in Cryptology—EUROCRYPT 2011* (pp. 109-128). Springer Berlin Heidelberg.
- [105] Roy, K., Mukhopadhyay, S., & Mahmoodi-Meimand, H. (2003). Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proceedings of the IEEE*, 91(2), 305-327.
- [106] Saarinen, M. J. O. (2012, January). Cryptographic analysis of all 4×4 -bit s-boxes. In *Selected Areas in Cryptography* (pp. 118-133). Springer Berlin Heidelberg.
- [107] Saeki, M., & Suzuki, D. (2008). Security evaluations of MRSL and DRSL considering signal delays. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91(1), 176-183.
- [108] Satyanarayana, H. VHDL code downloaded from the website <http://www.opencores.org/>, (2004).
- [109] Schaumont, P., & Tiri, K. (2007). Masking and dual-rail logic don't add up (pp. 95-106). Springer Berlin Heidelberg.
- [110] Schimmel, O., Duplys, P., Boehl, E., Hayek, J., Bosch, R., & Rosenstiel, W. (2010). Correlation power analysis in frequency domain. *COSADE*, February, 4-5.
- [111] Shamir, A. (2000, January). Protecting smart cards from passive power analysis with detached power supplies. In *Cryptographic Hardware and Embedded Systems—CHES 2000* (pp. 71-77). Springer Berlin Heidelberg.
- [112] Shamir, A., & Tromer, E. Acoustic cryptanalysis: On nosy people and noisy machines. *EuroCrypt 2004 Rump Session*, 2004.
- [113] Shannon, C. E. (1949). *Communication Theory of Secrecy Systems**. *Bell system technical journal*, 28(4), 656-715.
- [114] Souissi, Y., Nassar, M., Guilley, S., Danger, J. L., & Flament, F. (2011). First principal components analysis: A new side channel distinguisher. In *Information Security and Cryptology-ICISC 2010* (pp. 407-419). Springer Berlin Heidelberg.
- [115] Standaert, F. X. (2010). Introduction to side-channel attacks. In *Secure Integrated Circuits and Systems* (pp. 27-42). Springer US.
- [116] Standaert, F. X., Malkin, T. G., & Yung, M. (2009). A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology—EUROCRYPT 2009* (pp. 443-461). Springer Berlin Heidelberg.

- [117] ST Microelectronics. (2013). Handbook of the CMOS065 technology process.
- [118] Suzuki, D., & Saeki, M. (2006). Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In *Cryptographic Hardware and Embedded Systems-CHES 2006* (pp. 255-269). Springer Berlin Heidelberg.
- [119] Synopsys, Inc. Design Compiler User Guide, Version D-2010.03-SP2. (2010, June).
- [120] Tiran, S., Ordas, S., Teglia, Y., Agoyan, M., & Maurine, P. (2014). A model of the leakage in the frequency domain and its application to CPA and DPA. *Journal of Cryptographic Engineering*, 1-16.
- [121] Tiri, K., Akmal, M., & Verbauwhede, I. (2002, September). A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proc. of the 28th European* (pp. 403-406). IEEE.
- [122] Tiri, K., Hwang, D., Hodjat, A., Lai, B. C., Yang, S., Schaumont, P., & Verbauwhede, I. (2005). Prototype IC with WDDL and differential routing-DPA resistance assessment. In *Cryptographic Hardware and Embedded Systems-CHES 2005* (pp. 354-365). Springer Berlin Heidelberg.
- [123] Tiri, K., & Verbauwhede, I. (2004, February). A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings of the conference on Design, automation and test in Europe-Volume 1* (p. 10246). IEEE Computer Society.
- [124] Tiri, K., & Verbauwhede, I. (2004). Place and route for secure standard cell design. In *Smart Card Research and Advanced Applications VI* (pp. 143-158). Springer US.
- [125] Tiri, K., & Verbauwhede, I. (2006). A digital design flow for secure integrated circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(7), 1197-1208.
- [126] Tiri, K., & Verbauwhede, I. (2005, March). Design method for constant power consumption of differential logic circuits. In *Design, Automation and Test in Europe, 2005. Proceedings* (pp. 628-633). IEEE.
- [127] Trotta, F., Menicocci, R., & Trifiletti, F. (2013). RTL Countermeasures Against Power Analysis Attacks Implemented on FPGA. In Ph. D. Thesis.
- [128] Weiser, M. (1991). The computer for the 21st century. *Scientific american*, 265(3), 94-104.
- [129] Yu, P., & Schaumont, P. (2007, September). Secure FPGA circuits using controlled placement and routing. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2007 5th IEEE/ACM/IFIP International Conference on* (pp. 45-50). IEEE.