TRANSIENT ERRORS IMPACT ANALYSIS FOR SUB-POWERED CMOS CIRCUITS AT MULTIPLE LEVELS OF ABSTRACTION OF A DIGITAL SYSTEM

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea Politehnica Timişoara
în domeniul CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI
de către

ing. Sergiu Nimară

Conducător ştiinţific: prof. dr. ing. Mircea Popa
Referenţi ştiinţifici: prof. dr. ing. Liviu Cristian Miclea
prof. dr. ing. Lucian Vinţan
prof. dr. ing. Mircea Vlăduţiu

Ziua susţinerii tezei: 11.11.2016

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Calculatoare şi Tehnologia Informaţiei al Universităţii Politehnica Timişoara, în perioada 2013-2016.

Mulţumiri deosebite se cuvin conducătorului de doctorat prof. dr. ing. Mircea POPA, pentru îndrumarea, consilierea şi sprijinul acordat pe parcursul perioadei de cercetare ştiinţifică.

Timişoara, noiembrie 2016 Nimară Sergiu

Nimară, Sergiu

Transient Errors Impact Analysis for Sub-Powered CMOS Circuits at Multiple Levels of Abstraction of a Digital System

Teze de doctorat ale UPT, Seria 14, Nr. 32, Editura Politehnica, 2016, 120 pagini, 44 figuri, 20 tabele.

ISSN: 2069-8216
ISSN-L: 2069-8216
ISBN: 978-606-35-0098-5

Cuvinte cheie: sub-powered CMOS circuits, reliability assessment, fault injection techniques

Rezumat,

Dezvoltarea de circuite integrate CMOS cu consum redus de energie a devenit critică pentru supravieţuirea industriei semiconductorilor. Metoda cea mai eficientă pentru atingerea acestui obiectiv este scăderea tensiunii de alimentare sub tensiunea de prag a tranzistoarelor nanometrice. În aceste condiţii, circuitele manifestă un comportament probabilistic, indus majoritar de variaţiile de proces, voltaj şi temperatură, precum şi de zgomot.

Scopul cercetării îl constituie analiza impactului erorilor tranzitorii ale circuitelor CMOS subalimentate, la următoarele niveluri de abstractizare ale unui sistem digital: nivelul tranzistor, nivelul poartă logică şi nivelul registru. Abordarea propusă constă în extragerea probabilităţilor şi a modelelor de defectare la nivel tranzistor şi folosirea acestor rezultate pentru dezvoltarea unor metodologii de evaluare a fiabilităţii la nivelurile superioare de abstractizare.

Table of Contents

ABSTRACT

This thesis, entitled "Transient errors impact analysis for sub-powered CMOS circuits at multiple levels of abstraction of a digital system" includes the research performed during the doctoral studies at Politehnica University of Timisoara for a duration of 3 years in the topic of digital systems based on sub-powered CMOS circuits.

According to Moore's law, the number of transistors integrated on a chip has doubled roughly every 18 months, leading to a nearly exponential growth of the capabilities of microelectronic devices. Since 2002, several scientific papers anticipated the moment when further increase of the degree of integration of chips will face a physical limit. Actual requirements impose a multi-objective design strategy of future integrated circuits, according to the following parameters: energy efficiency, performance, silicon area and fault tolerance.

Developing low-power CMOS circuits has become critical for the survival of the semiconductor industry and the preferred method for achieving this desideratum is the reduction of the supply voltage, because it influences both the static and the dynamic components of power. The actual dimensions of transistors, situated in the nanometer domain, combined with a low supply voltage, in the near or sub-threshold regions and with process and temperature variations which become prevalent in this context, all lead to an important reliability decrease, which can no longer be ignored. Sub-powered logic gates exhibit a probabilistic behavior, which is caused mainly by two factors: the inability of the gate to perform the transition between the two logic levels in the desired time window or the occurrence of a transient error which affects the logic value at the output of the gate. Transient errors, called "single event upsets", are usually induced by electromagnetic interference, noise, radiation, thermic fluctuations, crosstalk effect, supply voltage variations and process variations.

In this context, analyzing the behavior of sub-powered CMOS circuits and developing efficient reliability assessment methodologies at different levels of abstraction becomes imperative. The present work is organized at three levels of abstraction of digital systems: circuit level, gate level and RTL. The fault models and the experiments carried on at lower levels are used in order to derive upper level techniques and results.

Timişoara, 2016 Sergiu NIMARĂ

ACKNOWLEDGEMENTS

This work was partially supported by the Seventh Framework Programme of European Union under Grant Agreement 309129, project “i-RISC – Innovative Reliable Chip Design from Low Power Unreliable Components”. I would also like to acknowledge the academic and financial support of University Politehnica Timisoara.

Firstly, I would like to thank my coordinator, prof. Mircea POPA, who accepted me as his PhD student and offered me the opportunity to consolidate my technical knowledge in the field of Computer and Information Technology.

Secondly, I wish to express my deepest gratitude to Assoc. Prof. Alexandru AMARICAI-BONCALO, who encouraged me to follow this path and counted on my work as a member of the research team of the i-RISC project. He inspired me to take up the challenge of becoming a PhD candidate, he offered me the opportunity to attend several international conferences and scientific events and he guided me through all the technical issues arisen during these years. Thirdly, I would like to thank lecturer Oana AMARICAI-BONCALO for her continuous support and inspiring ideas.

Thanks go also to my family, who encouraged me to never give up.

List of figures

List of tables

1. INTRODUCTION

1.1 Motivation

First expressed by Gordon Moore in 1965 and soon after that referred in the literature as Moore's law, this statement has proven to represent the driving force behind the computing technology revolution over the past decades. According to it, the number of transistors on an integrated circuit doubles every one to two years. The degree of integration of the chips has increased exponentially in a period of over 50 years, following Moore's law and generating a nearly constant exponential growth of the capabilities of silicon-based microelectronics. By 1970, at least 1000 transistors could be integrated on a chip; by 1990, the number of transistors integrated on a chip has reached one million and by 2010, one billion has been reached. Moore's prediction has revealed to be an accurate one and it has become a target that must be met by semiconductor device manufacturers in order to remain competitive [1][2].

Fig. 1.1 illustrates the exponential increase of processors speed, measured in millions of instructions per second (MIPS), during a 35 years period. As far back as 2002, several papers were published, in which the authors manifested their fear that further increase of the integration density of chips may face a physical limit. These papers anticipated that serious miniaturization problems will be faced in 6-10 years, a fact that was already acknowledged. The main factors that will potentially lead to a sudden end of Moore's law are the increasing thermal noise voltage on decreasing characteristic capacitances, along with the necessity of using lower supply voltages on purpose to reduce the power consumption, without increasing clock frequency [1].

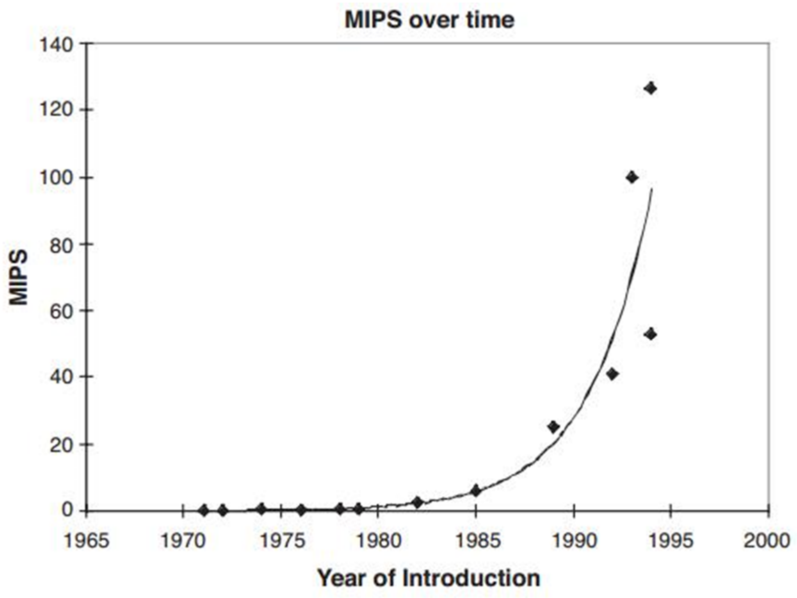


Fig. 1.1 Processor speed in MIPS versus year of introduction, [3]

Today's design optimizations can handle only one or two objectives, which are performance and power. This mindset must change, moving towards multivariable design optimizations. Future designs will have to meet the increasing requirements for power efficiency and performance and they will have to be optimized for density, active and leakage power, low fabrication cost and low error rate [4]. Due to fundamental physical limitations, current designs will not be able to meet all these requirements and to function reliable at nanometer scale, so a wide range of new nanoscale devices is being analyzed in order to permit the efficient processing and storage of digital data. The higher complexity and the higher performance demanded for nowadays integrated circuits inevitably lead to an increase of power consumption, so present and future levels of integration will require new approaches in order to design, manufacture and test reliable low-power devices.

One of the biggest challenges of the emerging nanoelectric era is represented by the ability to control the fault tolerance characteristics [5]. For example, in circuit design, replacing regular flip-flops by soft-error-tolerant hardened flip-flops will improve soft-error tolerance by almost 10 times. A VLSI or ULSI chip can comprise even tens of billions of transistors, but many of them might be unusable due to extreme static variations. Besides, dynamic variations of supply voltage and temperature, frequent and intermittent soft errors and transistors that slowly age and degrade over time will all lead to a decrease in performance [4]. Despite all these reliability issues, users will expect the system to remain reliable and to continue to function at the required performance. In this context, improvements and paradigm shifts will be necessary during all the stages of the VLSI design flow.

Another important barrier is represented by energy and power dissipation, which is an important issue especially for mobile battery-powered electronic devices. In this context, developing energy-efficient solutions has become critical to the survival of the semiconductor industry. In the past, each generation of electronic devices was replaced by its successors when its energy overheads became prohibitive [6].

Low-energy consumption can only be sustained through low-powered components. The preferred method for reducing the power dissipation of digital CMOS integrated circuits is represented by aggressive scaling of the supply voltage to sub and near-threshold regimes. But, low supply voltages coupled with the scaling of transistor sizes to nanometer levels and with process and temperature variations affecting these circuits, make them inherently unreliable, causing a probabilistic behavior. The output of a logic gate supplied at a low voltage will be considered as a logic "0" with a probability *p* and it will be considered as a logic „1" with a *1-p* probability. This can be the consequence of two factors: the inability of the logic gate to switch in the desired time window or a single event upset which causes a bit-flip of the output of the gate.

Several papers in the literature establish the basis of a grandiose purpose: building reliable circuits from unreliable components [7][8][9], while other papers suggest to turn to advantage the probabilistic behavior of such circuits in certain classes of applications [10]. New solutions for efficient and fault-tolerant data processing and storage must be studied in order to make the production of low-power reliable chips possible. Both device and system-level fault tolerant solutions are required, using mathematical models, algorithms and strategies belonging to information theory. Even if the system is based on unreliable hardware, error correcting codes and encoder / decoder architectures will assure system-level fault tolerance, using a telecommunications inspired approach.

In this context, this PhD thesis addresses one of the most critical challenges for the next-generation electronic circuits design: building reliable nanoscale chips out of unreliable low-powered components. In order to build the basis of such a system, the impact of transient errors induced by the low-powered CMOS gates must be analyzed at all levels of abstraction of a digital system. The probability density functions for the low-powered unreliable components must be extracted, as they will be used to derive fault models and reliability assessment methodologies at higher levels of abstraction.

1.2 Thesis Goals

Throughout this thesis, a bottom-up approach is employed in order to analyze the impact of the probabilistic behavior of sub-powered CMOS circuits, at three levels of abstraction: transistor level, gate level and register transfer level (RTL). The lowest level, transistor or circuit level, features a representation of the system from the analog point of view, being oriented on the input-output characteristics of the circuit. At this level, the continuous voltage variation at the output of a gate is monitored. In order to decide if the output of the gate is correct, the value of the voltage is compared with a reference value, usually Vdd / 2. The analysis at transistor level is essential, because the continuous variations of current and voltage must be monitored for different parameters of the noise affecting the circuit. From these variations, we can extract the exact moment when the gate performs the switching activity, therefore the delay required for both 0 to 1 and 1 to 0 transitions can be measured. This way, we can decide either the gate is able to function reliable under different noise assumptions, at the required frequency, or it has a probabilistic behavior, with an associated probability of failure.

The probability of failure of a low-powered gate, prone to variations, is an essential characteristic that must be known when analyzing the circuit from the logic point of view. This is why the probabilities associated to each type of gate are used at the next level (gate level) in order to assess the impact of faulty transitions propagating alongside different paths of a digital circuit composed of thousands of gates. However, for modern digital systems, analyzing the propagation of logic errors only at gate level becomes prohibitive, due to the enormous number of gates. Hardware description languages offer the alternative of specifying the implementation of the system using RTL statements. An RTL description of a large system is easier to follow and easier to modify, but a correspondence between gate-level faults and RTL faults must be found. As described in chapter 6 of the thesis, a hybrid view of the system, at both gate level and RTL proves to be very efficient.

To perform a relevant simulation based assessment of circuits' reliability, we need to define very accurate fault models. The aim of this research is to make a rigorous study and analysis of digital systems based on sub-powered CMOS gates, under different disturbance assumptions, at several levels of abstraction.

At transistor level, SPICE simulations are used in order to extract the fault models of basic logic gates under different noise model assumptions, for an ample number of runs. The results of these simulations are used to derive gate-level fault models and the reliability parameters of small and medium circuits composed of low-power gates are analyzed.

1.3 Organization

This PhD thesis is organized as follows:

Chapter 2 sketches the basic concepts, issues and challenges of sub-powered CMOS circuits, by presenting state-of-the-art implementations of sub-powered devices, their domains of applicability and the problems they are facing.

Chapter 3 gives an overview of the division of a digital system into multiple abstraction layers and presents the work performed at the first level of abstraction (circuit level) in order to derive the probabilistic behavior and the associated fault models.

Chapter 4 is dedicated to reliability assessment methodologies at gate-level. Two different approaches are presented, along with their advantages and drawbacks.

Chapter 5 comprises a technique for reliability assessment of probabilistic interconnects, based on saboteurs with different accuracies.

Chapter 6 tackles the reliability assessment issue for sub-powered CMOS circuits at a superior level of abstraction, the Register-Transfer Level (RTL), by introducing a new hierarchical hybrid approach. The new methodology is validated for a simple circuit of medium-size and its benefits are demonstrated by calculating the reliability parameters of a complex AES crypto-core.

Chapter 7 is dedicated to RTL error detection and correction with Low-Density Parity-Check (LDPC) codes. In the beginning of the chapter, a brief presentation of the LDPC codes is made, followed by the characteristics of the LDPC decoder architectures found in the literature. A new architecture is introduced and the reliability assessment methodologies presented in the previous chapters are employed in order to derive the reliability characteristics of the decoders.

Chapter 8 is reserved for the concluding remarks, it outlines the thesis contributions and it enounces the research directions that may be tackled in the future.

2. SUB-POWERED CMOS CIRCUITS

2.1 The Need for Reducing Energy Consumption

In the last three decades, continuous technology scaling has permitted an increase of five orders of magnitude of the VLSI performance, leading to important progress in various computing devices used in healthcare, education, communications and security. Transistor integration in a VLSI design has been limited by die size, chip yields and design productivity. But as far back as 2005 [4], the main focus of the designers has shifted to energy consumption and power dissipation. As stated in [6], while Moore's law continues to provide additional transistors for every generation of integrated circuits, "power budgets are beginning to prohibit those devices from actually being turned on".

Starting with the 65 nm technology node, the supply voltage of the transistors has remained approximately the same, leakage currents continue to increase and dynamic energy efficiency improvements are limited. These factors have determined the VLSI designers to confront with a serious issue: more gates can fit on a die with each new generation, but an increasing percent of them cannot be used due to strict power requirements [6].

Extending battery and system lifetime has become one of the hot topics in the last years and digital circuits' manufacturers are bringing forward the low power consumption as a key feature of their new products, which can cover a wide range of applications, from radio frequency identification (RFID) tags to mobile devices. These low-power requirements have led to an important research effort in the field of sub-powered circuits and to the development of several prototypes, which have not yet gained widespread commercial adoption, but are proved to provide important benefits.

This chapter describes the theoretical foundation of super-threshold, near-threshold and sub-threshold computing in the first part and realizes a comparison between key characteristics of state-of-the-art implementations of sub-powered CMOS circuits. Moving forward, an overview of the application areas of these circuits is presented and the reliability issues are brought into discussion, along with the possible types of errors and the methodologies used in the literature for reliability assessment. Furthermore, the concept of probabilistic CMOS is explained and some applications that embody the probabilistic behavior naturally are brought into attention.

2.2 Super-Threshold, Near-Threshold and Sub-Threshold Computing

Complementary metal-oxide-semiconductor (CMOS) is the most widely used technology for developing integrated circuits in today's microprocessors, microcontrollers, digital signal processors (DSPs), static RAMs and other digital circuits. CMOS is also used for some analog circuits such as image sensors (CMOS sensors), data converters and highly integrated transceivers for several communication technologies. The two most important characteristics of CMOS devices are high noise immunity and low static power consumption. Logic gates in this technology are implemented using a combination of p-channel and n-channel metal-oxide-semiconductor field-effect transistors (MOSFETs) [11].

CMOS is used in most very large scale integrated (VLSI) or ultra-large scale integrated (ULSI) circuit chips. Usually, chips containing thousands or millions of MOSFETs belong to the VLSI category and chips containing billions or even more MOSFETs belong to the ULSI category. Historically, the power consumption was not the primary concern when designing CMOS circuits, so the supply voltages were much larger than the threshold voltage: V_{dd} might have been 5 V and V_{th} approximately 700 mV.

According to Dennard's Scaling Theory, also known as MOSFET scaling, the power density should remain constant with each new and smaller technology node, so that the power use stays in proportion with area: both voltage and current are supposed to scale (downward) with length. Also, device delay should decrease linearly as the dimensions of the transistors are getting smaller [12]. But, when analyzing real world data, we can observe that since the 90 nm technology node, the supply voltages have hardly decreased and Dennard's Scaling Theory doesn't apply anymore. Consequently, instead of remaining constant, we assist to an almost exponential growth of the power density for technology nodes below 90 nm (fig. 2.1). This represents a real problem, because more gates can fit on a die, but they cannot actually be used due to the power limits, which are also known in the literature as the power wall [12]. Because the power consumption per unit area of the chip has risen dreadfully for the last generations, several attempts to lower the supply voltage to values near the threshold voltage or even further have been made and new concepts like near-threshold computing (NTC) or sub-threshold computing (sub-V_{th}) have emerged.

As V_{dd} is scaled down in order to reduce the power density and to minimize the energy per operation, field effect transistors (FETs) make the transition from superthreshold (super-V_{th}) operation in strong inversion with large gate overdrives to near-V_{th} (NTC) operation in weak inversion with very small overdrives and finally into sub-V_{th} operation. Most existing designs have maintained a „safe" difference between the values of supply voltage and threshold voltage in order to target increased robustness and high performance [13].

Near-Threshold Computing (NTC) refers to a regime for which V_{dd} is set to a value only slightly higher than the transistors' threshold voltage, V_{th}. For modern technology nodes, this corresponds to V_{dd} being equal to approximately 500 mV, while the V_{dd} in conventional Super-Threshold Computing (STC) environments is set to approximately 1 V. NTC reduces the energy per operation several times compared to STC, so it manages to pushback the so-called manycore power wall. Therefore, the power is expected to be reduced by an order of magnitude in NTC, compared to STC, which represents a major advantage for multicore systems which also implement parallelism. Fig. 2.2 shows the inverse of energy per operation, namely the energy efficiency, measured in MIPS/Watt (left Y axis) and the transistor delay (right Y axis) as a function of V_{dd}. When analyzing these graphs, we can observe that the energy efficiency is high and the transistor delay is relatively low in the NTC region. A higher value of the V_{dd} determines an important reduction of the energy efficiency [14].

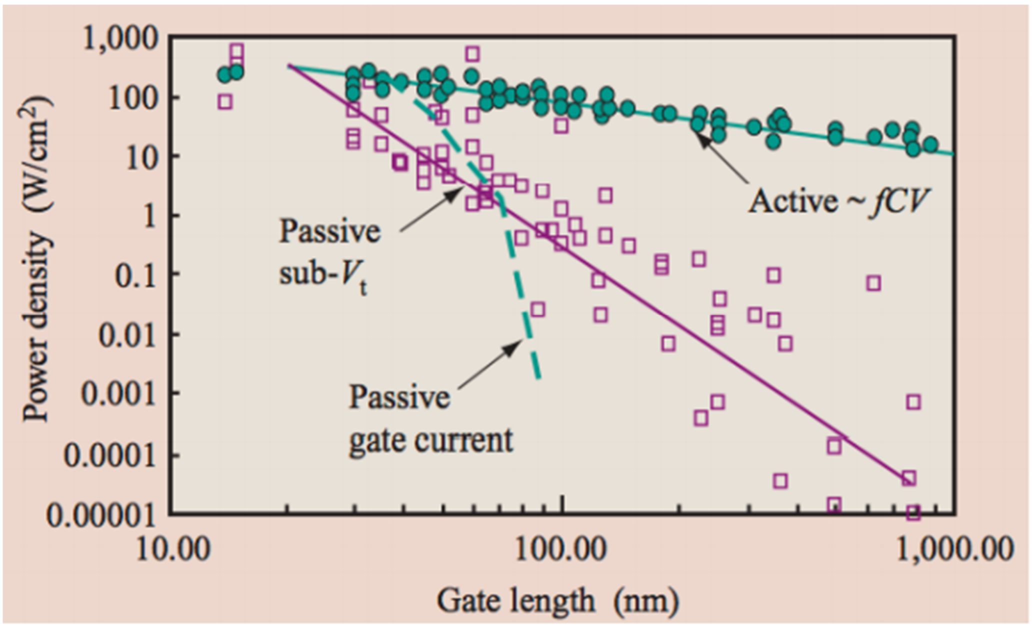


Fig. 2.1 Power density evolution for different technology nodes, [13]

Power dissipation in the active mode, which is composed of the dynamic and static components has a cubic dependence on supply voltage. Empirical studies show that circuit speed has an approximately linear dependence on supply voltage. Therefore, a reduction in supply voltage can contribute to important energy savings at a modest performance loss, which can be compensated for in parallelizeable workloads by a linear increase in the number of processors, which is equivalent to a linear increase of the chip area [15]. The reduction in Power / MIPS, which is a measure of energy / operation and the increase in area / MIPS, which is inversely related to performance in a parallel system, can be the subject of a trade-off solution, which is a balance point achieved for a supply voltage of approximately 0.5 V, across different technology generations. The authors in [15] have plotted the dependence of power dissipation and chip area on supply voltage in fig. 2.3 and claim that the operation of a circuit at 0.5 V can provide an 8x improvement in power efficiency with a moderate 4x frequency loss, which can be compensated for by using parallelism.

The ability to reduce the power consumption of a system by using near-threshold computing must be analyzed in accordance with the possibility of utilizing parallel algorithms and multiple cores on that system. The pervasiveness of near-threshold operation is limited by single-thread performance needs, so the use of heterogeneity in system design will be required. In order to find a trade-off solution between throughput and single-thread performance needs, combining near-threshold cores with traditional super-threshold cores into heterogeneous systems may provide the optimal solution. The authors in [15] envision two possible solutions: (1) a parallel heterogeneous system that has a few high-frequency high-voltage cores and many efficient, moderate-frequency near-threshold cores or (2) a dynamically adjustable parallel system in which the supply voltage of some cores can be either augmented in order to target single-threaded performance, either lowered in order to improve throughput performance.

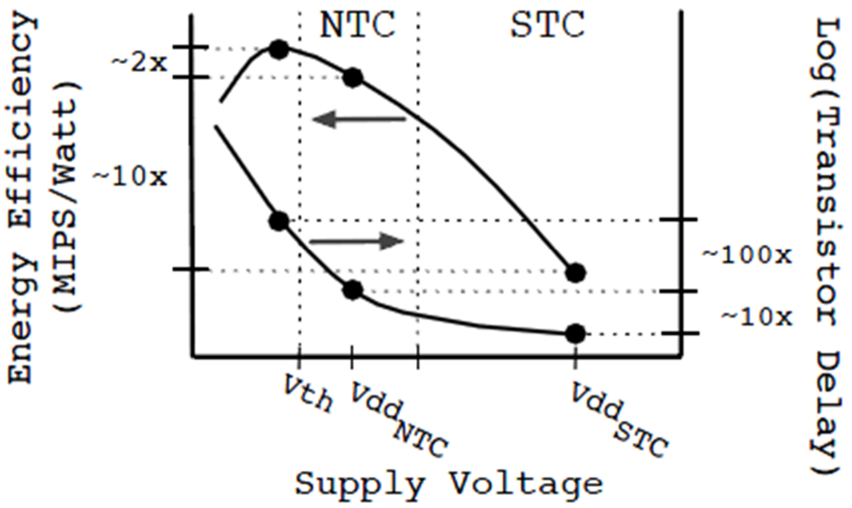


Fig. 2.2 Impact of supply voltage on energy efficiency and delay, [14]

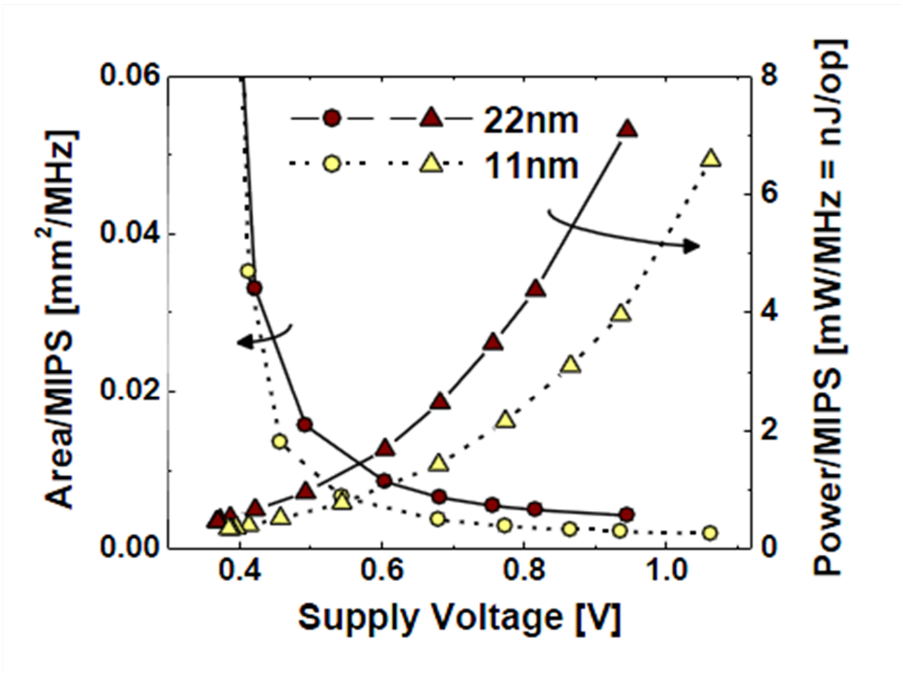


Fig. 2.3 The balance between parallelizeable performance (area/MIPS) and power efficiency (power/MIPS) for near-threshold operation, [15]

The concept of sub-threshold computing was first discussed in the year 1972 as the means to minimizing the supply voltage of the CMOS circuits. Low power applications which used analog sub-threshold circuits were studied the years that followed, but research in the field of digital sub-threshold circuits was hardly performed for the first time in the late 1990s. The subthreshold regime is based on scaling the supply voltage below the threshold voltage, where load capacitances are charged / discharged by subthreshold leakage currents. This fact limits the maximum performance of subthreshold circuits, because leakage currents are orders of magnitude lower than drain currents in the strong inversion regime. Operating the circuit in the subthreshold region implies to be able to use the subthreshold leakage current as the operating drive current [16][17]. Architectural techniques, especially those that involve heterogeneous multiprocessor designs, can be used to improve the performance penalty suffered as a result of low-voltage functioning. An example of such a system is a multiple-core microprocessor in which the V_{dd} of each core is varied according to performance needs and power constraints during operation [13].

According to [13], sub-V_{th} operation differs from super-V_{th} operation primarily because the sub-V_{th} on-current (I_{on-sub}) depends exponentially on threshold voltage (V_{th}) and power supply voltage (V_{dd}), while the typical super-V_{th} on-current ($I_{on-super}$) depends linearly on V_{th} and V_{dd}. The I_{on-sub} exponential sensitivities to V_{th} and V_{dd} are described by the following equations:

$$I_{on-sub} = \frac{W}{L_{eff}} \cdot \mu_{eff} \cdot C_{ox} \cdot (m-1) \cdot v_T^2 \cdot \exp\left(\frac{V_{gs} - V_{th}}{m \cdot v_T}\right) \cdot \left[1 - \exp\left(-\frac{V_{ds}}{v_T}\right)\right], where\ v_T = \frac{kT}{q}$$

W represents the gate width, L_{eff} represents the effective gate length, μ_{eff} is the effective mobility, C_{ox} is the oxide capacitance, m is the subthreshold slope factor, v_T is the thermal voltage, k is Boltzmann's constant, T represents the temperature and q is the charge of an electron.

This exponential sensitivity of I_{on-sub} to V_{th} has an important effect on circuit behavior, because the circuit delay and power also depend exponentially on V_{th} and V_{dd}.

Static power is estimated to represent approximately 15-20 % of the total power of an integrated circuit designed in the 130 nm technology, with significant percentages in both active and standby mode, so the reduction of the leakage current (I_{off}) is an important objective. Leakage reduction is especially important during stand-by mode for energy-constrained systems and it can be achieved by voltage scaling, because both subthreshold current and gate current decrease dramatically with V_{dd}. In order to maintain reasonable device switching speeds at low supply voltages, the threshold voltage (V_{th}) must also be reduced, knowing the exponential relationship between V_{th} and I_{off} [11].

When operating a circuit in the subthreshold region, the dynamic energy consumption is reduced quadratically with V_{dd}, so the minimum energy operation point usually occurs in this region. As CMOS devices are scaled more and more, the main challenges that appear are reduced on-off current ratios and increased sensitivity to variations. In the strong inversion region, the active energy of a gate dominates the total energy dissipated and it can be formulated like $E_{active} = \alpha C V_{dd}^2$, where α is the activity factor, C is the total switched capacitance of the gate and V_{dd} is the power supply. The fluctuation of the static and dynamic components of the total energy, when V_{dd} is varied, and also the region where the minimum energy point occurs, are shown in fig. 2.4.

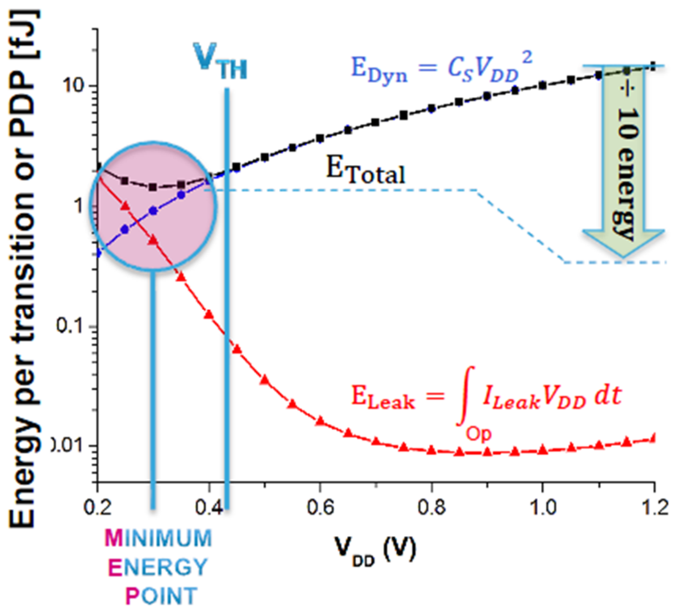


Fig. 2.4 Minimizing energy consumed in digital circuits, [18]

2.3 Physical Implementations of Sub-Powered CMOS Circuits

The permanent quest for reducing the power consumption, combined with the loss of performance exhibited by low-power circuits, has determined the microelectronic suppliers to offer two categories of devices for technology nodes below 100 nm: a high-performance (HP) category and a low-power (LP) category. Hand-held devices, battery powered devices and low standby power applications represent the natural candidates for sub-powered CMOS implementations, belonging to the LP category. This section makes a review of a few physical implementations of sub-powered CMOS devices found in the literature, highlighting the key aspects of each of them. Firstly, a near-threshold voltage 32 nm processor developed by Intel [19] is described, followed by a 180 mV subthreshold FFT processor [20], a 65 nm subthreshold microcontroller with integrated SRAM [21], a sub-200 mV processor [22] and a 32 nm near-threshold voltage register file [23]. At the end of the section, a short comparison of the main parameters of each of these devices is made.

2.3.1 A Near-threshold voltage 32 nm Pentium Processor

Paper [19], written by Kaul and Anders from Intel Corporation, analyzes the design techniques which are required for reliable operation of digital circuits for different levels of the supply voltage, from nominal down to sub-threshold region. The authors highlight the idea that scaling the supply voltage for a conventional circuit can be performed only within certain limits, because voltage sensitive circuits will start failing much before reaching the threshold voltage value. This is the reason why the design of the circuits which are supposed to function at very low supply voltages must follow some principles.

For example, conventional 6T static memory cells are designed with small transistors for high density so they are more prone to variations and stability issues at lower voltages. Therefore, the authors in [19] claim that larger 6T memory cells or even 8T or 10T SRAM cells are desirable for low-voltage operation. Among other techniques, the authors discuss the possibility of applying body bias in order to compensate for logic performance variations. System level techniques are also employed in order to limit the overall logic throughput fluctuations in a many core system, where each core will exhibit a slightly different operation frequency, due to variation.

Intel corporation developed a 32 nm experimental CMOS processor, which is able to operate over the full voltage range, from nominal to subthreshold supply. The design was developed considering the issues and the challenges mentioned above. The maximum energy efficiency, almost 10x greater than the one corresponding to the nominal supply voltage, is achieved when the circuit operates close to the threshold voltage. At nominal supply voltage, the performance is the highest, with modest power and energy efficiency. The lowest power is achieved in the subthreshold regime, with reduced performance and modest energy efficiency, at a supply voltage of 280 mV and a frequency of 3 MHz, where the processor dissipates only 2 mW of power. Table 2.1 proves that the highest energy efficiency (5830 Mips/W) is achieved in the NTV (near-threshold voltage) regime, at a supply voltage of 0.45 V and a frequency of 60 MHz, where the processor dissipates 10 mW of power [19]. According to the data in table 2.1, the improvement in energy efficiency during NTV operation is almost 5X higher than in normal operation.

	Ultra-low power	Energy Efficient	High performance
Supply voltage	0.28 V - subthreshold	0.45 V - nearthreshold	1.2 V - superthreshold
Operating frequency	3 MHz	60 MHz	915 MHz
Dissipated power	2 mW	10 mW	737 mW
Energy efficiency	1500 Mips / W	5830 Mips / W	1240 Mips / W

Table 2.1 Performance obtained by the 32 nm Intel experimental processor (data courtesy of [19])

2.3.2 A 180-mV sub-threshold FFT processor

The authors in [20] aimed to obtain the minimum energy-point for a Fast Fourier Transform (FFT) processor, which operates at 180 mV, is designed in the 180 nm CMOS technology and is used for wireless sensor networks. For the fixed nominal threshold voltage of the FFT processor, which is 450 mV, the minimum energy point occurs for V_{dd} = 400 mV. But the propagation delay increases exponentially in the subthreshold region. The authors of this article perform minimum supply voltage analysis in addition to minimum energy point analysis. The FFT processor proposed in this paper was designed using a modified standard logic cell library, custom multiplier generators and custom memory generators. The lowest voltage supply for correct operation is 180 mV with a clock speed of 164 Hz and the power dissipated is 90 nW. The minimum energy point occurs at 350 mV. The authors consider that the optimum supply voltage is 350 mV with a clock frequency of 10 kHz, for which the dissipated energy was 155 nJ.

2.3.3 A 65 nm sub-Vt microcontroller with integrated SRAM

Paper [21] presents a 65 nm sub-threshold SoC which consists of a microcontroller core with 128KB of SRAM, which operates at sub-threshold voltages and a switched-capacitor DC-DC converter with output voltages of 0.3 to 0.6 V. The microcontroller features a 16 bits RISC architecture, it has unified instruction and data memory, GPIO ports, a watchdog timer, a JTAG interface and three low-power modes. In order to reduce the energy consumption, unused blocks are power-gated during standby, while SRAM and key CPU blocks are powered at 300 mV and they hold their state. In a sub-threshold register, the integrity of data may be affected because inverters with reduced output levels decrease the hold static noise margin (SNM) of latches. A multiplexer-based static register was designed in order to increase robustness.

The 128 KB SRAM of this system is designed to function down to the same minimum V_{dd} as the core logic. The authors find that the minimum energy point of this system occurs at 500 mV. When operating at a frequency of 434 kHz, the energy consumption of the system is 27.3 pJ/cycle. During stand-by mode, V_{dd} is scaled to 300 mV and the combined power for core logic and SRAM is less than 1 µW. The reduction of the supply voltage during stand-by leads to a leakage power reduction of 2.1x [21].

2.3.4 A sub-200-mV 8-bit processor

Scott Hanson et al. [22] have published a paper about an 8-bit sub-threshold processor used in ultra-low-energy sensor networks, that is functional below V_{dd} = 200 mV and achieves the minimum energy of 3.5 pJ/instruction at V_{dd} = 350 mV, with a frequency of 354 kHz. The use of body biasing is investigated in order to minimize the effects produced by process and temperature variations. Architectural decisions can have a notable impact on the energy efficiency of the processor. As a measure of reducing energy consumption, S. Hanson et al. choose a RISC architecture with instructions of 12 bits and divide both data and instruction memory into pages of 16 words each, which permits single cycle access to the contents of a certain page. A three-stage pipeline architecture is chosen for the CPU, keeping the number of sequential devices to a moderate level. Gates with large fan-ins have reduced noise margins at low voltages, so the authors use only CMOS gates with a maximum fan-in of two for this design. A robust memory built from latches and a mux-based read-out structure is used for the instruction memory, data memory and register file. The interconnect RC delay is only a function of materials and circuit geometry and does not depend on the V_{dd} scaling. The processor proposed in the cited paper was fabricated in a 0.13 µm technology with V_{th} = 400 mV. The data memory, instruction memory and register file consume more than 70% of the total energy.

2.3.5 A 32nm 8.3GHz near-threshold voltage register file

Amit Agarwal et al. propose in [23] a variation tolerant register file, fabricated in 32 nm CMOS technology, which operates at frequencies of 8.3 GHz and consumes only 83 mW. It is a 64-entry * 32b 1-read, 1-write ported register file, which can operate in the near-threshold region, at 340 mV. Contention in register file read/write circuits limits the active minimum operating supply voltage of a microprocessor core. Other issues that appear when operating at low voltages are: the increase in variation, sub-par scaling of the minimum device width and increase in PMOS strength relative to NMOS.

This register file was implemented using dual-ended transmission gate (DETG) write cells with inherent redundancy to compensate for parameter variation and contention-free shared (CFS) keepers to improve read delay at low supply voltages. The authors show that a DETG register file provide 12% area increase, 3% power and 6% leakage penalty compared to a conventional dual-ended (DE) write memory cell, while improving the minimum allowed supply voltage (Vcc-min) by 300 mV. When operating in the near-threshold region, the proposed register file functions down to 340 mV, consuming 540 µW at 297 MHz and a peak energy efficiency of 550 GOPS/W [23].

2.3.6 A 280 mV-to-1.1 V reconfigurable SIMD vector permutation engine

Paper [24] presents an ultra-low voltage reconfigurable 4-way to 32-way SIMD vector permutation engine, that was fabricated in 22 nm tri-gate bulk CMOS technology. In order to maximize the high-performance microprocessor vector datapath utilization in multimedia, graphics and signal processing, energy-efficient SIMD permutation operations are necessary. SIMD computations require many pre-processing instructions to parallelize data before any computation is performed. The SIMD vector bit-width has been increased in many microprocessor instruction set architectures. In order to keep the execution units fully utilized, an any-to-any permute crossbar is needed. SIMD permutation engines require both high-performance at the nominal supply and energy-efficient performance in the presence of variation at ultra-low supply voltages.

The proposed SIMD vector permutation engine consists of a 32-entry x 256b 3-read / 1-write ported register file with a 256b any-to-any permute crossbar for 2-dimensional shuffle. The register file integrates a vertical shuffle across multiple entries into read/write operations. The permute crossbar is implemented using an interleaved folded byte-wise multiplexer layout. The nominal performance of the register file was measured at 0.9 V and a temperature of 50 ºC and it operates at a frequency of 1.8 GHz, with a power consumption of 106 mW. The register file can function correctly at a reduced voltage of 280 mV, with a peak energy efficiency of 154 GOPS/W. The permute crossbar is capable of functioning up to 2.9 GHz, with a power consumption of 69 mW and the lowest power consumption is obtained when operating at 240 mV. At the nominal supply voltage of 0.9 V, the permute crossbar operates at 2.3 GHz and consumes 36 mW. Peak energy efficiency occurs at a supply voltage of 260 mV and it is equal to 585 GOPS/W. In order to show the effectiveness of the 2-dimensional shuffle, a key algorithm for multimedia and signal processing workloads is mapped onto the permutation engine: a 4x4 64b matrix transpose algorithm. When running the matrix transpose algorithm, the authors obtain 40% to 53% energy savings and 25% to 42% improved peak throughput measured at 1.8 GHz and 0.9V [24].

2.3.7 A 32-nm 64-core multiprocessor chip with dual-voltage rail and half-speed units

One of the main issues that arise in multiprocessor systems operating at low voltages is the increase in effects of parameter variation, which determines significant frequency heterogeneity between and within otherwise identical cores. The authors of paper [25] present a combination of techniques designed to reduce the effects of variation on the performance and energy efficiency of near-threshold multiprocessor chips.

The main techniques presented in paper [25] are dual voltage rail (DVR), which mitigates core-to-core variation with a dual-rail power delivery system and half-speed unit (HSU), which mitigates within-core variation by halving the frequency of some functional blocks. Variation affects severely the transistor threshold voltage, which causes heterogeneity in transistor delay and power consumption within processor dies.

The DVR technique provides two power rails, each one supplying a different externally controlled voltage. Each core in the multiprocessor chip can be assigned to one of the two power supplies using a simple power gating circuit. The HSU technique permits functional units to have two possible speeds: full speed (running at the core's frequency) or half speed (running at half the core's frequency). So, the frequency of a core can be increased substantially by allowing slower units to run at half speed [25].

These techniques have been evaluated on a 32 nm 64-core multiprocessor chip, with DVR and HSU applied both independently and in conjunction. Experimental results show that high variation has a dramatic impact on system frequency. Without variation, the system is expected to run at about 400 MHz at V_{dd} = 400 mV and with a 12% V_{th} variation, the average frequency across all dies resulted at the value of 149 MHz, with a minimum of 75 MHz and a maximum of 230 MHz, for the same V_{dd}. The authors find that DVR can reduce frequency variation from 30.6% standard deviation from the mean down to 23.1%, improving system frequency with 30%. DVR alone improves the performance of the 64-core system by 30% and HSU alone by 33%. When applied in conjunction, DVR and HSU achieve together a 48% average performance improvement.

2.3.8 Comparative analysis of state-of-the-art implementations

Table 2.2 presents a comparative analysis between the state-of-the-art systems described above. The table contains the CMOS technology node of each digital system, the nominal threshold voltage and the minimum functional voltage reported by the authors. Also, two coordinates of the minimum energy point obtained by the authors are reported for each system: the associated voltage and the frequency.

Name of the system	**CMOS technology**	**Nominal threshold voltage**	**Minimum functional voltage**	**Minimum energy point**	
				Voltage	**Frequency**
Intel processor [19]	32 nm	600 mV	280 mV	280 mV	3 MHz
FFT processor [20]	180 nm	450 mV	180 mV	350 mV	10 kHz
RISC microcontroller with integrated SRAM [21]	65 nm	-	300 mV	500 mV	434 kHz
Sub-200 mV processor [22]	130 nm	400 mV	160 mV	350 mV	354 kHz
Register File [23]	32 nm	-	340 mV	340 mV	297 MHz
SIMD Vector Permutation Engine [24]	22 nm	450 mV	280 mV	280 mV	16.8 MHz
64-core Multiprocessor chip [25]	32 nm	400 mV	300 mV	400 mV	400 MHz

Table 2.2 Comparison between the parameters of various state-of-the-art sub-powered CMOS based systems

The dissipated power of the Intel processor in the sub-threshold region is only 2 mW, at a supply voltage of 280 mV. In comparison with this, the FFT processor has a much better power efficiency: it dissipated only 600 nW at 350 mV.

We can conclude that the minimum energy point of the studied systems occurs at a voltage of approximately 300 mV and the clock frequency of the processors running in this sub-threshold regime is reported in the kHz region. The minimum functional voltage of the processors depends on the CMOS technology in which they are fabricated. We notice that the minimum functional voltage cannot be reduced too much when we scale the devices deep into the nanometer zone, where the transistors’ channel lengths take values below 65 nm.

2.4 Application Areas

The potential of this research area is demonstrated mainly by the multitude of sub-domains where sub-powered circuits would bring important benefits and would have a large applicability. The target domain where sub-threshold digital circuits are suitable is represented by specific applications that don’t need high performance, but require extremely low power consumption. Several papers show that the performance in the sub-threshold region is adequate for most applications with low-to-moderate performance, meaning operating frequency ranges from 10 kHz to 100 MHz. The authors in [26] claim that sub-threshold circuits can be used in three main categories of applications. The first category is represented by energy-constrained applications that permit low performance, like microsensors, implants and RFIDs. The second category is represented by energy-constrained portable devices that must occasionally support high performance and in the third category we find systems that use sub-threshold circuits as low overhead support for high performance applications, such as standby management when strong inversion circuits are asleep.

According to [27], the main category of devices that take advantage of sub-powered circuits are medical equipment like hearing aids and pace-makers, wearable wrist-watch computation and self-powered devices. For example [30], wearable devices which process biomedical signals, like ECG, must function at frequencies lower than 1 MHz, so they are usually battery powered and have tight energy consumption constraints. Sub-threshold designs can improve the energy efficiency of such devices by several orders of magnitude, while causing a performance loss, which is not critical [30].

The authors in [28] also realize a taxonomy of the applications where probabilistic CMOS (PCMOS) technology will be suitable. They find the following two main categories: applications that benefit from probabilistic behavior at the device level and applications that can tolerate probabilistic behavior at the device level. According to the same paper, a probabilistic algorithm is defined as an algorithm “in which each step, upon repeated execution with the same inputs, could have several possible outcomes, where each outcome is associated with a probability parameter”. An important advantage of PCMOS circuits is that they can be used for applications that embody probabilistic behavior naturally like Bayesian inference (BN), Probabilistic Cellular Automata (PCA), Random Neural Networks (RNN) and Hyper Encryption (HE). All these applications have in common the notion of a core probabilistic step, which can be modeled like a probabilistic truth table.

In the second category, applications that tolerate probabilistic behavior, we find programs which can trade energy and performance for application-level quality of the solution. Digital signal processing represents a target domain for this kind of programs, where application-level quality of the solution is usually expressed using the signal-to-noise ratio (SNR) parameter. The authors of [28] succeed to prove the utility of sub-powered PCMOS circuits in this field by implementing a variant of the H.264 decoding algorithm, with filter primitives based on PCMOS. Results show that this approach contributes to significant energy savings, but degrades the quality of the resulted picture, when compared to the algorithm based on conventional CMOS.

2.5 Reliability Issues in Sub-Powered CMOS Circuits

2.5.1 Sources of variation

Process variation has become an important issue as process technology has moved to smaller and smaller feature sizes. For example, a homogeneous multi-core design can transform into a heterogeneous multi-core system due to cores exhibiting different performance characteristics due to process variation. In this situation, homogeneity can be brought back by running all the cores at the speed of the slowest core [29].

Increased sensitivity to the threshold voltage, combined with a low I_{on} / I_{off} ratio, may cause serious circuit-level robustness concerns when process variation is taken into account. Each technology generation manifests an increased vulnerability to process variations. Within-die (WID) process variations are caused by systematic effects (for example litographic irregularities) and random effects (for example varying dopant concentrations). Two important process parameters affected by variations are threshold voltage (V_{th}) and the effective channel length (L_{eff}). V_{th} and L_{eff} variations have a major impact on the fluctuations in transistor switching speed and static power consumption, which are more pronounced at NTC, than at STC. Dynamic power is also more sensitive to process variations at NTC than at STC, because it is a function of frequency and transistor delay is more sensitive to changes in V_{th} at low V_{dd} [30].

Process variation that affects low-power devices has two components: systematic and random. The systematic component is usually spatially correlated, so the amount of variation affecting neighboring devices will be the same. The effects of this type of variation can be counter-balanced by applying coarse-grained techniques such as body-bias to increase or decrease the delay [44]. Body biasing is an effective technique in the sub-threshold region due to the exponential dependence of sub-threshold current on body bias. This fact offers the possibility to eliminate performance variation, while maintaining energy efficiency. Measurements showed that body biasing is a more energy efficient global technique than V_{dd} scaling over the frequency range considered [22]. Secondly, the random component tends to have no spatial correlation and this will determine neighboring transistors to exhibit different amount of variation. The percentage that both systematic and random variation affect the propagation delay increases as supply voltage is lowered [44].

There are three major sources that may cause variations in transistor behavior. The first source is called random dopant fluctuations and it appears as a result of discreteness of dopant atoms in the channel of a transistor. Transistor channels are doped with dopant atoms to control their threshold voltage, but the number of dopant atoms in the channel decreases exponentially over generations. Therefore, it is very common for two transistors sitting side by side to have different electrical characteristics because of randomness in a few dopant atoms, which will result in variability [4].

The second source of variability is represented by sub-wavelength lithography. According to [4], since the 0.25-μm technology generation, we have used subwavelength lithography for patterning transistors. For example, fabrication processes used a 248-nm wavelength of light to pattern 0.25-μm (250-nm) and

0.18-μm transistors. The wavelength decreased to 193 nm for 130-nm technology and has since remained constant for even 65-nm transistors. Sub-wavelength lithography produces some undesired effects, such as line edge roughness, which will produce variability.

The first two sources explained above are considered static, because they occur during fabrication, but the third source of variations is dynamic because it depends on time and context. Depending on the functionality of the circuit block, the heat flux (power density), which is usually measured in Watts per square centimeter, takes different values. For example, the heat flux for an execution unit is higher than the one of a cache. A higher heat flux stresses more the power distribution grid, leading to resisitive and inductive voltage drops, which will determine dynamic time-dependent supply voltage variations, which will have a greater impact on circuit reliability if the supply voltage is scaled down. Also, a higher heat flux determines the occurrence of hot spots accross the die, which will lead to temperature variations [4].

Variations in sub-threshold circuits can lead to two types of failures: functional failure and parametric failure. Functional failures primarily occur in SRAM arrays as a result of random variations, while parametric failure can result from both random and global variations.

2.5.2 Types of errors and their physical causes

Shrinking geometries, low supply voltages and higher frequencies of operations contribute to a decrease in reliability, leading to an increase in the number of occurrences of faults [5]. The two main types of errors that affect digital circuits are hard and soft errors. Besides, faults experienced by semiconductor devices can be classified into three main categories: permanent, intermittent and transient. Permanent faults (hardware failures) are irreversible and are usually caused by manufacturing defects or device wear-out. Intermittent faults appear because of unstable or marginal hardware and they usually precede the occurrence of permanent faults [5]. Additionally, they occur repeatedly at the same location.

According to the authors of article [32], there are five common types of hard errors or hardware failure that can affect memory circuits. The most common type is represented by a single cell hard failure, a situation where a defect occurs in a single cell, making it no longer able to reliably hold data. The failure of a row or column selection circuitry can cause an entire row or column to become unreliable. But there are situations when even a so-called row-column failure may occur, causing a row and a column to fail simultaneously due to the shorting of row select and column sense lines. Another scenario is represented by the failure of a power circuit or chip select signal, which will cause the failure of the entire chip. Such hardware or permanent failures are irreversible and are usually caused by manufacturing defects, device wear-out or heavy ion radiation. Many times, these permanent faults are preceded by intermittent faults, which occur because of unstable or marginal hardware.

The authors of the same paper, [32], claim that layout strategies may cause the occurrence of additional hard-error types. Long selection lines are always a problem due to the large capacitance they have in relation to the cell capacitance or driving ability, so they cause the chip to be slow and error-prone. In order to avoid this inconvenience, large memory chips are organized into several blocks, each one with its own selection circuitry.

A transient fault caused by a single particle hit is referred to as a single-event transient (SET), while an error in a memory element that was caused either by a SET or from direct radiation hit is called a soft error or a single-event upset (SEU) [33]. Soft errors occur when highly energetic particles, like protons, neutrons, alpha particles or other heavy ions strike sensitive regions of the silicon. According to [34], “such errors are caused by three main radiation mechanisms: alpha particles emitted by trace uranium and thorium impurities in packaging materials, high-energy neutrons from cosmic radiation and low-energy cosmic neutron interactions with the isotope boron-10”. Soft errors can also be induced by electromagnetic interference, noise, capacitive coupling, power transients, crosstalk, ground bounce, IR drop and thermal fluctuations [35]. The authors in [4] claim that the increase in soft-error rate per logic state bit for each technology generation is expected to be equal to about 8 percent. Knowing that the number of logic state bits double each technology generation, following the Moore’s law, the soft-error failure-in-time rate of a chip is shown in fig. 2.5 for each technology node. We can observe that the failure rate increases dramatically as the transistor dimensions are scaling down, deep into the nanometer region.

Timing errors usually occur when a system operates at a very high data rate and are caused by timing jitter. Sampling clock fluctuations can cause an incorrect output because the signal at the output of a gate may be sampled before it reaches a steady value. Failures that are caused by timing jitter depend on gate history: they are not only dependent on the current input, but also on the previous inputs. The time at which the output should be sampled is decided by the transition with maximum delay, but in the presence of jitter, the output can be sampled before it reaches a steady value, leading to gate failure.

When a faulty component is affected by transient faults, it is assumed that this type of faults occurs at particular time steps and that they do not necessarily persist for later times. The approach according to which a failure occurs by flipping the correct result with some probability is referred to as “von Neumann type of error”.

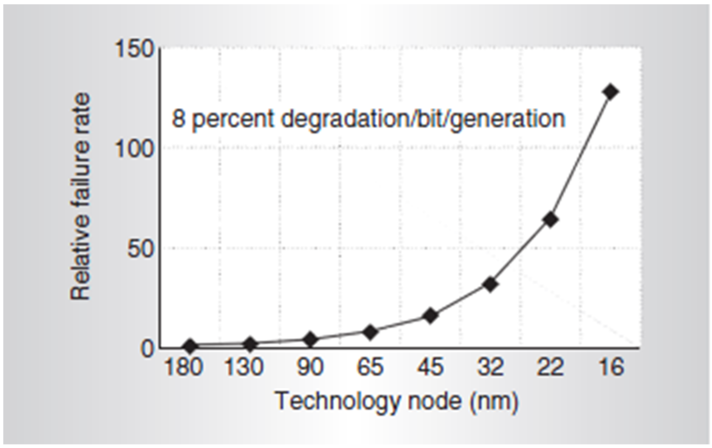


Fig. 2.5 Soft-error failure-in-time of a chip (logic and memory), [4]

Transient errors require a high level of attention because both early and recent studies of failures in digital systems demonstrated that about 90% of failures where transient in nature. Studies from IBM and DEC systems showed that over 85% of all computer failures are due to transient errors, so the level of system activity depends on the occurrence of transients [36].

Regarding the propagation of transient errors in digital circuits, we must mention the significance of three important masking factors:

1. Logical masking – the effect of a glitch present at an input of the gate is logically masked when at the other input / inputs we have a controlling value; for example a “1” glitch arrived at one input of an AND logic gate will have no effect when the other input is connected to a logic “0”.
2. Electrical masking appears when the noisy glitch is not large enough compared to the gate delay, so the gate won’t assure its propagation.
3. Latching-window masking appears when the glitch arrives too late to the input of a latch in order to be stored.

2.5.3 Fault injection techniques for reliability evaluation of digital circuits

Fault-tolerant systems at lowest possible cost represent the hot topic in digital design in the last years, as a result of the demands of the complex current market. During the design cycle of a digital system, it is important to be able to do diagnosis in the early phases, because it saves time and money when the actual system is developed. In order to evaluate the dependability attributes of a digital system, fault injection represents the most widely used technique. Fault injection can be regarded as the process of deliberate fault insertion at designated locations of the system under test, therefore it is a technique which allows the study of the behavior of the target system in the presence of faults. Fault injection techniques assure very accurate reliability estimation because they are performed on the target system itself and the injected faults are similar or identical to the ones in the working environment. The three main categories of fault injection techniques are: physical or hardware-implemented fault injection (HWIFI), software-implemented (SWIFI) and simulation-based [37][38].

HWIFI methods rely either on the disturbance of the target system with parameters of the environment, like ion radiations, voltage disturbances, either the modification of the values of the pins. Commonly used HWIFI methods are: pin-level injection, heavy-ion radiation, electromagnetic disturbances and non-destructive laser exposure. These techniques closely imitate real fault situations, but they are usually expensive and can be applied only after the physical chip is available [39].

There are several suitable moments when the fault injection process can be carried out during the design phase or the prototype phase. In the design phase, the injection of faults takes into account the system’s model developed using advanced computer-aided design (CAD) tools. In the prototype phase, a fault injection environment usually contains the circuit under test, the controller, the fault injection tool, the data collector and the data analyzer [38].

Physical and simulated fault injection at system level was performed as far back as 1980s by several researchers associated with NASA AIRLAB [36]. Fault latency distributions through hardware fault injection was performed by the authors in [40] and the dependency of error propagation on the location of faults and on the type of instruction was analyzed in [41]. If we take into consideration the microprocessor level, an analysis of the vulnerability of the Z80 microprocessor when exposed to ion-bombardment campaigns was firstly presented in [42] and the effects of transient errors on a 32-bit pipelined RISC microprocessor was studied in [43]. Physical and simulated fault injection campaigns performed by various researchers confirm the fact that transient and intermittent faults can induce computational errors: a process also called silent data corruption [5].

Simulation is used to evaluate the circuit under test during the design phase, using computer-aided design (CAD) environments. The simulation-based fault injection is used to test the effectiveness of fault-tolerant mechanisms and to evaluate the dependability, providing valuable feedback to system designers. For an effective simulation, accurate input parameters, validation of the results and suitable fault models and fault patterns are required.

Simulated fault injection (SFI) can be performed at various levels of abstraction: electrical level, logic level or functional level. At the functional level, behavioral models are used to perform fault injection, which seems to be the most cost-effective at this level. SFI techniques have been classified in two main categories: approaches that don't require any code instrumentation (i.e. simulator commands and scripts) and those that require modifications of the hardware description language (HDL) code (i.e. mutants and saboteur techniques) [49].

A hardware description language (HDL) represents a specialized computer language used to program the structure, design and operation of electronic circuits. It enables a precise, formal description of an electronic circuit, which can be used for automated analysis, simulation and testing. HDL simulated fault injection is a powerful tool to analyze the circuit behavior in the presence of faults and it is commonly performed in VHDL or Verilog. SFI can be applied as soon as a system model is available in the design phase. A saboteur is defined as a special component that alters the value or timing characteristics of one or more signals, while a mutant represents a component description which replaces the correct architecture of a module [37]. Although simulator commands do not require code intervention, they are dependent on the simulator environment capabilities and its command languages.

Saboteurs can be classified into two categories, serial and parallel, and they can be simple or complex depending on the fault pattern that is being modeled. A serial saboteur breaks the signal path between a driver output and its corresponding receiver input, while a parallel saboteur is commonly added as an additional driver for a resolved signal for the receiver [52].

Regarding the fault injection tools described in the literature, we can mention the importance of two state-of-the-art tools developed since the early ages of reliability assesment: the MEFISTO tool [52] and the VFIT tool [37]. The most relevant features of these tools are:

- Automation of the fault injection process, at different stages, which is implemented using dedicated software modules for setting up and running the simulation. For example, for the setup part the mutant generation is performed by some automated code and there are also scripts designed to run the simulated fault injection campaigns.
- Extraction and processing of error related information;

- Fault injection process divided into three main phases.

The three main phases of the simulated fault injection process performed by both MEFISTO and VFIT tools are:

- The setup phase, during which the simulation parameters are tuned. Among the parameters that can be chosen, we mention the fault model type, the fault occurrence pattern, the number of simulations to be performed and the input data vectors.
- The actual simulation phase of the circuit under test, which is described using hardware description languages. During this phase, information is collected, in order to be analyzed during the last phase.
- The results processing phase is the last phase of the process and it comprises the comparison of the faulty trace with that of a golden run (a simulation result obtained from a fault-free functioning system). During this phase, various dependability and simulation related parameters are extracted and processed.

In order to automate fault injection experiments and analyze the observations made during the experiments, several tools have been further created and they are described in the literature. One example is GOOFI, which represents an object-oriented injection tool that is designed to be portable to different platforms. An advanced tool reduces simulation time by conducting more than one injection simultaneously, and also supports event handling mechanisms and multiple system / fault models. Since fault simulation space is so large, it is difficult to obtain accurate behavior analysis in a reasonable time frame. Therefore, the fault injection tools and techniques which suit the best depend on the particularities of each target processor [39].

The effects of transient faults are dependent on processor architecture and, most probably, on fault injection methodology. During an experiment, a jet engine controller called HS1602 was upset by current and voltage transients and the results show that faults in the arithmetic unit are most likely to propagate and result in logic failure. In another experiment, RTL model of the IBM RT PC was injected with single cycle inverted transient faults and about 60-70% of injected faults were overwritten. The authors also mention that the attributes of the workload such as instruction types and control flow structures are good indicators of error behavior. Another software modeled 32-bit RISC processor, called TRIP, was tested using VHDL and the fault injection was carried out by toggling the value of randomly chosen internal state element bits. While 34% of faults were overwritten at run-time, only 23% of faults were effective, meaning the faults resulted in processor failure. These experiments prove an important ability of processors: they are capable of masking out some faults without any intended fault protection mechanism. As a conclusion, each processor has a distinct level of sensitivity to soft errors and each new design requires separate dependability evaluations in order to obtain accurate results [39].

Once a soft error occurs in a logic block of a processor, its propagation is dependent on the architecture and workload of the processor. The transient upset rate defines how often the soft errors occur and this parameter is affected by the fabrication process and circuit technology. More upsets mean higher probabilities of soft error occurrence. During one experiment, the same heavy ion was individually radiated into three units of an ERC32 processor and upset rates were different because the units employed diverse circuit types. Errors occurred mostly in the register file and some in the combinational logic. Circuits of the integer unit were more susceptible to the ions than those of floating point and memory control units. Another radiation testing on 486DX4 microprocessors shows that different implementations of a common processor architecture react in different ways when they are supposed to the same dose of radiations. The authors claim that during one experiment, when six 486DX4 processors from AMD and Intel were bombarded with radiation beams, AMD's chips were more susceptible than Intel's [39].

2.6 Probabilistic CMOS

The probabilistic switch represents the foundational model of the Probabilistic CMOS (PCMOS) technology and it realizes a probabilistic one-bit switching function [28]. These elementary probabilistic switches can be mixed in order to obtain primitive boolean functions, such as AND, OR, NOT functions. For example, a probabilistic identity function, which has the logic value of 0 as the input, will output 0 with a probability of *p* and will output 1 with a *1-p* probability. When applying the logic value of 1 to the input, the output will be 1 with a probability of *p* and 0 with a probability of *1-p.* The basic schematic of a PCMOS switch, as well as the representation of digital values 0 and 1 and the probability of error for a PCMOS switch can be found in [28] and are depicted in fig. 2.6.

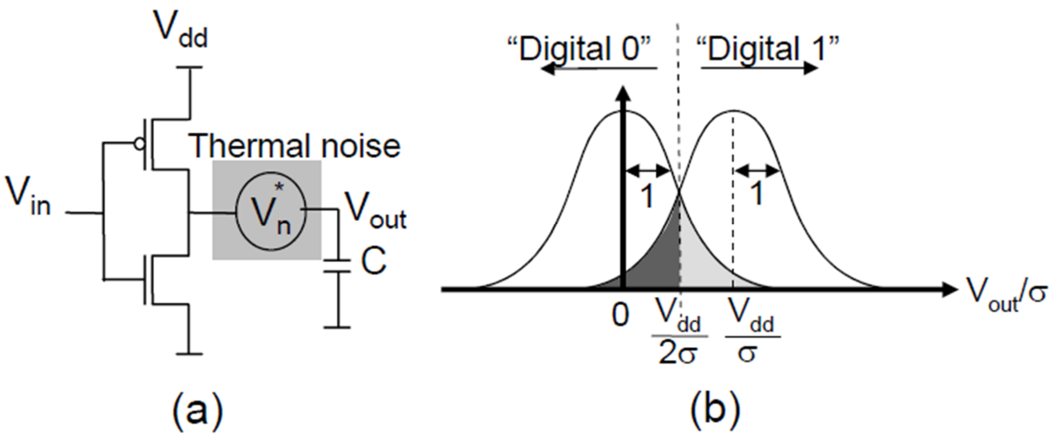


Fig. 2.6 (a) PCMOS switch; (b) Representation of digital values 0 and 1 and the probability of error for a PCMOS switch, [28]

Concerning the above picture, we can state that the PCMOS switch consists of a conventional deterministic CMOS inverter, composed of an NMOS and a PMOS transistor, with a thermal noise source coupled to the output, which gives the probabilistic behavior. When speaking about the energy consumption of a probabilistic switch, the authors of [28] claim that while a deterministic switch consumes at least *k*t*ln 2* Joules of energy, a probabilistic switch can realize a probabilistic non-trivial switching function with *k*t*ln(2p)* Joules of energy, where *p* is the probability parameter. Probabilistic switches are able to model noise-susceptible CMOS devices operating at very low voltages (in the sub-threshold region) and serve as the basic model for physical realizations of highly scaled devices, as well as emerging non-CMOS devices.

An important issue when dealing with PCMOS technology is how to control the probability *p* of correctness of a circuit, by varying the voltage. The application sensitivity depends on the probability parameters and the number of distinct probability parameters is also a concern, since it affects the number of voltage levels. If the probability of obtaining a 1 from a given PCMOS device is *p* and the probability of obtaining a 1 from another device is *q,* then a logical AND of the outputs of the two devices will be 1 with a probability of *p*q.* This technique is used to reduce the number of distinct probability parameters in a larger system [28].

2.7 Probabilistic Circuits State-of-the-Art Implementations

The following section describes a few state-of-the-art implementations of circuits that successfully benefit from the probabilistic CMOS technology. For each circuit, a short technical description is given, along with its performance (as measured by the designers of that circuit) and its domain of usability.

2.7.1 Probabilistic ripple carry adders

Paper [53] proposes a probabilistic carry adder architecture, which is applicable under a wide variety of noise assumptions, including the additive-noise assumption. This system is based on recursive equations that model accurately the propagation of carry errors. Using HSPICE simulations, the authors validate the model and demonstrate that it is able to predict multi-bit error rates of a simulated probabilistic carry adder.

The authors construct a probabilistic full adder cell (PFA) by coupling noise sources to the output terminals of the carry-out and sum of a deterministic full adder cell (FA). The system is shown in fig. 2.7. The noise sources are independent of each other and are independent of the inputs and outputs of the FA and PFA. The noise sources are simulated in HSPICE using voltage-controlled voltage sources (VCVS) with a voltage gain equal to the standard deviation σ or root-mean-square (RMS) of the noise [53].

The simulations are performed using Synopsis 90 nm technology, with a nominal voltage of 1.2 V and the lowest voltage of 0.8 V. During the experiment, 50,000 realizations of the three-bit input vector are randomly generated using Matlab. Each entry of the vector is a binary number of uniform distribution. Every 20 ns, a new variant of the three-bit vector is applied to the inputs of the FA and PFA simultaneously. Results show that for a supply voltage of 0.8 V, 1716 out of 50,000 samples taken at the seventh sum bit are incorrect. The predicted number is very close, it is 1714. The tests performed by the authors certify the accuracy and scalability, with respect to bit-length, of the proposed model [53].

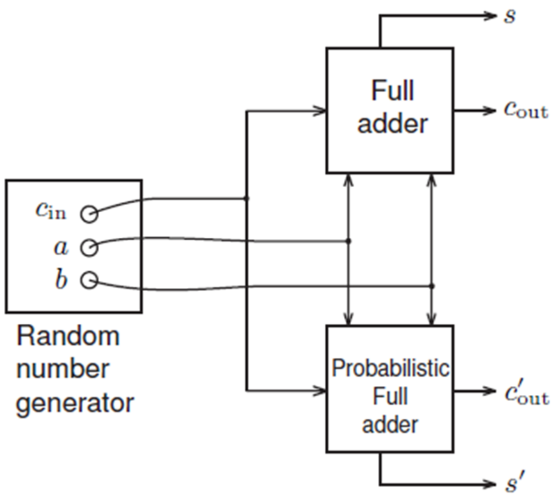


Fig. 2.7 Applying the 3 bit random generated vector to the inputs of the deterministic and probabilistic full adders simultaneously, [53]

2.7.2 SoC architectures based on probabilistic CMOS technology

The authors in [54] show that PCMOS technology provides significant improvements, both in the energy consumed as well as in the performance. An application-architecture-technology (A^2T) co-design methodology is introduced in order to provide an entirely novel family of probabilistic system-on-a-chip (PSOC) architectures. This paper focuses on PCMOS based ultra efficient (embedded) architectures and demonstrates the power of this technology in the context of a variety of applications.

A canonical PSOC architecture consists of a conventional deterministic host processor and a co-processor built using PCMOS devices, which will be used as an energy-performance accelerator. The energy consumption of an application executing on a PSOC architecture is composed of the energy consumed by the host, the energy consumed by the PCMOS co-processor and the energy cost for the communication between the processor and the co-processor. Among the applications that lead to the design of efficient PSOC architectures we can mention Bayesian Networks (BN), Random Neural Networks (RNN), Probabilistic Cellular Automata (PCA) and Hyper-Encryption (HE).

PCMOS is very efficient in computing with ultra-low energy. The energy consumed for generating one random bit using PCMOS is only 0.4 pJ. The authors succeed to demonstrate the value of PCMOS technology in the context of realizing ultra-efficient PSOC architectures, over a range of applications ubiquitous to embedded computing. The authors also claim that PCMOS has the ability of producing true random bits, while the conventional random number generators only produce pseudo-random bits.

2.8 Conclusions

Throughout this thesis, the behavior of sub-powered CMOS circuits affected by transient faults is taken into account. Their occurrence has a probabilistic nature and I have considered that the simulation-based fault injection (SFI) techniques are the most suitable for the reliability evaluation of circuits affected by this kind of errors. The main reasons for this choice are: SFI can be applied as soon as the system model is available in the design phase, SFI can be performed effectively at several levels of abstraction of a digital system and SFI can be efficiently implemented for the HDL description of the circuit. This is why the reliability evaluation techniques proposed in chapters 4, 5, 6 and 7 all rely on SFI.

3. TRANSIENT ERRORS IMPACT ANALYSIS FOR SUB-POWERED CMOS CIRCUITS, AT TRANSISTOR LEVEL (CIRCUIT LEVEL)

3.1 The Division of a Digital System into Multiple Levels of Abstraction

According to [55], an abstraction “is a simplified model of the system, showing only the selected features and ignoring the associated details”. An abstraction aims to reduce the amount of data to a level that can be managed easily. This Ph.D. Thesis aimed at analyzing the transient errors impact at all levels of abstraction of a digital system, as they are classified and described in the literature [55]: transistor or circuit level, gate level, register transfer level (RTL) and processor / system level. The division of the digital system in these levels of abstraction is depicted in fig. 3.1 and is based on the size of basic building blocks, which are the transistors, logic gates, function modules and processors, respectively.

The lowest level of abstraction, circuit level, consists of basic building blocks like transistors, resistors and capacitors. The description of the behavior of these circuits is usually made by sets of differential equations or current-voltage diagrams. The desired input-output characteristics can be derived by using analog system simulation software, belonging to the category of SPICE software. At this level of abstraction, a digital circuit is treated like an analog system, because all signals behave like continuous functions, varying over a defined time range [55].

The next level of abstraction, gate level, comprises typical building blocks like basic logic gates (AND, OR, XOR) and basic memory elements, such as latches and flip-flops. Instead of using continuous functions, like in the case of circuit level, we analyze only if the voltage of a signal is situated above or below a certain threshold and we take this in consideration as a logic “1” or a logic “0”, respectively. The input-output behavior of the circuit will be represented by a Boolean equation at this level of abstraction. This abstraction converts a continuous system to a discrete system and doesn’t take into consideration the complex differential equations anymore. At this level of abstraction, a very important parameter of a circuit is the propagation delay, which is defined as the time interval for a system to obtain a stable output response. The physical description at this level is represented by the placement of the logic gates and the routing of the interconnection wires [55].

At the Register Transfer Level abstraction, a digital system is comprised of modules constructed from simple logic gates, which include function units, such as adders, comparators or multipliers, storage components, such as registers and data routing components, such as multiplexers. The storage components use a common clock signal, which functions as a sampling and synchronizing pulse, putting data into the storage component at a particular moment, usually the rising or the falling edge of the signal. The physical layout at this level is known as the floor plan [55].

The highest level of abstraction is represented by the processor-level abstraction. The basic building blocks for this level are usually named intellectual properties (IPs) and they could be processors, memory modules and bus interfaces. At this level, time measurement is referred in terms of computation steps, which comprises the totality of operations executed between two successive synchronization points. The analysis at system level is not included in this research; it represents the subject of future work.

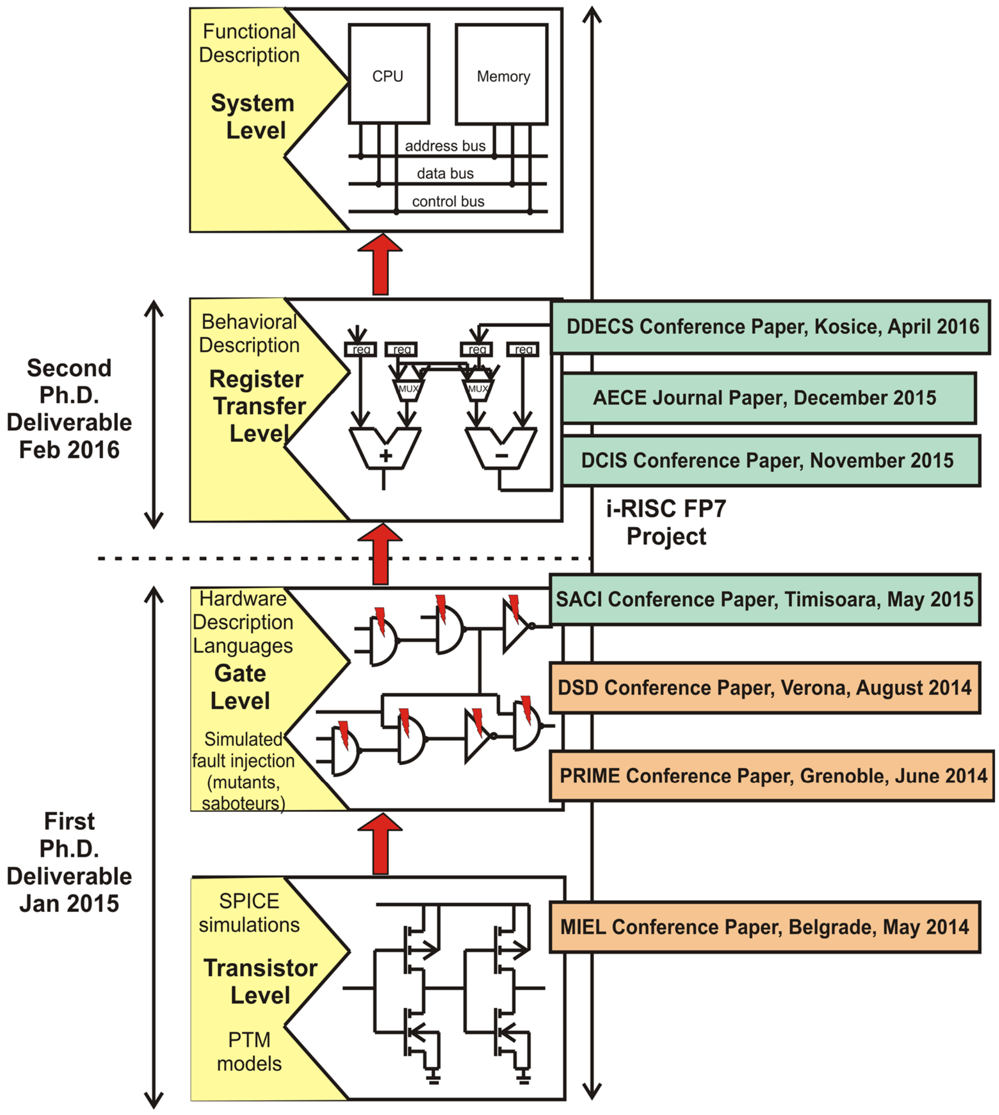

Functional Description
System Level
CPU
Memory
address bus
data bus
control bus
Behavioral Description
Register Transfer Level
reg
MUX
Hardware Description Languages
Gate Level
Simulated fault injection (mutants, saboteurs)
SPICE simulations
Transistor Level
PTM models
Second Ph.D. Deliverable Feb 2016
First Ph.D. Deliverable Jan 2015
DDECS Conference Paper, Kosice, April 2016
AECE Journal Paper, December 2015
DCIS Conference Paper, November 2015
i-RISC FP7 Project
SACI Conference Paper, Timisoara, May 2015
DSD Conference Paper, Verona, August 2014
PRIME Conference Paper, Grenoble, June 2014
MIEL Conference Paper, Belgrade, May 2014

Fig. 3.1 Research plan, at different levels of abstraction

3.2 The Basics of SPICE Analysis

Traditionally, the method used for testing electronic circuit designs consisted in building prototypes, applying various input signals or temperature changes to the circuit and then measuring its response using appropriate laboratory equipment. Unfortunately, this represents a costly and time-consuming approach for testing digital systems.

Testing the design of an integrated circuit requires a different method because the dimensions of ICs are smaller with each new generation and a breadboarded version of the intended circuit will not have the same parameters as the designed one. The parasitic components that are present in an IC differ greatly from the parasitic components present in the breadboard and signal measurements obtained from the breadboard usually do not provide an accurate representation of the signals that appear on the IC [56].

Extreme mechanical and electrical measurement precision is required in order to measure the desired signals directly on the IC itself, so this method can be applied only to specific types of measurements. Fortunately, the development of computer software that simulate the performance of an electronic circuit provided a simple and cost-effective means of verifying new designs that could improve circuit performance or power consumption [56].

SPICE (Simulation Program with Integrated Circuit Emphasis), the main industrial standard for computer-aided circuit analysis, was developed in the early 1970s at the University of California, Berkeley. SPICE is the most widespread program among the computer-aided circuit analysis software and nowadays various versions of SPICE are available for personal computers. Mainframe versions of SPICE are intended to be used by sophisticated integrated-circuit designers who require large amounts of processing power to simulate complex circuits, while commercial versions allow circuit simulation to be performed on a low-cost computer system. Among the commercial versions we can mention HSpice from Meta-Software, IG-Spice from A. B. Associates or LT Spice from Linear Technology. The main advantage of SPICE is that it simulates the behavior of electronic circuits on a computer and emulates both the signal generators and measurement equipment such as multimeters, oscilloscopes, curve tracers and frequency spectrum analyzers [56]. A SPICE based software usually offers the following features: DC operating point analysis, DC sweep analysis, transient analysis, AC analysis, Fourier analysis, temperature sweep analysis, noise and parameter sweep analysis. The last features are very useful when dealing with subpowered CMOS circuits, for which the behavior changes with the variation of multiple parameters of the transistors, like temperature of operation, amplitude and duration of noise, oxide thickness, threshold voltage.

3.3 Transistor-Level Analysis

The first step of my research has been represented by SPICE analysis of sub-powered CMOS circuits at switch-level (transistor or circuit level), along with the associated fault models extraction. In order to inject transient faults at switch-level and analyze the propagation of errors, I have created a library consisting of basic combinational logic circuits, like NOT, AND, OR, XOR gates, full adder cells, majority voters and sequential logic circuits like D-latch and transmission gate. All of the circuits were designed in CMOS technology, using the Linear Technology (LT) SPICE IV simulation software. The models used for NMOS and PMOS transistors are developed by Predictive Technology Model (PTM) [57] in the 65 nm and 45 nm technologies. The threshold voltage for the NMOS transistor is 0.423 V and for the PMOS is -0.365 V.

The proposed scenario for transistor-level analysis consists of two serial linked inverters powered by a voltage source which provides voltage supplies ranging from the 1 V nominal supply down to 300 mV. The injection of faults has been modeled by using an arbitrary behavioral voltage source intercalated on the line connecting the two gates, which generates noise signals with amplitudes following a normal distribution law. The described scenario is represented in figure 3.2:

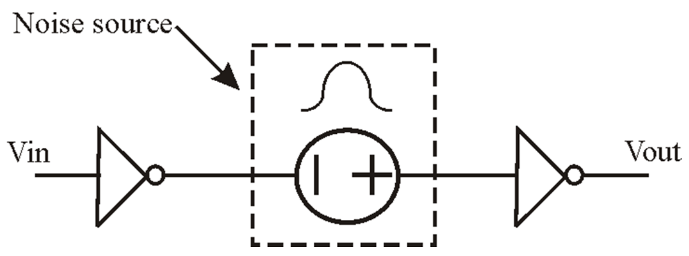


Fig. 3.2 The circuit used for Monte Carlo SPICE simulations, [35]

3.3.1 Influence of noise amplitude

The first set of experiments consisted in analyzing the effect of the amplitude of the transient noise on basic sub-powered 65 nm CMOS gates and it has been carried out using Monte Carlo simulations consisting of 50.000 individual runs. We have analyzed noise propagation for supply voltages ranging from 0.3 V to 0.8 V, with a resolution of 0.1 V, covering both the near and the sub-threshold regimes. For each supply voltage, we have used two different Gaussian distributions for the noise signal: one with sigma 0.2 and one with sigma 0.3, with a standard deviation of 0.5 V. The maximum voltage present at the output of the circuit during each simulation run was compared to the Vdd/2 threshold, in order to decide if it represents the correct output at logic level or the output is erroneous.

For the generation of noise signals, the “white” function provided by SPICE was used, which has a similar behavior with the “random” function. The Gaussian distribution was created using the function: *.function normal(nom,tol) nom+gauss(tol)*. During an individual simulation (one run), the voltage level of the input was maintained constant in order to reduce the time claimed by SPICE to process the transition between the two logic values. Hence, the simulations were separated in one set for the input signal taking the “0” logic value and one set for the input taking the “1” logic value. The duration of one simulation is 6 ns. For each set of 50,000 simulations, the results were exported from LT SPICE IV to text files and a C program was used to extract and compute the useful information.

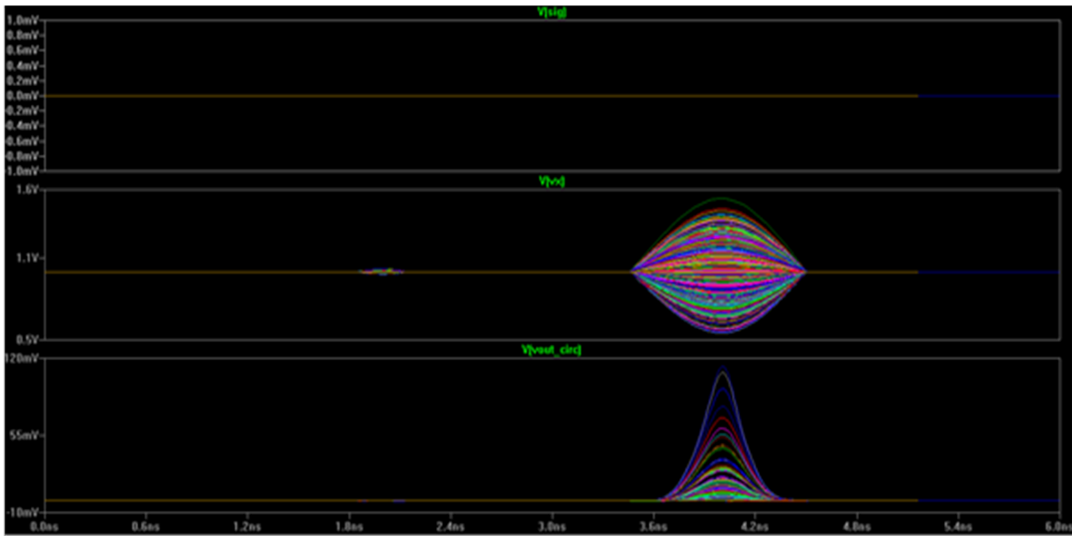


Fig. 3.3 Screen capture from LT Spice, running a set of Monte-Carlo simulations

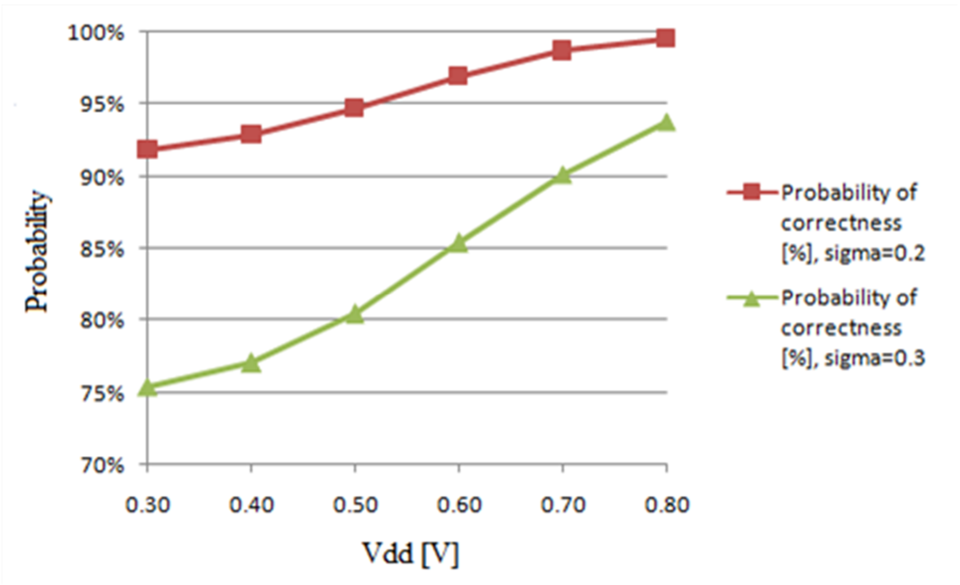


Fig. 3.4 The dependence between Vdd and the probability of correctness, [35]

Fig. 3.3 represents a screen capture showing the LT Spice medium, during the execution of one set of 50,000 simulations.

The results of the Monte-Carlo simulations are plotted in fig. 3.4. The decrease of the supply voltage has a considerable effect on the degradation of the overall gate reliability. Therefore, when the gates are supplied at very low voltages, even insignificant noise may produce the bit-flip of the logic output. These results were predictable because the decrease of the supply voltage has a negative impact on logic gates' noise margins, which will lead to an increased susceptibility of the circuit to transient errors.

3.3.2 Influence of noise pulse width

We have performed a second set of experiments on the same circuit, which aimed to determine the minimum pulse width for which the noise will propagate through one or two logic gates. For this one, we have used the 45 nm low-power PTM model and we performed the analysis on a two inverter chain and a two 2-input NAND gates chain, respectively. The supply voltage has been ranged between 0.2 V and 0.7 V, with a resolution of 0.1 V, in order to cover both near and sub-threshold regions. For this analysis, we have varied the PMOS transistor width with respect to NMOS transistor width, in order to simulate the unequal PMOS / NMOS stages found in many logic gates. For the inverter, we have considered the PMOS having the width equal, double or four times greater than the width of the NMOS transistor, which was set to 500 nm. Besides, for the NAND gate, we considered the width of the PMOS being half, equal or double with respect to the NMOS width, which was set to 1000 nm.

	width_PMOS = width_NMOS				width_PMOS = 2 * width_NMOS			
Vdd [V]	Minimum pulse width[ns] "0" Glitch		Minimum pulse width[ns] "1" Glitch		Minimum pulse width[ns] "0" Glitch		Minimum pulse width[ns] "1" Glitch	
	Gate 1	Gate 2	Gate 1	Gate 2	Gate 1	Gate 2	Gate 1	Gate 2
0.2	630	1530	390	1040	480	1290	610	1480
0.3	57	156	36	103	43	128	56	150
0.4	5.25	15.7	3.4	10.1	4	12.7	5.2	14.9
0.5	0.6	1.85	0.35	1.26	0.45	1.49	0.6	1.75
0.6	0.1	0.3	0.06	0.19	0.08	0.24	0.1	0.28
0.7	0.03	0.09	0.02	0.05	0.03	0.07	0.03	0.08

Table 3.1 – Minimum pulse width which ensures glitch propagation through two inverters (a)

	width_PMOS = 4 * width_NMOS			
Vdd [V]	Minimum pulse width[ns] “0” Glitch		Minimum pulse width[ns] “1” Glitch	
	Gate 1	Gate 2	Gate 1	Gate 2
0.2	410	1205	1050	2290
0.3	37	116	95	240
0.4	3.4	11.3	8.8	24.1
0.5	0.38	1.32	1	2.85
0.6	0.06	0.21	0.16	0.46
0.7	0.02	0.07	0.05	0.12

Table 3.2 – Minimum pulse width which ensures glitch propagation through two inverters (b)

	width_PMOS = 0.5 * width_NMOS				width_PMOS = width_NMOS			
Vdd [V]	Minimum pulse width[ns] “0” Glitch		Minimum pulse width[ns] “1” Glitch		Minimum pulse width[ns] “0” Glitch		Minimum pulse width[ns] “1” Glitch	
	Gate 1	Gate 2	Gate 1	Gate 2	Gate 1	Gate 2	Gate 1	Gate 2
0.2	1110	2390	795	2080	820	1990	1180	2770
0.3	103	251	82	212	74	204.5	119	291
0.4	9.3	25.5	8.2	21	6.9	20.7	11.9	29.3
0.5	1.06	3.02	0.97	2.46	0.78	2.44	1.4	3.45
0.6	0.17	0.49	0.16	0.4	0.13	0.4	0.22	0.57
0.7	0.06	0.15	0.04	0.11	0.04	0.12	0.06	0.15

Table 3.3 – Minimum pulse width which ensures glitch propagation through two NAND gates (a)

We have applied both „0” and „1” glitches and we have assisted to an exponential increase of the minimum pulse width with the decrease in supply voltage. The results are presented in tables 3.1 to 3.4. For high voltages (0.7 V in our case), almost all common glitches, with durations between 100 ps and 300 ps, propagate through one or even both gates. Nevertheless, for sub-threshold voltages, the duration of the glitches must be very high in order to propagate through even one single gate.

Results presented in tables 3.1 – 3.4 show that “0” glitch propagation is favored when the PMOS drive strength is greater than the NMOS drive strength. In an analogous way, the “1” glitch propagation is favored when the NMOS drive strength is greater with respect to PMOS drive strength.

	width_PMOS = 2 * width_NMOS			
Vdd [V]	Minimum pulse width[ns] “0” Glitch		Minimum pulse width[ns] “1” Glitch	
	Gate 1	Gate 2	Gate 1	Gate 2
0.2	670	1895	1930	4080
0.3	61	191	200	441
0.4	5.7	19.3	19.2	45.1
0.5	0.64	2.3	2.25	5.33
0.6	0.1	0.38	0.37	0.87
0.7	0.03	0.11	0.1	0.23

Table 3.4 – Minimum pulse width which ensures glitch propagation through two NAND gates (b)

In this paragraph, we have analyzed the impact of the pulse width and amplitude typical to transient errors in CMOS circuits operating at very low supply voltage. Concerning the amplitude, the performed simulations confirm that lowering the supply voltage will determine an important decrease of the reliability, due to the narrow noise margins specific for sub-threshold and near-threshold regimes of operations. From tables 3.1 to 3.4 we also observe that the propagation of transient faults is detained for circuits operating at lower Vdd, mainly due to the electrical masking effects. Outside the simulation environment, real life circuits will augment this electrical masking effect due to the interconnecting wires between logic gates. Hence, we can state that narrow glitches propagate better through logic gates supplied at high voltages and they hardly propagate through logic gates functioning in the sub-threshold regimes.

As a conclusion, the effect of glitches generated by thermal noise, radiation or electromagnetic interference have a bounded influence on the overall reliability of circuits operating at sub and near threshold voltages. The work presented during sections 3.3.1 and 3.3.2 of this document has been published in 2014, in conference paper [35].

On the other hand, the authors of paper [48] performed some additional simulations at circuit-level. They considered NOT,NAND, AND, Majority Voters and XOR gates, in 45 nm CMOS technology, which were simulated using HSPICE. These circuits have been tested under different supply voltage and temperature variations: the considered levels for the supply voltage were 0.25 V, 0.3 V and 0.35 V, while the values chosen for the temperature were 25ºC, 50ºC and 75ºC. The threshold voltages for the SPICE transistor models are -0.302 V for PMOS and 0.322 V for NMOS [49][58].

Their work consisted in performing Monte-Carlo simulations of 10.000 runs for each parameter, considering both supply voltage variations and process variations. For supply voltage variations, a Gaussian distribution with 0.05 V deviation and sigma 1 has been selected. Whereas, for process variations, two categories of parameters have been considered: threshold voltage and oxide thickness. The threshold voltage has been varied using a Gaussian distribution with 0.05 V deviation and sigma 1, while the oxide thickness has been varied using a Gaussian distribution with 10% deviation and sigma 3. Each gate permits four identical gates as output loads. The rise and fall times of the inputs are 0.1 ns. Delay has been considered as the time gap between input cross half of the supply voltage and output cross half of Vdd.

Figure 3.5 exemplifies the results extracted for the 2-input NAND gate, in the case of a “00” to “11” input switch. The authors claim that the probability of a correct output depends on the considered gate delay (a greater delay results in a higher probability of a correct switch), the type of switching occurring at the gate inputs and the supply voltage. Their results also show that very large gate delays (more than 5 ns for a gate) yield a correct output.

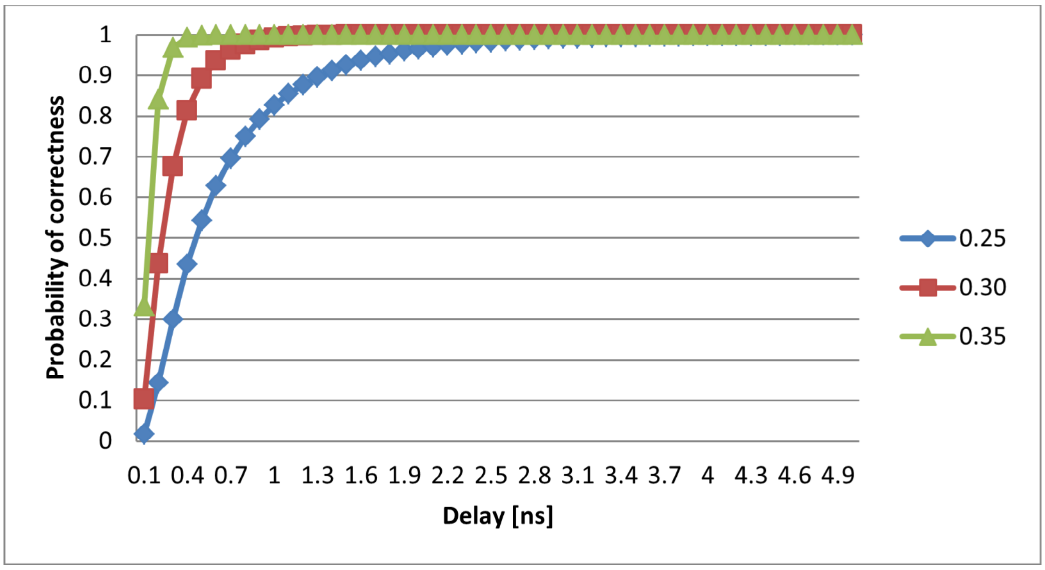


Fig. 3.5 Probability of correctness function of considered gate delay (expressed in ns) for 2-input NAND gate, Vdd={0.25V, 0.3V and 0.35V}, temperature 25ºC, for 00 to 11 input switch, [49]

For these experiments, the authors have considered several assumptions: the input transition from 11 to 10 has been considered the same as 11 to 01; skew at input signals has not been taken into account.

Furthermore, the authors in [48] have performed delay dependent reliability evaluation of the NAND-based D flip-flop. The results for both charging and discharging processes at a supply voltage of 0.3 V are depicted in fig. 3.6. The delay has been considered as the time gap between clock edge cross half of the supply voltage and output cross half of the supply voltage.

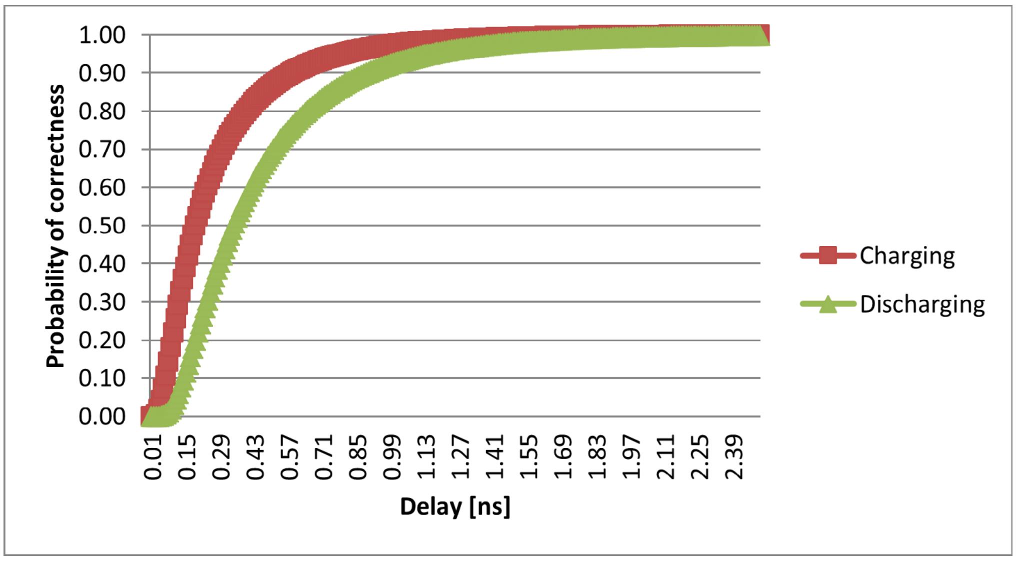


Fig. 3.6 Probability of correctness for D flip-flop, at 0.3 V, dependent on delay constraints, [50]

3.4 Concluding Remarks and Contributions for Transistor-Level Analysis

In this chapter, the impact of the amplitude and duration of pulses has been analyzed, using several sets of SPICE Monte-Carlo simulations for 45 nm and 65 nm low-supply voltage CMOS circuits.

The contributions and conclusions at transistor level are:

- the effects of the amplitude and pulse width typical to transient errors in the CMOS circuits operating at low supply voltages are analyzed;
- from the noise amplitude point of view, a decrease in reliability is noticed, with the decrease of the supply voltage, for different noise assumptions, with normal distributions of the amplitude;
- regarding the propagation of transient faults, gates operating at low supply voltages show increased resilience to glitches, despite the fact that the noise margins of the circuits are diminishing; simulations have proven that glitches with shorter duration propagate better for higher supply voltages.

4. TRANSIENT ERRORS IMPACT ANALYSIS FOR SUB-POWERED CMOS CIRCUITS, AT GATE / LOGIC LEVEL

4.1 The Motivation for Using Simulated Fault Injection at Gate Level

Process-voltage-temperature (PVT) variations have a greater impact on transistors, as they scale more and more, down into the nanometer domain. Two process parameters are mainly affected: the threshold voltage and the effective channel length, which determine modifications in the transistor switching speed and the static power consumption. The first effect means a reduction of the operating frequency of the circuit, which is reflected by the time window needed by the transistor to perform a correct switch.

As stated in [44], devices may experience a threshold voltage shift and therefore a delay shift, due to systematic and random process variations. The systematic component is usually spatially correlated, so the amount of variation that affects neighboring components will be approximately equal in most of the cases. One of the methods for countervailing this problem is to use the body-bias technique, which increases or decreases the delay shift. On the other hand, the random component, usually caused by random dopant fluctuations and line edge roughness, increases as transistors are scaled more and more. As supply voltage is scaled, the percentage that both systematic and random variation affect the delay increases, causing an important performance loss in the sub-threshold region. Simulations carried out by the authors in [44] for the 65 nm technology node show energy per operation can be minimized by operating in the near-threshold region, but process variation exhibits an increase from 280% to 690%. For synchronous systems, which use a global clock, performance is limited by the critical path (the slowest path), so we must take into account this increase in variation by adding a timing margin to the clock frequency. But this increases delay and static leakage power per operation, affecting the energy efficiency gained by operating in this region.

The behavior of the sub-powered CMOS circuits must be analyzed in a stochastic manner, knowing that the correct logic value of the output of a logic gate will occur with a probability less than one. Therefore, efficient methods for the dependability evaluation of digital systems based on sub-powered CMOS gates are required. Such techniques have been thoroughly investigated in the literature, the prevalent approach being based on fault injection. As defined in [45], fault injection represents the "*validation technique of the dependability of fault tolerant systems, which consists in the accomplishment of controlled experiments, where the observation of the system's behavior in presence of faults is induced explicitly by the written introduction (injection) of faults in the system*".

The dependability parameters of digital systems have been thoroughly investigated in the state-of-the-art literature, the most widely used method being based on fault injection. Fault injection techniques are classified in three main categories: (i) Hardware Implemented Fault Injection (HWIFI), which is accomplished at physical level by disturbing the hardware with different stimuli like radiation or electromagnetic interference, (ii) Software Implemented Fault Injection (SWIFI), which reproduces at software level the errors that would have been produced upon the presence of faults in the hardware or software and (iii) Simulated Fault Injection (SFI), which implies the simulation of the system under test in another computer system [45]. The third category, simulated fault injection, is the preferred method when dealing with reliability assessment, because it permits the identification of design flaws in the early stages of the development of a digital system, bringing important savings in terms of time and cost. Simulated fault injection (SFI) techniques have been used by many researchers, for the dependability assessment of various circuits, ranging from SRAM based FPGAs [46] to quantum circuits [47].

According to [45], simulated fault injection techniques can be implemented either with simulator commands, either with HDL code modification. Simulator commands can be used to modify the values of the model signals and variables, without altering the HDL code. The sequence of commands needed to perform the fault injection for both transient and permanent faults can be included in a macro. The advantage of using a macro is that the parameters of the fault injection process, like injection place, injection instant, fault duration or fault value, can be varied without modifying the command code. On the other hand, the techniques based on HDL code modification change the model, by adding saboteurs or using mutants of the model components.

4.2 Mutant-Based Gate Level Simulated Fault Injection for Sub-Powered CMOS Combinational Circuits

In this chapter, I will present a methodology for transient errors impact analysis at gate level, for combinational circuits. Using the results provided by the circuit level analysis, we have derived probabilistic fault models with different accuracies, which have been used in order to develop a mutant-based simulated fault injection methodology for reliability evaluation of sub-powered CMOS combinational circuits. This work has been published in 2014, in conference paper [49] and in i-RISC project deliverable D2.1 [50].

Considering the probabilistic nature of faults occuring in CMOS circuits, operating at low or very low supply voltages, we have derived the following four fault models:

1. Gate Output Probabilistic model (GOP) – For this model, we have considered that a faulty output may appear at any time during the activity of the gate, independent of the input pattern, switching type or previous outputs. Mainly, two factors can justify such a faulty behavior: a random bit-flip of the output, caused by a single event upset, which could be produced by radiation, electromagnetic interference, etc or the inability of the gate to terminate the switching process in a

given time window. Taking into consideration that some undesired bit-flips may be masked during the propagation through the circuit, the effect on the overall circuit reliability may be diminished in some cases.

2. Gate Output Switching probabilistic model (GOS) – This fault model considers that the probabilistic behavior occurs only when the gate switches, regardless of the switching type. Each time the gate performs a switch of the output from logic „0” to logic „1” or backwards, a fault may appear with a probability dependent on the supply voltage, functioning temperature and imposed delay constraint.

3. Gate Output Switching Type probabilistic model (GOST) – This fault model is similar to the previous one, but more complex, because it considers different probabilities for charging and discharging processes. We have adopted this approach because it reflects a situation found in many physical implementations of CMOS circuits, where nMOS and pMOS stages are asymmetrical in terms of drive strength. If we consider the case with balanced stages, having the same drive strength, this fault model becomes equivalent with the previous one (GOS).

4. Gate Input Switching Probabilistic model (GISP) – For this model, we have considered different probabilities for each of the input switching combination. We have adopted this approach because each distinct input induces the turn-on or turn-off of a pair of nMOS/pMOS transistors contained by the gate. This is the most complex fault model developed in this chapter and it simulates the probabilistic behavior of sub-powered CMOS gates with a high level of accuracy.

For all of the above fault models, the probabilities of failure are equal to a value consistent for a considered supply voltage, temperature and delay, according to the data obtained for the transistor-level analysis described in chapter 3 of this PhD thesis.

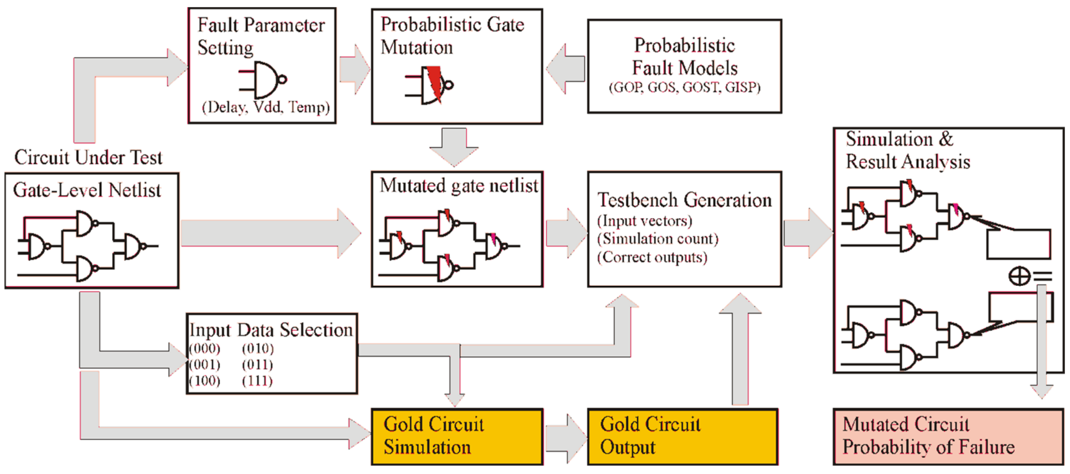


Fig. 4.1 The simulated fault injection methodology for reliability analysis of sub-powered CMOS circuits, [49]

The simulated fault injection methodology was developed in Verilog hardware description language and consists of two main phases: (a) setup phase and (b) simulation and result analysis phase.

For the setup phase, we can highlight the following steps:

1. Fault parameter settings – For each gate of the desired design, we have set the values of the three discussed parameters: power supply voltage (Vdd), temperature and gate delay. Choosing an individual set of parameters for each gate of the design offers the possibility to simulate different scenarios, like unbalanced delay paths, multiple voltage islands or asymmetrically heated regions of the circuit.
2. Probabilistic gate mutation – Depending on the gate fault parameters and the desired fault model, the associated mutant is selected for each gate, which allows the generation of the mutated circuit netlist.
3. Input data selection – three of the four proposed fault models are data dependent, so the data patterns applied at the inputs of the circuit play an important role.
4. Gold circuit simulation – The simulation of the correct circuit with the selected input data must be performed, in order to further compare the gold outputs with the fault affected outputs.
5. Testbench generation – The testbench unit was developed to assure both simulation control and results analysis, therefore it uses the input data vectors, the correct outputs and the number of simulations in order to calculate the reliability parameters.

Due to the probabilistic nature of faults and to the high level of desired accuracy, a large number of simulations has been performed. The simulation phase has been done almost simultaneously with the results analysis phase, because after each run the resulted outputs and the gold outputs have been compared.

Our proposed simulated fault injection technique uses the mutant-based approach, by developing a mutant architecture associated to each of the above fault models. The architecture of each mutant comprises a random number generator and an error insertion module which inserts faults according to a probabilistic function, which depends on three parameters: supply voltage, temperature and delay. These parameters can be tuned independently for each gate of the design, allowing a high level of flexibility.

Fig. 4.3 depicts the pseudo-code associated to each of the four mutant architectures. The algorithm of mutants *c* and *d* includes a random number generation step, which is performed for each type of output or input switching, respectively, with the purpose to generate an independent failure condition for each considered case.

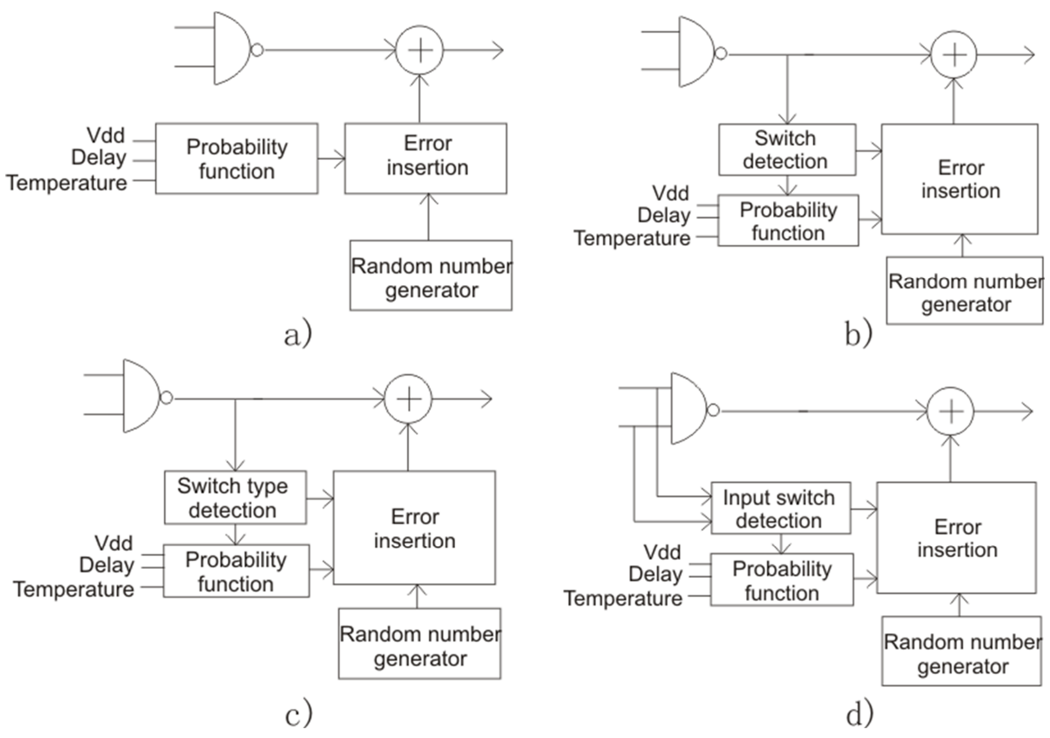
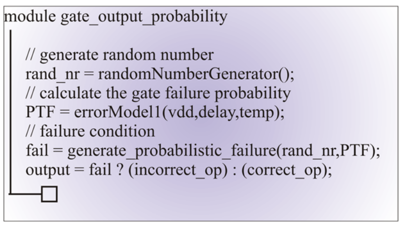


Fig. 4.2 The architectures of the mutants associated with the 4 fault models –(a) GOP, (b) GOS, (c) GOST, (d) GISP

The proposed SFI methodology has been applied for 6-bit ripple carry adders (RCA) and carry select adders (CSeA), which have been implemented using only 2-input NAND gates. The particular 6-bit configuration of the adder was chosen because it represents the basic building block for variable node units and check node units of LDPC decoders. The usual quantization of $\breve{\gamma}_i$ is 6-bits, so the operations performed by the processing units have a 6-bit width.

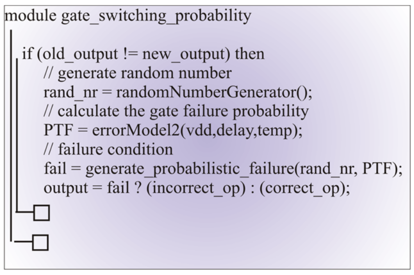
We have performed several simulation campaigns using Modelsim 10.05 SE commercial simulator on a computer with Intel Core i5 processor at 2.4 GHz, 4 GB of main memory, running Windows 7 OS. Each campaign consisted in applying 16.000 test vectors to the inputs of each circuit under test, for each of the fault models GOS, GOST and GISP, meaning a total number of 48.000 runs / campaign. We have extracted two reliability parameters: the probability of failure of each bit of the result and the overall probability of failure for the entire circuit.

For the first two simulation campaigns, we have considered the same delay and supply voltage for all the gates of the design. On the other hand, for the next two campaigns we have considered a more realistic situation: the gates on the critical path (in this case the carry chain) have the smallest delay, while the gates on the other paths have larger delays. We have considered the following cases for our simulations: (i) 6-bit RCA and CSeA with same Vdd and same delay for all gates, (ii) 6-bit RCA and CSeA with same Vdd and different delays for different paths, (iii) 6-bit RCA and CSeA in triple modular redundancy (TMR) configuration with same Vdd and different delays and (iv) 6-bit RCA and CSeA in TMR configuration, with higher Vdd for the voter and different delays. The last case aimed to simulate the situation of integrated circuits with multiple voltage islands, where critical operations in terms of reliability and performance are executed by modules with higher Vdd.



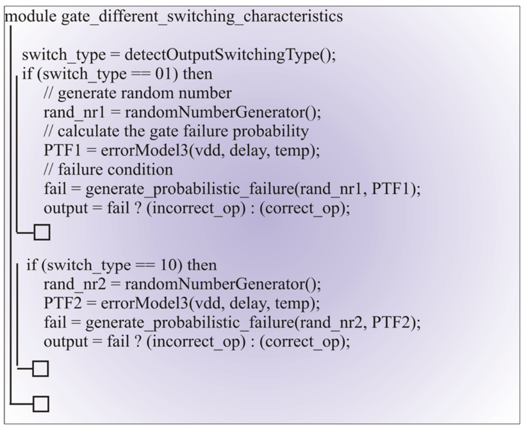
```
module gate_output_probability
    // generate random number
    rand_nr = randomNumberGenerator();
    // calculate the gate failure probability
    PTF = errorModel1(vdd,delay,temp);
    // failure condition
    fail = generate_probabilistic_failure(rand_nr,PTF);
    output = fail ? (incorrect_op) : (correct_op);
```

a)



```
module gate_switching_probability
    if (old_output != new_output) then
        // generate random number
        rand_nr = randomNumberGenerator();
        // calculate the gate failure probability
        PTF = errorModel2(vdd,delay,temp);
        // failure condition
        fail = generate_probabilistic_failure(rand_nr, PTF);
        output = fail ? (incorrect_op) : (correct_op);
```

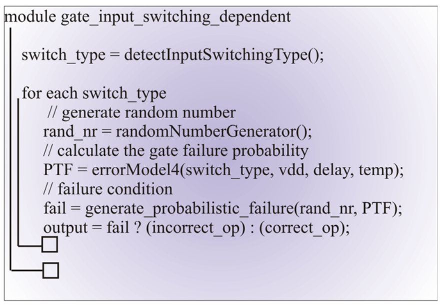
b)



```
module gate_different_switching_characteristics
    switch_type = detectOutputSwitchingType();
    if (switch_type == 01) then
        // generate random number
        rand_nr1 = randomNumberGenerator();
        // calculate the gate failure probability
        PTF1 = errorModel3(vdd, delay, temp);
        // failure condition
        fail = generate_probabilistic_failure(rand_nr1, PTF1);
        output = fail ? (incorrect_op) : (correct_op);

    if (switch_type == 10) then
        rand_nr2 = randomNumberGenerator();
        PTF2 = errorModel3(vdd, delay, temp);
        fail = generate_probabilistic_failure(rand_nr2, PTF2);
        output = fail ? (incorrect_op) : (correct_op);
```

c)



```
module gate_input_switching_dependent
    switch_type = detectInputSwitchingType();

    for each switch_type
        // generate random number
        rand_nr = randomNumberGenerator();
        // calculate the gate failure probability
        PTF = errorModel4(switch_type, vdd, delay, temp);
        // failure condition
        fail = generate_probabilistic_failure(rand_nr, PTF);
        output = fail ? (incorrect_op) : (correct_op);
```

d)

Fig. 4.3 The pseudo-code of the mutants associated with the 4 fault models – (a) GOP, (b) GOS, (c) GOST, (d) GISP

Fault Model	Delay [ns]	Vdd [V]	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time [s]
			6	5	4	3	2	1	0		
GOS	1.5	0.35	1.01	2.34	3.05	2.88	2.74	1.90	1.66	10.16	1
		0.30	6.38	13.50	16.53	16.72	15.63	11.53	10.84	50.09	2
		0.25	22.63	39.07	42.90	42.28	43.79	37.58	35.80	90.16	2
GOST	1.5	0.35	0.97	2.03	2.56	2.54	2.39	1.88	1.86	9.46	1
		0.30	5.23	13.19	14.59	15.19	14.14	10.48	12.54	49.53	1
		0.25	21.25	39.09	41.06	42.56	42.73	38.19	37.27	90.03	1
GISP	1.5	0.35	0.99	2.94	3.25	3.44	3.31	2.51	2.60	12.96	1
		0.30	5.50	14.22	17.48	17.66	17.79	12.56	15.59	55.44	1
		0.25	20.08	39.29	41.18	43.59	43.74	41.29	39.79	90.48	1

Table 4.1 – Simulation results for 6-bit RCA, for the case of equal delay on all gates

Fault Model	Vdd [V]	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time [s]
		6	5	4	3	2	1	0		
GOS	0.35	3.34	7.27	7.66	6.88	5.76	3.21	3.44	20.44	1
	0.30	12.39	25.73	26.02	25.28	21.58	12.78	13.77	64.03	1
	0.25	35.61	48.42	45.68	46.68	46.16	38.21	45.53	92.41	2
GOST	0.35	3.10	6.50	6.39	5.97	5.29	2.09	3.54	18.53	2
	0.30	11.26	24.14	24.21	24.16	21.49	10.78	14.91	62.16	2
	0.25	31.89	46.72	44.44	46.26	46.03	36.71	44.85	92.03	1
GISP	0.35	3.11	7.73	8.41	8.26	6.76	3.06	4.75	22.73	1
	0.30	10.99	24.79	25.81	25.96	21.89	11.94	17.64	65.64	2
	0.25	33.38	46.11	45.63	46.84	47.01	39.16	46.66	92.31	1

Table 4.2 – Simulation results for 6-bit RCA, for the case of different delays: 1 ns for carry chain gates and 1 – 4 ns for sum gates chain

Analyzing the results of the simulations from tables 4.1, 4.2, 4.3 and 4.4, we notice that the CSeA configuration has better reliability with respect to the RCA configuration. The TMR set-up, with all modules operating at the same supply voltage, does not significantly improve the RCA reliability (table 4.5). For the 6-bit adders that were tested, we can conclude that the most significant 3 bits of the sum are the most error prone, while the least significant 2 bits and the carry-out bit represent the most resilient ones. Regarding the simulation time, the correct 6-bit RCA required 0.5 s, the same as the simulation of the correct 6-bit CSeA. The gold 6-bit RCA in TMR configuration required a simulation time of 1 s. It results that the proposed approach has a simulation overhead of 2 to 4 times higher than the gold circuit, mainly due to two factors: the mutant-based simulation mechanism and the time required by the result analysis step, which is performed almost simultaneously with each run.

Fault Model	Vdd [V]	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time [s]
		6	5	4	3	2	1	0		
GOS	0.35	1.39	3.13	3.69	3.99	2.91	1.79	1.66	12.17	1
	0.30	8.73	15.84	18.90	19.04	15.01	11.24	11.03	55.59	1
	0.25	29.81	39.71	42.14	44.28	42.28	37.73	36.83	91.09	2
GOST	0.35	1.35	2.73	3.08	3.26	2.32	1.58	1.91	11.35	1
	0.30	6.96	14.23	16.51	18.29	14.34	11.28	12.99	54.36	1
	0.25	26.28	36.64	40.93	44.69	42.66	38.75	37.21	90.64	2
GISP	0.35	1.14	3.23	4.04	3.98	3.10	2.29	2.56	14.13	1
	0.30	6.53	15.95	20.18	21.51	17.80	13.17	15.69	59.55	1
	0.25	26.09	39.94	41.50	45.06	45.11	40.64	39.85	91.70	1

Table 4.3 – Simulation results for 6-bit CSeA, for the case of equal delay on all gates: 1.5 ns

Fault Model	Vdd [V]	Sum Bits Failure Probabilities							Circuit Failure Prob.	Sim. Time [s]
		6	5	4	3	2	1	0		
GOS	0.35	3.60	7.31	7.67	6.94	7.51	3.71	3.21	23.87	1
	0.30	13.73	26.85	26.70	26.84	26.78	16.47	15.73	71.61	1
	0.25	39.25	46.47	45.81	46.74	47.75	43.70	45.66	92.69	2
GOST	0.35	2.71	7.05	6.72	6.22	6.96	3.06	3.53	22.36	2
	0.30	12.31	24.28	24.73	26.12	26.99	15.35	16.57	69.98	1
	0.25	35.30	43.57	43.62	46.28	47.40	42.72	44.12	92.60	2
GISP	0.35	3.16	8.73	8.28	8.01	9.53	3.96	4.81	27.63	1
	0.30	11.04	25.41	26.74	28.61	29.29	16.79	19.54	73.05	2
	0.25	35.74	46.54	44.00	46.59	47.67	44.83	44.06	92.56	2

Table 4.4 – Simulation results for 6-bit CSeA, for the case of different delays: 1 ns for carry chain gates, 1 – 2 ns for sum chain gates and 1.5 ns for multiplexer gates

Fault Model / Delay [ns]	Vdd [V]	Sum Bits Failure Probabilities							Circ. Fail. Prob.	Sim. Time [s]
		6	5	4	3	2	1	0		
GOS / Carry chain 1 Sum gates 1-4 Voter 2	0.30	10.99	20.72	20.99	17.88	14.71	7.99	8.15	56.81	5
GOST / Carry chain 1 Sum gates 1-4 Voter 2	0.30	12.46	20.33	19.84	17.83	14.46	8.33	9.87	57.09	4
GISP / Carry chain 1 Sum gates 1-4 Voter 2	0.30	10.29	21.14	21.74	20.61	16.33	7.81	10.73	57.88	4
GOS / Carry chain 1 Sum gates 1-4 Voter 2	Adder 0.3 Voter 0.35	8.05	18.69	19.19	17.02	12.37	4.75	6.14	48.23	5
GOST / Carry chain 1 Sum gates 1-4 Voter 2	Adder 0.3 Voter 0.35	8.83	17.53	17.71	16.21	11.95	4.83	6.11	46.39	4
GISP / Carry chain 1 Sum gates 1-4 Voter 2	Adder 0.3 Voter 0.35	9.16	18.60	20.09	18.85	13.79	5.55	8.56	50.94	4

Table 4.5 - Simulation results for 6-bit RCA in TMR configuration

4.3 An Alternative Methodology for Reliability Assessment at Gate Level

Section 4.2 has presented a mutant-based simulated fault injection methodology, carried on at the gate level abstraction of combinational circuits. This methodology has proposed several mutant architectures with different accuracies, based on the probabilistic results provided by the circuit level analysis and it has been published in paper [49]. The simulation campaigns have been performed for 6-bit ripple carry adders (RCA) and carry-select adders (CSeA), designed using only 2-input NAND gates. The fault injection process has been described in detail and the resulted probabilities of failure have been analyzed.

In order to validate the previous technique's results, a new methodology for reliability assessment is proposed, based on simulator commands and scripts. Previously determined probabilities of failure of logic gates are used in order to perform simulated fault injection campaigns with the new approach on the same type of adders. The new methodology is implemented using two different approaches: one that uses a dedicated Verilog module in order to decide the moment when faults are injected and another one based entirely on simulator commands. The second approach doesn't bring any overhead to the HDL code, so it requires lower computational resources.

The proposed methodology, based on simulator commands and scripts, uses the probabilities of failure for basic logic gates supplied at very low voltages, derived in [49] as a function of delay. Basic logic gates supplied at 0.25, 0.30 and 0.35 V have been simulated in [49], applying a Gaussian distribution for the supply voltage and process variations (thickness of oxide and threshold voltage). The SPICE simulations results demonstrated that the probability of correctness of the output of the gate increases when considering a larger delay, which means higher performance penalty.

The proposed SFI methodology consists of three main phases, which compose the block diagram depicted in fig. 4.4: the set-up phase, the actual simulation and the results analysis phase. The script is tuned as a function of the parameters of the faults that are injected in the design, during the set-up phase. We have considered that a fault occurs on the output of a logic gate with a certain probability, which depends on the supply voltage, temperature and delay of the gate, as stated in [49]. For the current experiment, contrary to the one in [49], the transitions that occur at the inputs of the gate are not taken into account. The fault is injected only at the output of the gate, according to the requested probabilities, using two equivalent methods which lead to similar probabilistic faulty traces:

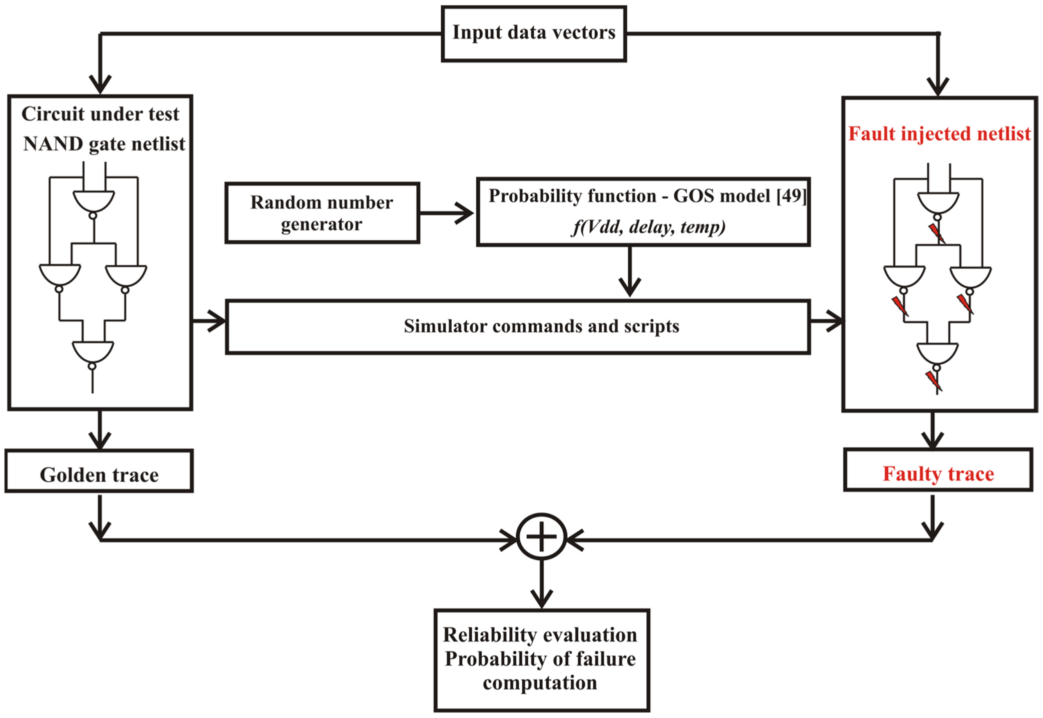


Fig. 4.4 SFI methodology based on simulator commands and scripts

i) The first approach is based on a fault injection module designed in Verilog HDL, which asserts a series of control signals in the design each time a fault is injected with the probability of failure considered in [49] for the Gate Output Switching (GOS) model. This module consists of random number generator modules and probability function modules and each of them activates some control signals associated with the desired failure moments. For each run, each input combination applied to the adder triggers the generation of new control signals, which are parsed by a script in order to decide when to modify the logic value of the output of one or more gates in the design. This method comes with an overhead for the Verilog code of the circuit under test, because the random number generation and the probability calculation are implemented using the hardware description language. A part of the TCL script code is presented below, where the time related parameters found in the "force" commands show the transient nature of the faults injected in the system under test.

```
project compileall
vsim -novopt work.RCA_6bits_tb
# verify if fault injection control signal is enabled
when {sfi_control_signal1 == 1}{
  # inject faults for FAC1 gates
  force -deposit signal1.1 2#0 {10 ns}
  force -deposit signal1.2 2#0 {10 ns}
  force -deposit signal1.3 2#0 {10 ns}
  ...
}
when {sfi_control_signal2 == 1}{
  # inject faults for FAC2 gates
  force -deposit signal2.1 2#0 {10 ns}
  force -deposit signal2.2 2#0 {10 ns}
  force -deposit signal2.3 2#0 {10 ns}
  ...
}
#inject faults for FAC3, FAC4, ..., FACn
...
# record the simulation start moment
set before_run [clock milliseconds]
run 160 us
# record the simulation stop moment
set after_run [clock milliseconds]
# compute the total simulation time
set total_run [expr $after_run - $before_run]
```

ii) The second approach has the advantage of eliminating the Verilog code overhead, by moving the computation necessary for random number generation and probability calculation from the hardware description language code to the TCL script code. A sequence of the TCL code developed for this approach is presented below. The *random_int* procedure generates a random number situated in the [1, upper_limit] interval. The probability to failure parameter is denoted by the acronym PTF. The presence of a new vector at the inputs of the adder, verified by the instruction *when {sim:/RCA_6bits_tb/adder1/x }*, triggers the beginning of a new step, which includes a new set of randomly generated numbers.

```
  #the procedure which generates a random number between 1 and
upper_limit
  proc random_int {upper_limit }{

      global myrand

      set myrand [expr int(rand() * $upper_limit + 1)]

      return $myrand

      }
```

```
#verify if input vector has changed
when {sim:/RCA_6bits_tb/adder1/x }
{   set nr1 [random_int 1000000]
    ...
    if {$nr1 %  $PTF == 1} {
      force -deposit  signal1 2#0 {10 ns}
      force -deposit signal2 2#0 {10 ns}
     ...
# inject faults for multiplexer gates
    force -deposit mux_signal1 2#0  {10 ns}
   ...  }
```

The actual simulation of the circuit under test and the result analysis step, which takes place immediately after each run, compose the second phase of the SFI process. A Verilog testbench module controls the actual simulation and the results analysis, by controlling the total number of runs, by selecting the input vector for each run and by comparing the faulty trace obtained at the outputs of the circuit with the golden trace associated to a fault-free run. The testbench module also implements the equations for calculating the reliability of the circuit under test.

The circuits under test were represented by two types of adders, implemented using only 2-input NAND gates in Verilog HDL: ripple carry adders (RCA) and carry select adders (CSeA).

In order to make a precise comparison between the two methodologies, the circuits under test have been used in the same configuration as the one described in [49], namely: 6-bit RCAs and 6-bit CSeAs. The block design and the implementation of the RCA used in the experiments is depicted in fig. 4.5. The block design of the CSeA is depicted in fig. 4.6.

Modelsim 10.05 SE commercial simulator was used in order to perform the simulations, on a desktop computer with Intel Core i5 at 3.2 GHz and 4 GB of main memory, running Windows 8.1 OS. 16000 test vectors have been applied at the inputs of each adder (16 input vectors for each of the 1000 runs), in order to maintain the compatibility with the simulations included in [49]. The methodology has been validated by computing two dependability parameters, the probability of failure associated with each bit of the result and the probability of failure of the entire output vector, and by comparing them with the ones in [49].

For the first simulation campaign, a 6-bit ripple carry adder (RCA) configuration with the same delay of 1.5 ns on each gate has been considered. Each gate of the design was supplied at the same voltage, with the following values: 0.25, 0.30 and 0.35 V respectively. The results of this simulation campaign are displayed in Table 4.6.

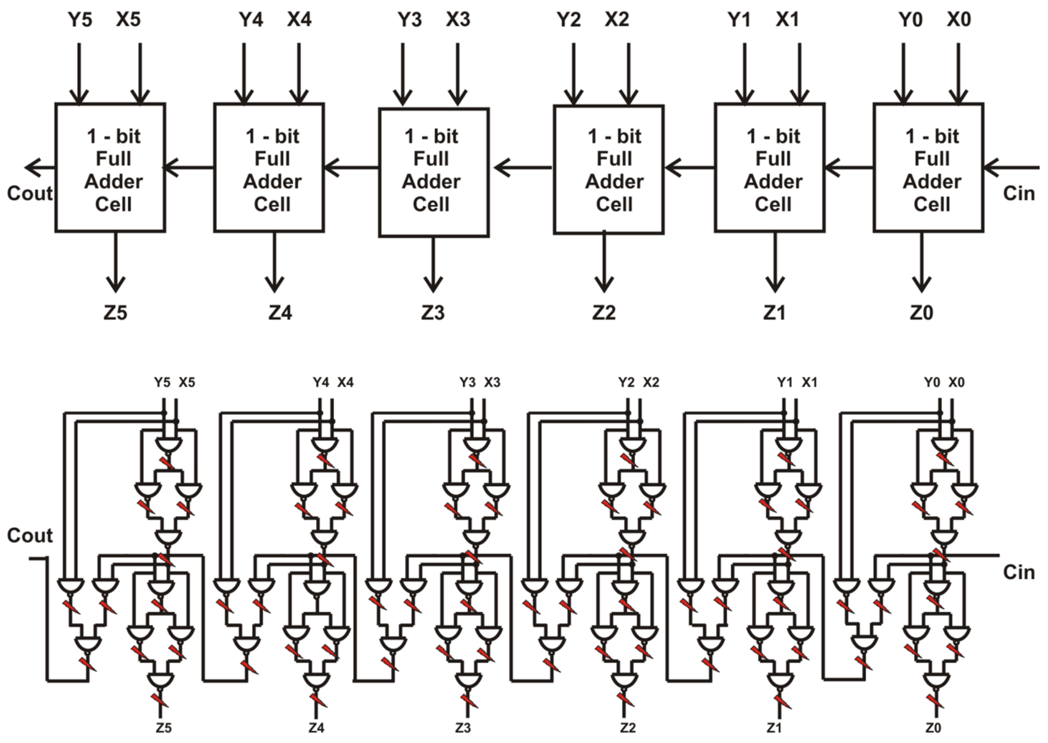


Fig. 4.5 - 6-bits RCA block scheme and implementation using only 2-input NAND gates

Both the bit-independent and the overall probabilities of failure obtained are only slightly higher than the ones in [49], for the Gate Output Switching (GOS) model applied to the 6-bit RCA. The small differences noticed for the values of the output probabilities can be explained by the fact that GOS model used in [49] injects faults at the output of the gate, with a given probability, only in the case the gate performed a switch. The number of faults injected by the new methodology in a certain time window is higher: despite the fact that it uses the same probabilities, the new model doesn’t take into account the switching activity.

The second simulation campaign targeted a 6-bit carry select adder (CSeA) configuration with only one delay value (1.5 ns) associated for each NAND gate of the design and its results are presented in Table 4.7. The simulation of the correct RCA and CSeA adders both require 0.5 s. According to tables 4.6 and 4.7, a simulation campaign of 1000 runs, based on simulator commands and scripts requires between 3 and 70 seconds, depending on the number of faults that must be injected in the design. For usual gate probabilities of failure, found in practice, the simulation overhead is 6x – 30x with respect to the gold circuit. The authors in [37] expect that the techniques based on simulator commands will provide the lowest temporal cost associated with the simulation. However, our measurements suggest that the temporal cost of this method is higher with respect to the one required by the mutant-based method, but it still represents an affordable value for simulations running on modern computers.

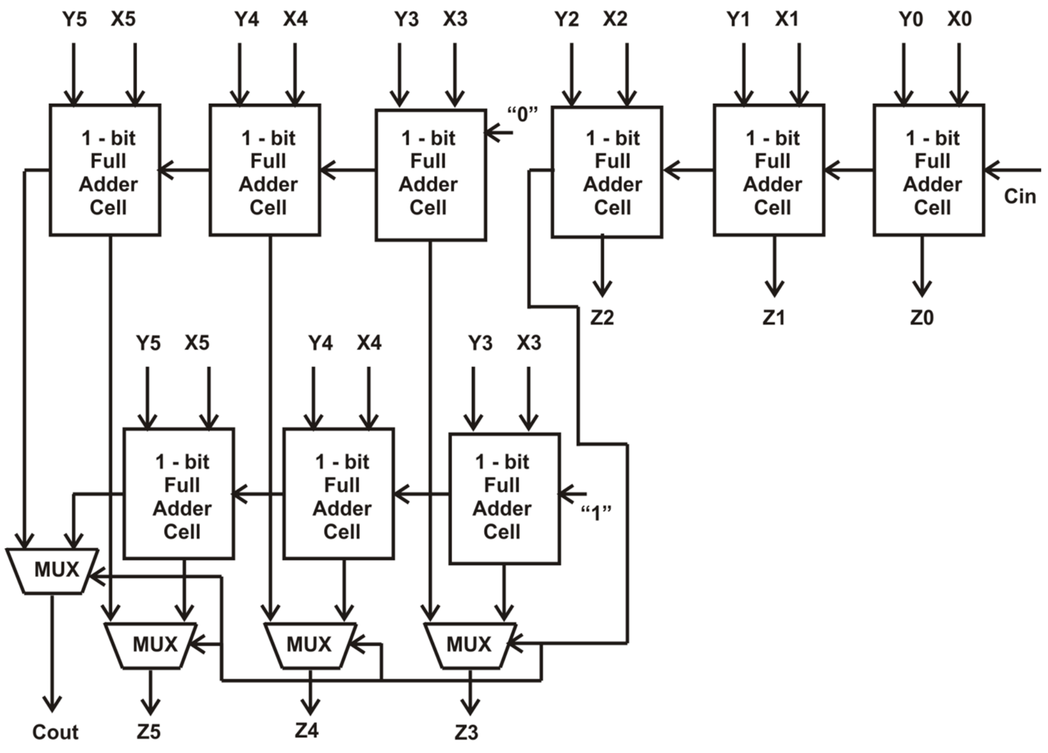


Fig. 4.6 6-bit CSeA block scheme

The alternative methodology for transient errors impact analysis at gate level, presented within this chapter, can be successfully applied for small and medium digital systems. In the case of large circuits, this approach becomes prohibitive in terms of simulation time and resources required. The methodology has been validated by confronting its results with a similar strategy, based on HDL code alteration of the circuit under test, which has been published in [49].

The described methodology stands out with respect to the existing techniques due to the following advantages: the easiness in the simulation set-up process and the high degree of flexibility. The technique has been implemented by two different procedures; the one based entirely on simulator commands has the additional advantage of generating no overhead for the Verilog code of the circuit under test. When applied for several types of adders, the results prove that the probabilities of failure of the circuits under test are tightly related to the ones reported in [49], while the simulation time remains within reasonable bounds for small and medium-size gate-level netlists. The work presented throughout section 4.3 has been published in paper [51].

Vdd [V]	Gate failure probability [%]	Result bits failure probabilities [%]							Circuit failure prob. [%]	Circuit failure prob. obtained in [49] [%]	Run-time [s]
		6	5	4	3	2	1	0			
0.35	0.2827	3.0187	5.3125	4.9875	4.3438	3.6938	1.6563	1.2375	11.5813	10.1625	3
0.30	1.9460	17.9750	29.7813	28.3562	23.1563	19.8813	10.6563	9.7188	62.0187	50.0875	15
0.25	10.1300	21.6375	14.1938	28.0375	49.1938	43.9937	37.0250	35.1625	90.7062	90.1625	70

Table 4.6 – Simulation results for 6-bit RCA, using simulator commands and scripts

Vdd [V]	Gate failure probability [%]	Result bits failure probabilities [%]							Circuit failure prob. [%]	Circuit failure prob. obtained in [49] [%]	Run-time [s]
		6	5	4	3	2	1	0			
0.35	0.2827	3.7437	6.8438	5.2687	4.0938	3.6938	1.6563	1.2375	12.2063	12.1687	3
0.30	1.9460	20.9875	38.1000	30.5125	22.0562	19.8813	10.6563	9.7188	65.0250	55.5938	14
0.25	10.1300	72.5750	81.0750	68.6000	43.7625	43.9937	37.0250	35.1625	93.3875	91.0875	63

Table 4.7 – Simulation results for 6-bit CSeA, using simulator commands and scripts

4.4 Concluding remarks and contributions for gate-level analysis

Throughout this chapter, a bottom-up approach has been used in the first part: SPICE Monte-Carlo simulations represented the starting point for deriving higher level error models, implemented using hardware description languages. The simulations of circuits affected by process-voltage-temperature (PVT) variations, performed by the authors in [48], have shown the delay-dependent probabilistic nature of faults. Reliability evaluation has been performed by employing mutant-based SFI for small and medium combinational circuits. The simulations have proven that the proposed SFI methodology exhibits a simulation time overhead between 2x and 5x, with respect to the time required by the simulation of the fault-free circuit.

The original contributions of the gate-level analysis described in section 4.2 are:

- the definition of 4 fault-models for basic logic gates, with different accuracies;
- data-dependency feature assured using different probabilities of failure for each input combination that determines the gate switching (the fourth fault model discussed);
- flexible mutant-based SFI architectures for gate-level description of sub-powered circuits, it can be applied for circuits with unbalanced delay paths, multiple voltage islands or asymmetrical heated-up regions;
- the flexibility of the proposed methodology for small and medium netlists has been shown by varying the selected parameters (voltage, temperature, delay) according to the topology of the circuit.

Section 4.3 of this chapter presented an alternative methodology for gate-level SFI, using simulator commands. The characteristics of this technique and the contributions are:

- easy to implement reliability technique based on simulator commands and scripts;
- the technique’s accuracy has been validated by confronting its results to the ones of the previously discussed methodology, based on code alteration;
- the simulation time required is reasonable if applied to small and medium complexity netlists; the overhead of this method is 6x – 30x with respect to the fault-free circuit simulation time;
- the technique has been implemented using two different approaches: one of them has the advantage that it brings no overhead to the Verilog code of the design under test.

5. PROBABILISTIC INTERCONNECTS

5.1 Saboteur-Based Logic Level Simulated Fault Injection for Sub-Powered Interconnects

This chapter aims to describe a methodology for transient faults impact analysis at gate / logic level in sequential circuits, using HDL saboteur-based simulated fault injection techniques. I have focused on reliability issues of signals transmitted on low supply voltage interconnects. This work has been published in 2014, in conference paper [59] and, also, in i-RISC project deliverable D2.2 [60].

Reliability issues in interconnects occur mainly due to two factors: process variation and crosstalk induced faults. Process variations may be caused by device geometry variations, device material, electrical parameter variation, interconnect geometry and material parameter variations [61]. These variations will affect the metal thickness or length, dielectric thickness, contact and via size, metal resistivity or dielectric constant. Therefore, the resistance, capacitance or inductance parameters of a wire will be altered. Process variation in interconnects may modify the timing characteristics of the signals. An erroneous result at the moment when a certain signal is sampled may appear due to increased resistance or ground capacitance of the wire.

The main target of this analysis is to perform reliability evaluation for different groups of signals contained by the interconnect, using a saboteur-based approach. According to the classifications found in literature, there are several types of saboteurs: serial simple unidirectional saboteur, serial simple bidirectional, serial complex saboteur, serial complex bidirectional saboteur, *n*-bit unidirectional serial saboteur, *n*-bit bidirectional serial saboteur, parallel saboteur. Our methodology is based on n-bit unidirectional saboteurs.

In order to analyze the behavior of interconnects in the presence of probabilistic errors, we have proposed several types of saboteurs, with different accuracies. The most simple one performs a probabilistic bit-flip on a single signal of the interconnect, while the most accurate one takes into account that the probability of error occurrence on one wire depends on the values transmitted on the entire interconnect. The four types of saboteurs proposed for our analysis are:

1. *Standard Signal Probabilistic* (SSP) saboteur. This type of saboteur performs a simple bit-flip of the logic value of the signal on which it is applied, with a considered stand-alone probability. This model doesn't account for the last type of transition that took place on that line, nor the data pattern. The architecture of this saboteur includes a fault insertion module, which is triggered according to the considered probability of failure and to a randomly generated number.
2. *Switching-Aware Probabilistic* (SAP) saboteur. For this model, we have considered a probabilistic behavior of the signal, related only to the moment when a transition took place. There are two versions of this type of saboteur: the first one has a simplistic behavior and considers the same probability for both types of switching, while the second one is more complex and considers distinct probabilities for charging and discharging processes. In order to accomplish the desired behavior, the architecture of this saboteur includes a switching detector.

3. *Full Data Dependent* (FDD) saboteur. This model considers that the probabilities for a line are expressed as a function of the data pattern transmitted on the entire bus. Taking into consideration that the crosstalk effect is strongly data dependent, this case models very accurate the occurrence of crosstalk induced faults. The high accuracy of this model comes with an important drawback: diminished scalability. This statement is justified by the high number of probability values that must be derived for each line, namely 2^{2n} probabilities, where n represents the bus width.

4. *Partial Data Dependent* (PDD) saboteur. This model represents a simplification of the previous one, because we have considered that the probability of failure for a line depends only on the data pattern transmitted on the neighboring lines (1-wire vicinity or 2-wire vicinity). The 1-wire vicinity situation models the case of buses affected by the capacitance effect of crosstalk, which manifests only on the adjacent line.

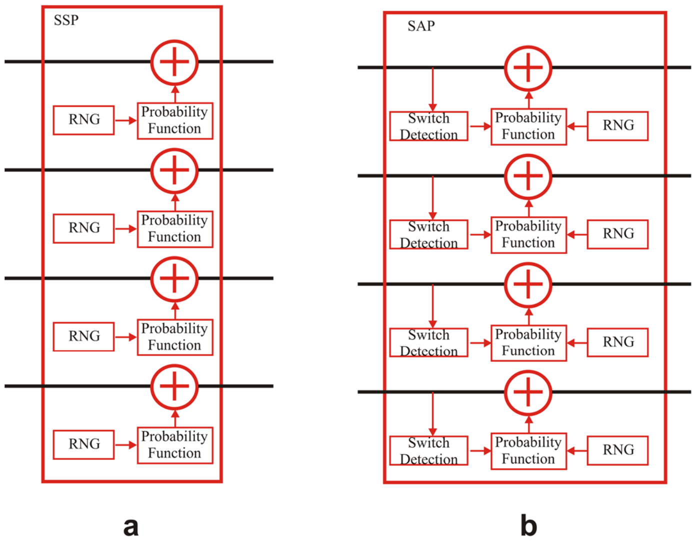


Fig. 5.1 The saboteurs' architectures according to fault models 1 and 2 (a - SSP, b - SAP) [59]

Figures 5.1 and 5.2 depict the architectures corresponding to all four types of saboteurs. All saboteurs consist of a random number generator, which is used to compute the probability of error occurrence. The SAP incorporates a switch detection module, while the PDD and FDD monitor the data on the lines.

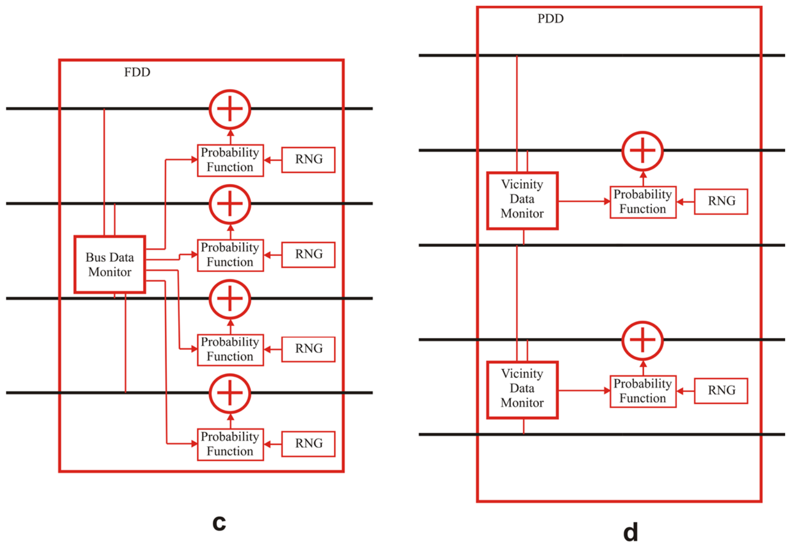


Fig. 5.2 The saboteurs' architectures according to fault models 3 and 4 (c -FDD, d -PDD) [59]

The circuit under test chosen for this simulation has been the open-source Wishbone bus, designed in Verilog HDL and available on the OpenCores website [75]. We have simulated conventional read and write cycles on a particular system consisting in 2 master units and 5 slave units, with 32-bit data and address buses. The simulations have been carried out using Modelsim 10.3 commercial HDL simulator on desktop computer with Intel Core 2 Duo at 2.4 GHz and 2 GB of main memory, with Windows XP OS. Each simulation campaign consisted of 1000 runs, with data sets chosen randomly for each run. We have created several groups of signals, which were the subject of fault injection campaigns:

- Data write signals (the 32-bit unidirectional data bus from master to slave)
- Data read signals (the 32-bit unidirectional data bus from slave to master)
- Address signals – a distinction between the first 4 address bits (the ones used to select the slave) and the rest of the address bits (which are used to address within the slave)
- Master control and handshaking signals (*we, cyc, stb* and *sel*)
- Slave handshaking signals (*ack, rty, err*)

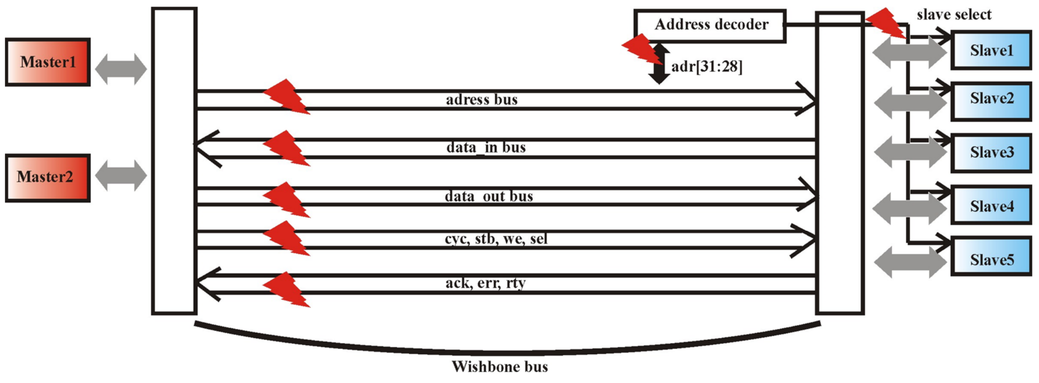


Fig. 5.3 The wishbone's signal groups that are the subject of fault injection campaigns, [59]

Concerning the reliability parameters of the Wishbone bus, the following concluding remarks can be drawn:

- Faults which occur on the most significant signals of the address bus will determine an erroneous slave selection, leading to a dramatic effect on the overall reliability. Therefore, the most significant bits of the address bus have a critical role in the reliability of the Wishbone system.
- Faults affecting master to slave control and handshaking signals (*cyc, stb* and *we*) have the following consequences: (a) wrong type of transaction, meaning that a read transaction may be performed instead of a write one or vice-versa, (b) the bus performs no transaction because the bus arbiter cannot grant the bus to the master which had asserted the *cyc* signal, (c) prematurely terminated transactions, due to errors occurring on *cyc* and *stb* signals during the time they are enabled (these signals must be activated throughout an entire transaction).
- Faults affecting the slave to master handshaking have the following consequences: (a) the ongoing transaction may remain frozen because the master hasn't received any of the *ack, rty* or *err* signals, so it doesn't disable the *cyc* signal; (b) a transaction may be completed earlier than normal, because the master receives a wrong *ack, err* or *rty*; (c) longer transaction than normal when errors occur on the *rty* signal, because the master reinitiates the transaction when receiving the *rty* signal.

Faults that affect *sel* lines and data signals have a less significant impact on the overall reliability, because they affect only the data transmitted on the bus, so they don't interfere with the transaction timing or flow.

The fault model type and the victim signal / signals chosen for each fault injection campaign, along with the injected probability of failure and the exact simulation time required by each campaign, are all depicted in table 5.1. A simulation set consisting in 1000 runs requires less than 2 seconds, while the gold circuit simulation requires about 1 second. Our measurements indicate that, generally, the simulation time for a SFI campaign is 1.7x higher with respect to the correct circuit, which represents a reasonable simulation overhead for a system of such complexity.

Fault model type	**Victim signal**	**Probability of failure**	**Runtime [ms]**
SSP during WRITE cycle	sel	3%	1828
	sel and data	3%	1765
	adr[31:28]	3%	1812
	adr[31:28]	5%	1750
	adr[31:28]	10%	1750
	cyc, stb, we, sel	3%	1750
SSP during READ cycle	ack, err, rty	3%	1703
SAP during WRITE cycle	adr[31:28]	5% for 0->1 3% for 1->0	1766
	adr[31:28]	10% for 0->1 5% for 1->0	1766
	cyc, stb, we, sel	5% for 0->1 3% for 1->0	1750
	cyc, stb, we, sel	10% for 0->1 5% for 1->0	1781
	data	5% for 0->1 3% for 1->0	1766
	data	10% for 0->1 5% for 1->0	1782
SAP during READ cycle	ack, err, rty	5% for 0->1 3% for 1->0	1703
	ack, err, rty	10% for 0->1 5% for 1->0	1703
PDD during WRITE cycle (1-wire vicinity)	adr[31:28]	[3% ÷ 20%], depending on the transition pattern	1797
	adr[27:0]		1797
	slave select signals		1766
	cyc, stb, we, sel		1766
	ack, err, rty		1765
	data		1782
PDD during READ cycle (1-wire vicinity)	ack, err, rty	[3% ÷ 20%], depending on the transition pattern	1782
	data		1906
Gold circuit – WRITE cycle	NO fault injection	0%	1078
Gold circuit – READ cycle	NO fault injection	0%	1046

Table 5.1 - The results of the simulation campaigns performed for reliability assessment of Wishbone bus

5.2 Conclusions and contributions for interconnects

The SFI based reliability evaluation technique for probabilistic interconnects, described throughout chapter 5, stands out with the following features:

- four types of saboteurs have been defined: the simplistic probabilistic type, the switching-aware probabilistic saboteur, the partial data-dependent and the full data-dependent type;
- the crosstalk noise affecting the interconnects is data dependent, so a high accuracy analysis is required, which is implemented as follows: the partial data-dependent saboteur takes into account the influence of transitions occurring on the lines situated in a vicinity of the analyzed wire, while the full data-dependent takes into account the transitions that occur on all wires of the interconnect;
- probabilistic faults have been injected on address, control and data signals of the Wishbone bus and the simulations have indicated the most critical signals in the overall reliability;
- the simulation overhead for a SFI campaign is 1.7x higher with respect to the fault-free circuit.

6. SIMULATED FAULT INJECTION FOR RELIABILITY ANALYSIS OF REGISTER TRANSFER LEVEL CIRCUIT DESCRIPTIONS

6.1 General Considerations about RTL Reliability Assessment Based on Simulated Fault Injection

With increased device integration, increased affinity for using low supply voltages and a gradual trend towards higher operating frequencies, the effects of transient errors can no longer be ignored. We have shown that electrical, logical and temporal masking represent the key factors that prevent the majority of single event transients from becoming functional failures. The previous chapters of my PhD thesis presented several gate-level reliability assessment methodologies, tailored for analyzing the effects of transient errors and probabilistic faults. However, during an industrial design cycle of VLSI circuits (fig. 6.1), by the time a gate-level netlist is available, it is too late and too costly to make design changes. This is the reason why the development of efficient reliability evaluation methodologies at higher levels of abstraction of a digital system, hence earlier during the design cycle, is imperative.

In this chapter, the work carried on at the third level of a digital system abstraction is presented. As described in the proposed research plan, the third level was considered to be the Register Transfer Level. This work consists mainly in a data dependent reliability assessment methodology for digital systems described at RTL, using a hierarchical approach. In order to analyze the impact caused by transient errors in a digital system based on low-power components, at all levels of abstraction, the characteristics and the results obtained at lower levels must be used in a hierarchical manner to tackle the upper layers methodologies and issues and to derive the numeric results.

Various gate level (GL) and register transfer level (RTL) reliability evaluation techniques are presented in the literature. Performing simulated fault injection for the GL description of a system could provide highly detailed and accurate information, but it may become computationally prohibitive due to the large number of instances that must be handled simultaneously by the simulation environment. Also, fault simulation efforts carried out in the post logic-synthesis phase are too late in the design cycle to be useful for design-for-test related improvements in the architecture [62]. Working at higher levels of abstraction, especially at RTL, is preferred among designers because the simulation time is orders of magnitude lower, rapid prototyping of function is allowed, the fault injection methodology is easier to follow and it offers portability to similar designs [63].

The RTL based SFI has been used either for testing purpose [62], in order to derive the fault coverage in early design phases of specific test vectors, either for reliability assessment purposes [37][63][64]. The authors in [62] use stratified fault sampling in order to estimate the gate-level fault coverage of given test patterns.

The RTL coverage of a module is experimentally found to follow the gate-level coverage within the statistical limits.

The authors in [62] discuss the problem of fault coverage in the context of fault injection techniques. The paper emphasizes the idea that none of the RTL fault injection techniques described in the literature establishes the relationship between high level fault coverage and gate level fault coverage. To support their statement, they discuss about one solution from the literature that determines only empirically the difference between the RTL and the gate-level fault coverage, but doesn't bring any theoretical foundation. They also discuss other solutions which are applied to RTL circuit descriptions in order to produce the behavior of all possible gate-level single "stuck-at" faults, but those are not accurate enough. The solution proposed by the authors in [62] assumes that not all gate-level faults are represented at RTL, since RTL represents a higher level of abstraction and may not provide the low-level structural information needed to exactly replicate all gate-level failures. The authors consider the efforts to model all gate-level faults at RTL being inefficient, because the gate-level netlist changes with every logic synthesis iteration.

Regarding the SFI components for RTL, we can state that two types of approaches have been proposed. One approach is based on altering the signals within the RTL design [62]. For this scenario, the modification of behavioral statements is not considered. The other approach is based on altering the behavioral components of the RTL descriptions [64][65]. These include: replacing the values of conditions in *if* and *case* statements (*stuck-then, stuck-else, dead process, dead clause*), disturbing assignment statements (*assignment control, global stuck-data*), or disturbing operators in expressions (*micro-operation, local stuck-data*), etc. They can model in an accurate way simple faults, such as stuck-at faults. Nonetheless, these solutions don't have the capability to accurately model transient errors or probabilistic faults.

The fault injection experiments conducted by the authors in [65] use the tool called VFIT (VHDL-based Fault Injection Tool) to inject intermittent faults in the memory elements and buses of a microcontroller. The fault models considered by the authors are: intermittent stuck-at, intermittent pulse, intermittent short and delay and the fault injection campaigns are based on simulator commands. These experiments have the disadvantage that they don't take into account the data dependency, so they cannot be applied for different transitions that exhibit different probabilities of failure. The authors claim that the tool has limitations for injecting complex fault models, so the methodology is not accurate enough for studying the errors induced by sub-powered CMOS circuits.

A multi-level hierarchical single event transients (SET) analysis approach that can be applied at RTL is proposed in [66]. The authors compute the SET sensitivity of a complex circuit by decomposing it into blocks and combining the compact SET models. The methodology uses GL simulations for building blocks in order to derive appropriate fault models for the RTL simulation. However, this approach does not take into account the input stimuli. Although the SEU fault model does not consider data dependency, the input stimuli represent an important parameter in SFI campaigns, as errors may be masked by different input combinations.

The SFI methodologies presented in the literature are mainly built with two goals in mind: the fault modeling capability and the simulation overhead. Good fault modeling capability is usually obtained when using low level circuit descriptions, while simulating complex systems at low layers of abstraction is generally prohibitive due to the simulation time required. Several approaches described in the literature [66][67][68][69] have tried to find a trade-off between the fault modeling capability and the simulation overhead. Fault tolerance analysis is performed on multiple layers of abstraction: usually fault models and fault behavior corresponding to higher abstraction layers are derived using low-level descriptions of circuits. Additionally, the reliability of the entire system is derived using high level fault tolerance analysis. Papers [66], [67] and [69] describe methodologies to evaluate the reliability of digital systems described at RTL under Single Event Transients (SET) fault models. Static timing analysis for combinational blocks is used in [67] in order to reduce the set of faults and to find the blocks which may produce errors at blocks' primary outputs. Besides, SET fault injection for gate level characterization is used in [69].

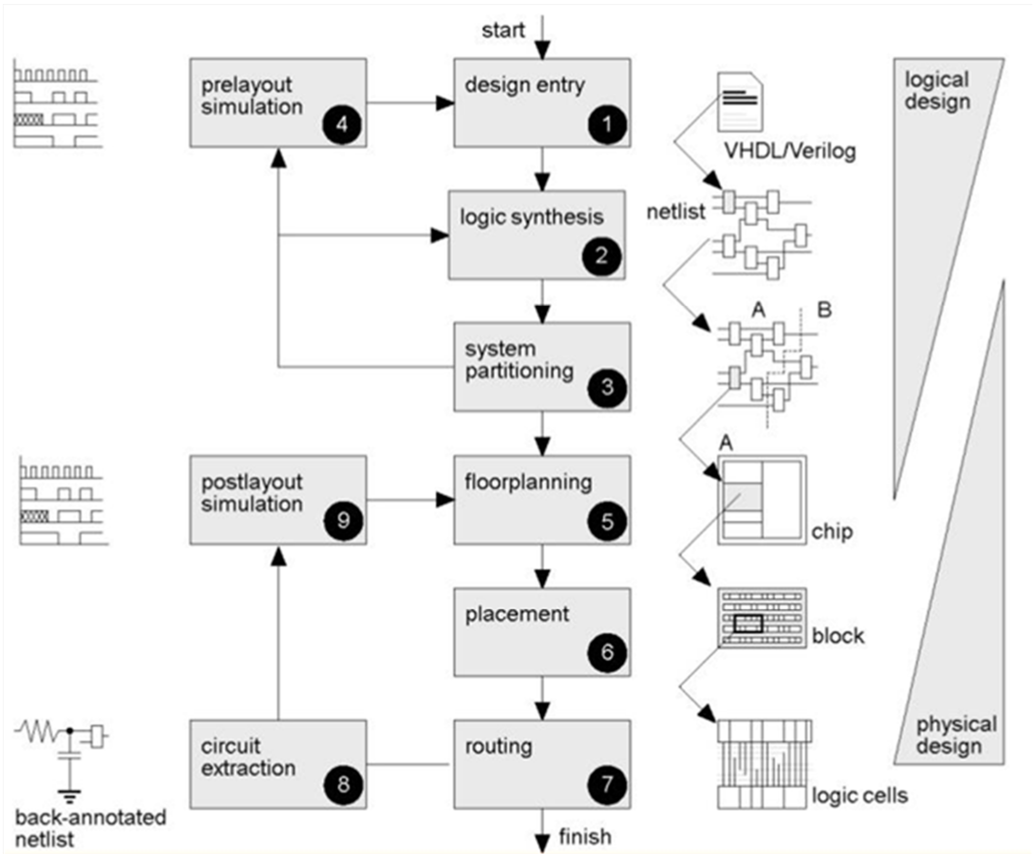


Fig. 6.1 Typical VLSI Design Flow, [70]

6.2 A Novel Hierarchical Hybrid Methodology for Reliability Assessment of RTL Circuit Descriptions

The methodology proposed in this chapter for RTL reliability assessment combines the Gate Level (GL) data dependent Simulated Fault Injection (SFI) for reliability metric extraction of building blocks and the RTL simulation. This way, we aim at approaching the accuracy of the GL SFI, while maintaining the low simulation overhead specific to RTL based evaluation. Performing SFI at higher levels of abstraction, such as RTL, requires orders of magnitude lower simulation time with respect to GL SFI. The GL simulations performed for smaller blocks have the target to capture accurately the data dependency, which is indispensable for the analysis of the probabilistic behavior of sub-threshold or near-threshold CMOS devices. The inputs for GL simulations have been extracted using the RTL fault-free simulations. The error probabilities obtained after the GL SFI phase are used later in the development of RTL saboteurs. The work presented in this section has been published in i-RISC project deliverable D2.2 [60] and, also, in journal paper [76].

The methodology has the following phases: (i) correct simulation for a specific set of inputs of the RTL description, which aims at capturing the inputs for each of the components, (ii) hierarchical block decomposition, which splits the RTL designs in simple building blocks, (iii) logic synthesis of the components obtained after the previous step, (iv) data dependent SFI of the GL netlists and (v) the RTL SFI using the probabilities derived in the previous step. We have validated our methodology for a 128-bit Advanced Encryption Standard (AES) crypto-core, for which the GL simulation could not be performed on the target machine.

The multi-level simulation-based fault injection reliability evaluation methodology is depicted in fig. 6.2. The hierarchical block decomposition phase is intended to partition the high-complexity RTL system into blocks of low complexity, which are suitable for straight-forward GL simulation. The obtained low-complexity components are classified in two categories, either fully combinational, either sequential components.

The second phase of the proposed methodology consists in the RTL correct simulation of the entire digital system, with a given set of input stimuli. As a result of this simulation campaign, the inputs and the corresponding correct outputs of each block are extracted, in order to be used for the GL analysis and for the saboteur development step.

In order to obtain the GL netlist for each of the previously partitioned modules, the logic synthesis step is performed. The mutant-based simulated fault injection methodology described in the first PhD deliverable and in paper [49] is then used for determining the reliability parameters. Therefore, each combinational or sequential element of the resulted netlists will be affected by probabilistic faults, according to one of the models derived during previous work: GOP, GOS, GOST or GISP. The probabilities of failure of each output signal of the considered blocks are calculated, by confronting the correct outputs obtained during phase 2 with the results of the GL SFI step.

The last phase of the methodology consists in developing RTL probabilistic saboteurs according to the probabilistic values obtained during the previous phase. RTL SFI of the entire digital system is performed using these saboteurs and the overall probability of failure of the system is calculated by confronting the outputs obtained during the saboteur based SFI with the correct outputs derived during phase 2.

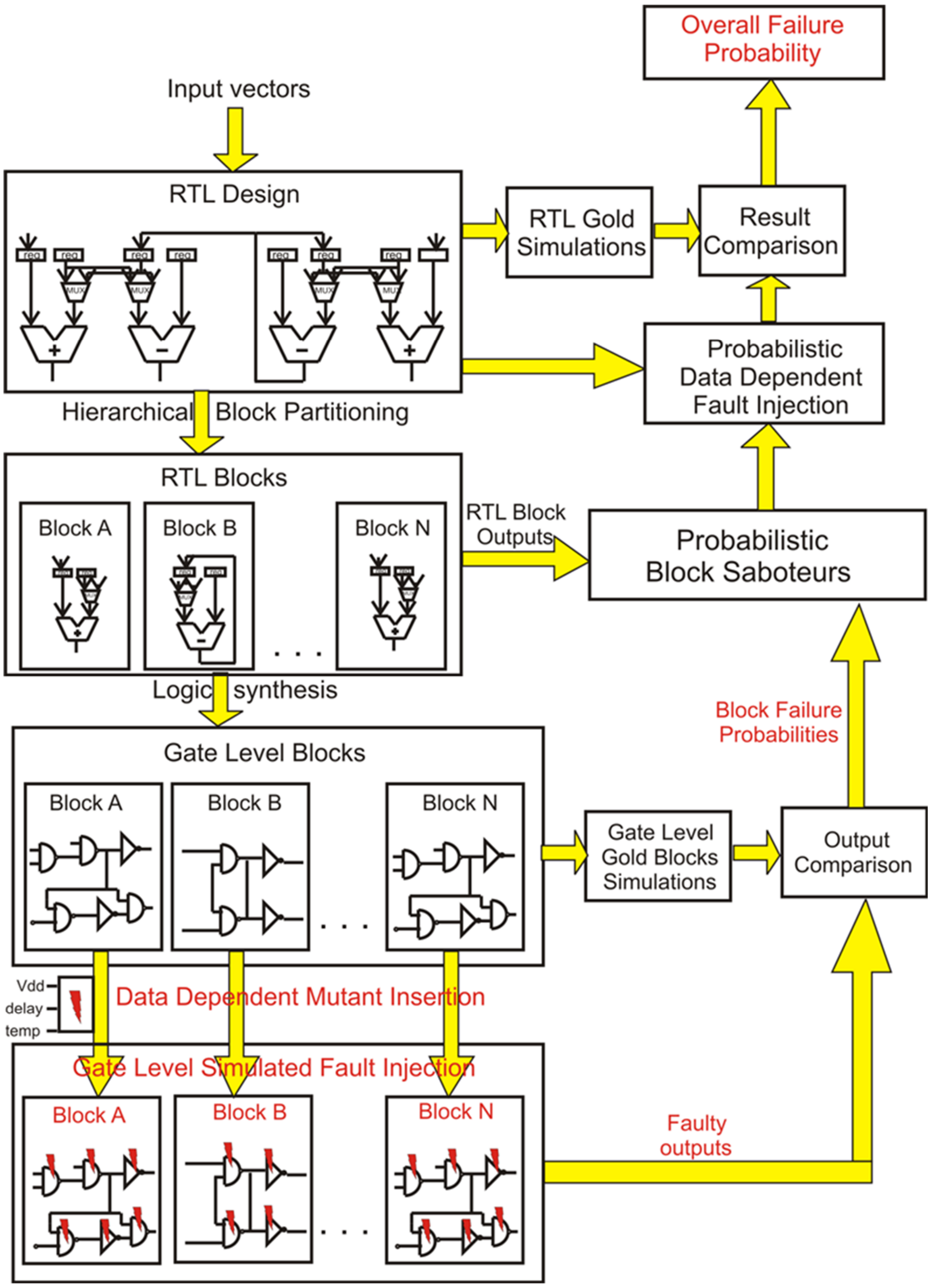

Overall Failure Probability
Input vectors
RTL Design
RTL Gold Simulations
Result Comparison
Probabilistic Data Dependent Fault Injection
Hierarchical Block Partitioning
RTL Blocks
Block A
Block B
Block N
RTL Block Outputs
Probabilistic Block Saboteurs
Logic synthesis
Block Failure Probabilities
Gate Level Blocks
Gate Level Gold Blocks Simulations
Output Comparison
Vdd
delay
temp
Data Dependent Mutant Insertion
Gate Level Simulated Fault Injection
Faulty outputs

Fig. 6.2 Multi-level simulation-based fault injection reliability evaluation methodology, [76]

6.3 Case Study for Validating the Accuracy of the Proposed RTL Fault Injection Methodology: the Reliability Assessment of a Medium Size Circuit

In order to establish some correlations between the previously proposed hybrid reliability assessment methodology and the classical GL SFI methodology, we have chosen a medium size circuit, for which the entire GL simulation is achievable on the selected target machine. The circuit is represented by a parallel comparator, which is a basic component of the check-node unit (CNU) processing modules found within the Low-Density Parity-Check (LDPC) decoders. We have chosen this circuit because LDPC decoders will be used in the final part of the PhD Thesis, for different fault injection experiments. Several LDPC decoder architectures will be analyzed in terms of error correction capability, throughput and required area resources in the FPGA fabric. LDPC decoders are also used in the final part of the FP7 i-Risc project, for the proof of concept.

A bi-partite graph called Tanner graph can conveniently represent an LDPC code. The Tanner graph contains two types of nodes: variable nodes, corresponding to the columns of a sparse parity check matrix denoted by H and the check nodes, corresponding to the rows of the same matrix. The decoding process is performed using message passing algorithms, which consists in the continuous exchange of messages between variable node units (VNUs) and check node units (CNUs).

The circuit of interest for this analysis is the parallel comparator of the CNU processing units. It is composed of two main blocks: (i) the sort module, which is used for arranging two pairs of inputs in an ascending manner and (ii) the compare-select module, which receives a set of four values as inputs and finds the first two minimums among them. The structure of such a parallel comparator used in the CNU processing units of the LDPC decoders is graphically represented in fig. 6.3.

The simulation results of the parallel comparator are presented in table 6.1, for both the hybrid RTL proposed methodology and the GL SFI. Considering the accuracy of the new methodology, we can conclude that the obtained results are similar with the ones provided by the GL SFI. Regarding the simulation time, our approach requires three orders of magnitude less time than the GL SFI approach.

Module	Probability of failure	Simulation time	Simulation type
Sort	0.0000%	0.10 ms / run	RTL gold simulation
Sort	0.0000%	0.11 ms / run	Gate level gold simulation
Sort	1.7116%	0.19 ms / run	Gate level FI simulation
Compare Select	0.0000%	0.18 ms / run	RTL gold simulation
Compare Select	0.0000%	0.27 ms / run	Gate level gold simulation
Compare Select	4.8260%	0.62 ms / run	Gate level FI simulation
Comparator	0.0000%	11.7 ms / run	Gate level gold simulation
Comparator	9.3333%	681 ms / run	Gate level FI simulation
Comparator	9.6667%	0.65 ms / run	Gate level + RTL FI simulation

Table 6.1 – Simulation results for parallel comparator

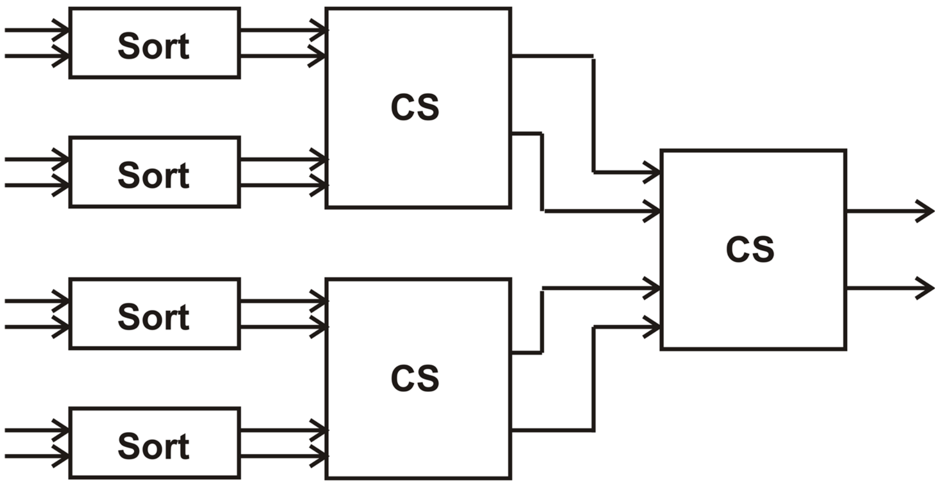


Fig. 6.3 Parallel comparator used in CNU modules of LDPC decoders (CS denotes the compare and select block)

6.4 Case Study for Applying the Proposed RTL Fault Injection Methodology: the Reliability Assessment of an AES Crypto-Core

Several hardware implementations of cryptographic algorithms are discussed in the literature. Some of them are suitable for field programmable gate array (FPGA) use, while others are tailored for application-specific integrated circuits (ASICs). The ASIC implementations offer a very low degree of flexibility when it comes to parameters or algorithm modifications, so the FPGA implementations are preferred. On the other hand, a software implementation of a cryptographic algorithm has the advantages of portability, flexibility, easiness in utilization and upgrade, but it fails in offering physical security, especially with respect to key storage. Many authors, including the ones in [73], consider that cryptographic algorithms implemented in hardware are more physically secure because they cannot be read or modified easily by an outside attacker.

The Rijndael algorithm was adopted in 2001 by the National Institute of Standards and Technologies as the Advanced Encryption Standard (AES) and replaced the old Data Encryption Standard (DES), which has been in use since 1976. The AES algorithm represents a symmetric block cipher which can encrypt and decrypt information using key sizes of 128, 192 or 256 bits [74]. Side channels attacks based on reliability evaluation under different conditions are common, so the analysis of cryptographic circuits is of great importance. Therefore, the proposed methodology has been applied for a 128-bit AES crypto-core, which has the Verilog source code available on OpenCores platform [75].

The plaintext of the AES algorithm is represented by the initial 128-bit state, which is modified by the round transformation and becomes the final state, which represents the output ciphertext. The state is organized as a 4*4 matrix of bytes and the round transformation scrambles these bytes either individually, row-wise or column-wise by applying the functions *SubBytes, ShiftRows, MixColumns* and *AddRoundKey* sequentially. The function *SubBytes* is the only non-linear function in AES, which substitutes all bytes of the state using table lookup, which is often called S-box. The *ShiftRows* function rotates the rows of the state by an offset, which equals the row index. The *MixColumns* function accesses the state column-wise and interprets a column as a polynomial over GF(2^8). The *AddRoundKey* function adds a round key to the state, a new round key being derived in every iteration from the previous round key [74].

The complexity of the considered crypto-core can be estimated by analyzing the synthesis results provided by Xilinx ISE 14.4 software for the Xilinx Spartan-6 LX45T FPGA device:

- 5792 out of 54576 slice registers (10% of the total capacity),
- 10992 out of 27288 slice LUTs (40% of the total capacity),
- 29 out of 166 block RAMs (17% of the total capacity).

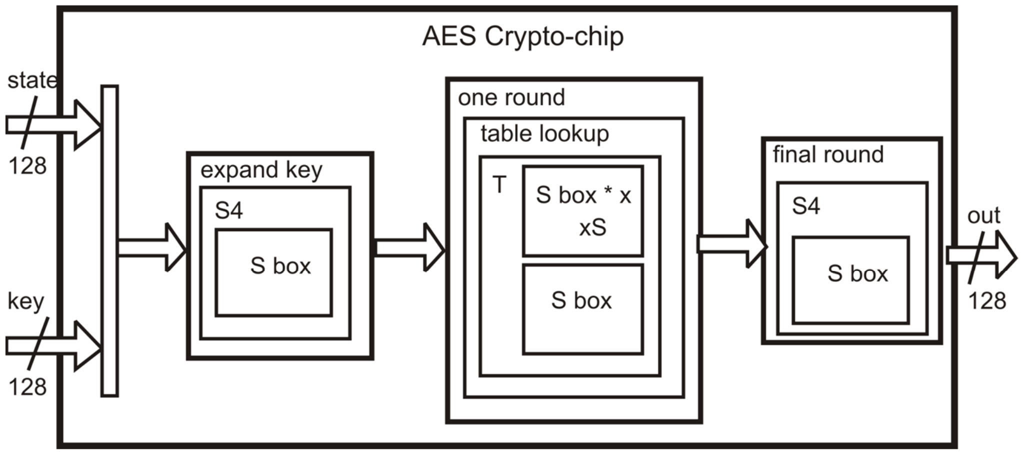


Fig. 6.4 AES crypto-chip, [76]

According to the first phase of the methodology presented above, the AES circuit has been partitioned in small complexity blocks, resulting 9 functional blocks and each of them has been further divided in the corresponding combinational and sequential sub-part. The description of each of the 9 blocks is depicted below:

- block A – the AES crypto-chip top module, which receives the 128-bit key and the 128-bit state as inputs and performs an exclusive-or on the two vectors; the top module contains one combinational sub-block, one sequential sub-block and instantiates block B 10 times, block C 9 times and block D one time;

- block B – referred as “expand key” in fig. 6.4 performs the expansion operation on the 128-bit key and contains one instance of S4 - block E; block B has been further divided in 7 combinational sub-blocks and 2 sequential sub-blocks; the structure of block B is depicted in fig. 6.6; as the majority of AES functional blocks, block B performs several XOR operations between the module’s input and the output of the block E instance; the AES crypto-core is composed of a total of 10 blocks B; after applying the logic synthesis phase for block B, a number of 464 NAND gates and 128 D-type flip-flops result;

- block C – referred as “one round” in fig. 6.4, performs XOR operations on the key bytes and instantiates block G 4 times; the structure of block C is depicted in fig. 6.5 c) and its components are further detailed in fig. 6.5 b); block C has been further divided in 6 combinational sub-blocks and 1 sequential sub-block; the AES crypto-core is composed of a total of 9 blocks C; after applying the logic synthesis phase for block C, a number of 512 EX-OR gates and 128 D-type flip-flops result;

- block D – referred as “final round” in fig. 6.4, instantiates 4 times block E and its structure is depicted in fig. 6.7; the AES crypto-core contains only one block D instance; block D has been further divided in 6 combinational sub-blocks and 1 sequential sub-block; the logic synthesis phase generated a netlist composed of 128 EX-OR gates and 128 D-type flip-flops;

- block E – referred as “S4” in fig. 6.4, substitutes four bytes in a word by calling 4 times the block F module; block E has no additional logic apart from the 4 instances specified before; there are 14 instances of block E in total, in the AES crypto-core;

- block F – referred as “S” or “S-box” in fig. 6.4, performs a table lookup operation and is being instantiated 184 times in the AES crypto-core; it doesn’t instantiate any other block; the netlist generated during the logic synthesis phase contains 335 2-input NOR gates, 526 2-input NAND gates, 65 inverters and 8 D-type flip-flops;

- block G – is part of the structure of block C and is referred as “table lookup” in fig. 6.4; it contains only one simple combinational sub-block; it uses the results provided by block H instances and it mixes them in order to derive its outputs; the AES crypto-core contains 36 instances of block G;

- block H – referred as “T” transformation in fig. 6.4, uses one instance of S – block F and one instance of xS – block I; in the structure depicted in fig. 6.5 a), we observe that it performs an XOR operation between the entire output of block I and a part of the output of block F in order to provide the output; the associated netlist contains only 8 EX-OR gates;

- block I - referred as “xS” in fig. 6.4, is similar to block F, performs table lookup operations; the associated netlist contains 364 2-input NOR gates, 533 2-input NAND gates, 67 inverters and 8 D-type flip-flops;

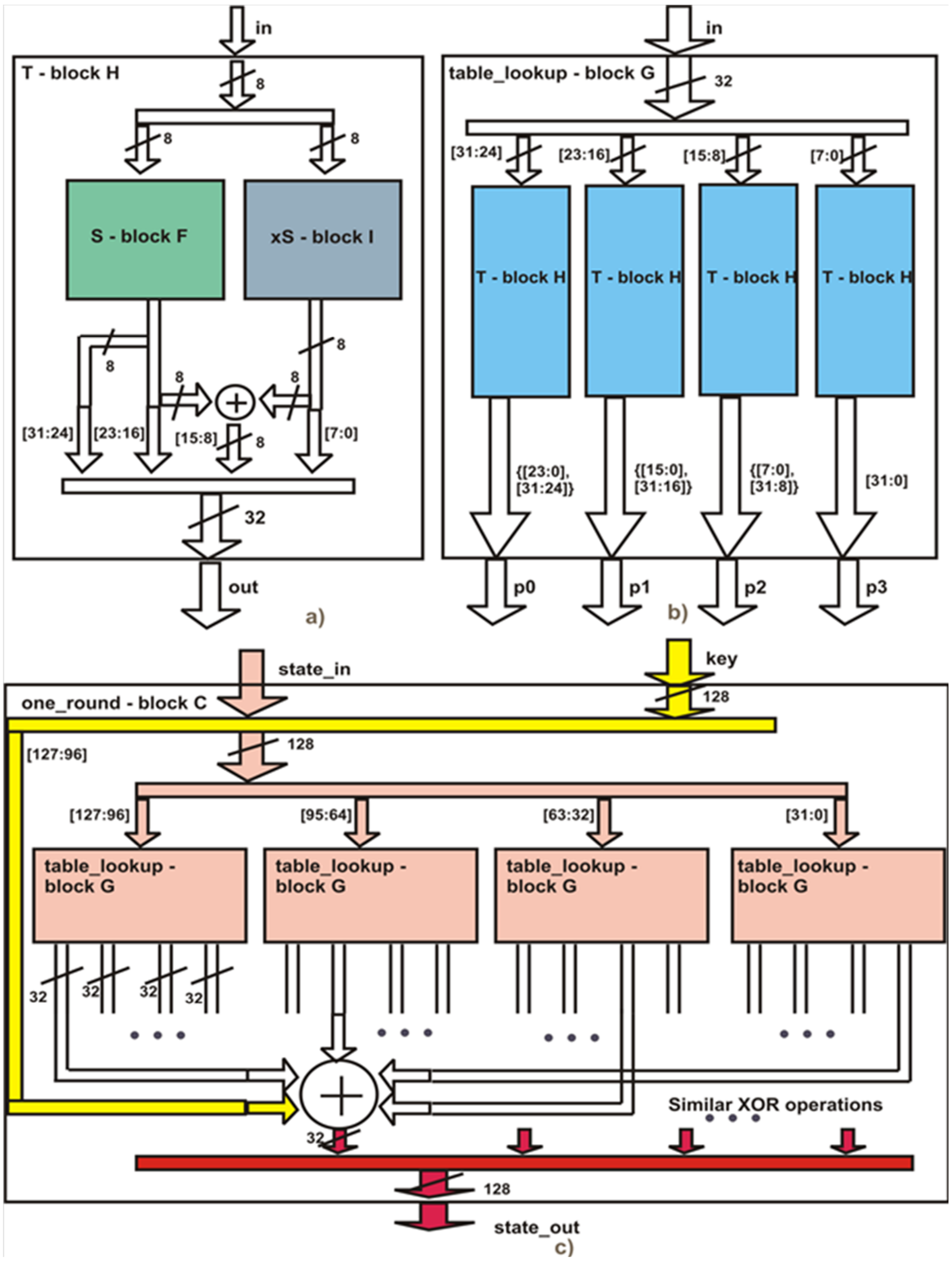


Fig. 6.5 AES crypto-chip block-level design - a) block H; b) block G; c) block C

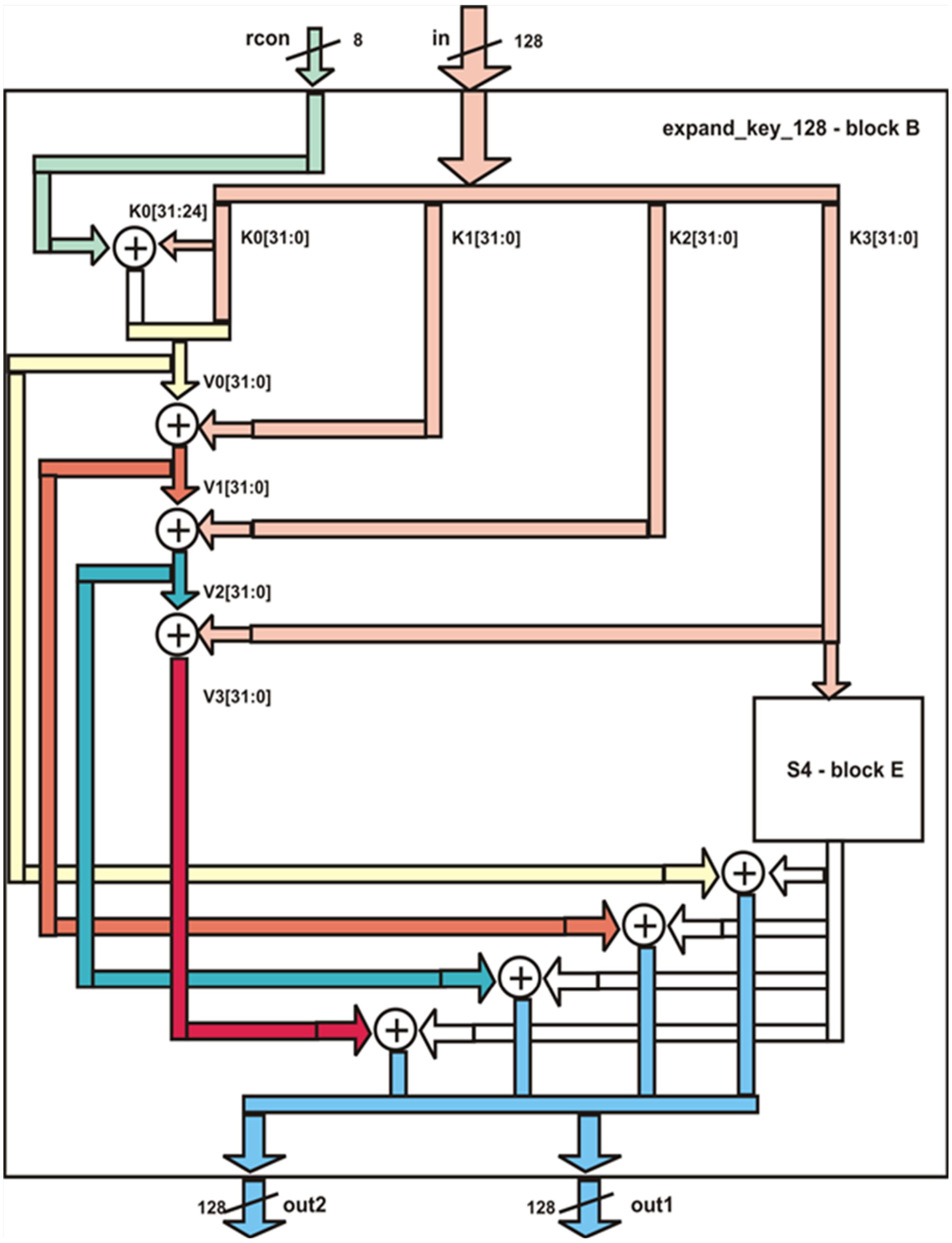


Fig. 6.6 AES crypto-chip block level design - block B

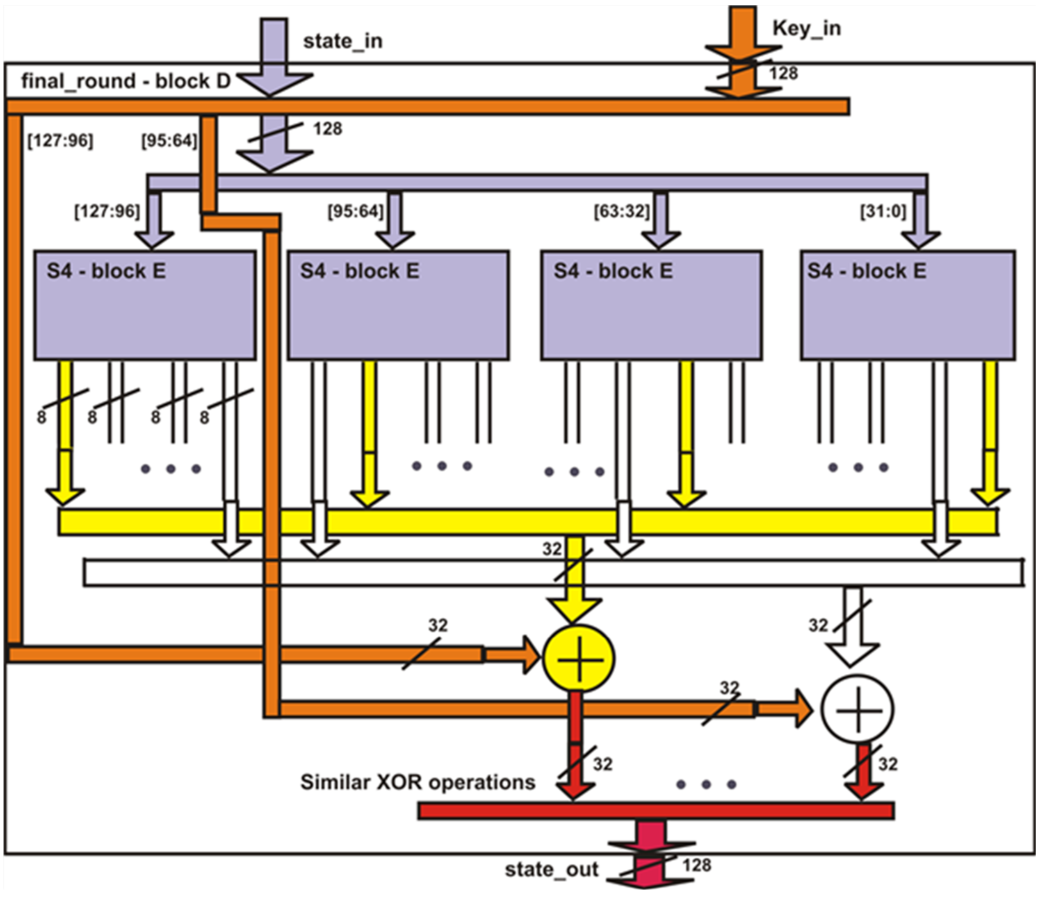


Fig. 6.7 AES crypto-chip block level design - block D

During the second phase of the methodology, RTL correct simulation of the entire system has been performed. For each combinational and sequential sub-block of all nine AES blocks, we have extracted the input vectors and the associated output vectors and we have stored them in several files. During this step, the simulation of block i, which contains at least one instance of block i+1, generates two files corresponding to block i+1: one contains the input vectors applied to all the instances of block i+1, while the other one contains all the correct outputs which corresponds to those inputs.

The logic synthesis phase has been performed for each of the nine blocks of the AES crypto-chip Verilog design. We have generated netlists containing only 2-input NAND gates for the combinational part and only D flip-flops for the sequential parts, in order to use the probabilities of correctness derived for the GL SFI methodology described in chapter 4. The synthesis process has been performed with Synopsys Design Compiler and the ABC synthesis tool, which had generated mapped netlists of each module, represented in terms of inverters, NAND gates, NOR gates and flip-flops. Each inverter and NOR gate of the mapped netlist has been further implemented using only 2-input NAND gates.

During the fourth phase of the reliability methodology, we have inserted mutants which correspond to the most accurate fault model (the input data dependent GISP model) for each NAND gate of the design. As stated in the previous research, the GISP model uses 4 probability values, each one associated to an input transition which determines a logic transition of the output of the gate. We have started with the blocks situated at the bottom of the design and we have performed mutant-based SFI in order to determine their probabilities of failure. The value of the probability of failure has been determined by counting the number of outputs that differed from the correct ones.

A bottom-up approach has been considered in order to build the RTL saboteurs for the last step. The probabilities obtained for one level of the modules hierarchy have been used in order to build the SFI components for the next level of the hierarchy.

6.5 Simulation Results for the AES Crypto-Core

The simulations of the AES crypto-core have been performed on a desktop computer with Intel Core i5 processor at 3.1 GHz and 4 GB of RAM, running Windows 7 and using Modelsim 10.05 SE commercial simulator. The GL SFI of the entire AES crypto-core could not be performed on this system due to the large number of instances (approximately 1,100,000), which will lead to a lack of memory.

Table 6.2 contains the input parameters considered for gate-level mutant insertion. The average probability of failure of a NAND gate has been considered to be 0.3314%, while the average probability of failure of a D flip-flop has been considered to be 0.1251%, values that corresponded to the SPICE based simulations results obtained for 45 nm CMOS technology in paper [49]. Table 6.3 contains the average probability of failure for each of the nine modules of the crypto-core and the simulation time required for each module. Due to the hierarchical structure of the architecture, although the number of input vectors of the crypto-chip is small (100 vectors), the number of input vectors for the blocks situated at the bottom of the design can reach values like thousands or tens of thousands. The second column of table 6.3 shows the number of input vectors that correspond to each block of the design.

Concerning the simulation time required by each module, we have considered the time per run, where one run represents the simulation of a block for only one set of input vectors. This is why the number of input vectors corresponding to one block of the design is equal to the number of runs performed for that block. Due to the hierarchical structure and the different number of instances of each block, the number of simulations that have been performed for the AES blocks varies a lot. 100 input vectors have been applied to the top module (block A), meaning that block A has been simulated 100 times. The instances of block A, namely blocks B, C and D, have been simulated a number of times equal to 100 vectors multiplied by the number of instances of the respective block. Knowing that block A creates 10 instances of block B, 9 instances of block C and 1 instance of block D, it results that block B has been simulated 1000 times, block C has been simulated 900 times and block D 100 times, respectively.

Therefore, the total simulation time for a block is equal with the number of runs (or the number of inputs) multiplied with the simulation time required by each run. The entire simulation campaign, consisting in the combination of RTL simulation and GL simulation, has required approximately 131 minutes, which represents a reasonable total time for an architecture that is composed of 1 million instances of NAND gates and D flip-flops.

Input parameters	**Vdd (V)**	**Delay (ns)**	**Temp (°C)**	**Fault model**	**Average Probability of failure**
NAND Gate	0.30	3.00	50	GISP	0.3314%
D Flip-flop	0.30	2.50	50	GISP	0.1251%

Table 6.2 - Input parameters for gate-level mutant insertion

Module	**No. Of input vectors**	**Output width**	**Components**	**Probability of failure**	**Simulation time**	**Simulation type**
Block I - xS	85435	8	-	9.1250%	33 ms / run	GL
Block F - S box	3952	8	-	9.0429%	27 ms / run	GL
Block H - T	85436	32	1 * block F 1 * block I	11.1357%	51 ms / run	GL + RTL
Block G - table_lookup	3560	128	4 * block H	11.2219%	105 ms / run	GL + RTL
Block C - one_round	900	128	4 * block G	33.9910%	140 ms / run	GL + RTL
Block E - S4	1000	32	4 * block F	9.0721%	88 ms / run	GL + RTL
Block D - final_round	100	128	4 * block E	10.0221%	4 ms / run	GL + RTL
Block B - expand_key_128	1000	128	1 * block E	12.8349%	5.8 ms / run	GL + RTL
Block A - AES	100	128	10 * block B 9 * block C 1 * block D	50.0625%	93 ms/ run	GL + RTL

Table 6.3 - Simulation results for the AES crypto-core and its components

6.6 Conclusions and Contributions Regarding the RTL Analysis

The hierarchical hybrid GL – RTL SFI methodology presented during chapter 6 of this thesis captured the data dependency using GL simulations for small complexity blocks and used these results for the RTL saboteur-based data dependent reliability estimation. It consisted of five main steps: hierarchical block decomposition, RTL correct simulation, logic synthesis, gate level SFI and RTL SFI. The accuracy of the proposed approach with respect to the GL SFI was demonstrated using a medium complexity circuit: a parallel comparator for the CNU processing units. The obtained results, 9.67% overall probability of failure of the comparator for the proposed methodology and 9.33% for the GL SFI, validate the good correlations between the two approaches. The novel methodology has been also tested for a large-complexity system, a 128-bit AES crypto-core containing more than 1 million logic gates and flip-flops, in order to prove the scalability of the approach and to measure the total simulation time required. The work presented during chapter 6 has been published in paper [76].

In conclusion, the described hierarchical SFI methodology has the following advantages with respect to the existing solutions:

- it provides high accuracy characteristic to lower abstraction levels due to the GL simulations performed for each block and sub-block of the design;
- the high accuracy is enhanced by embedding the data dependency concept in the GL simulations; this is achieved by exploiting the different accuracy levels provided by the mutant architectures associated to the 4 fault models defined in chapter 3 and in paper [49]: GOP, GOS, GOST and GISP;
- it maintains a reasonable total simulation time, characteristic to upper abstraction levels, due to its hybrid nature;
- scalability demonstrated for circuits of different complexity.

7. RELIABILITY ANALYSIS OF LOW-DENSITY PARITY-CHECK (LDPC) DECODERS

7.1 Low-Density Parity-Check (LDPC) Codes - General Considerations

According to [71], a parity-check code of length N is defined as a linear binary block code, whose codewords satisfy a set of M linear parity-check constraints. A parity-check code is defined in the literature by its M x N parity-check matrix H, where each of the M rows specify one of the M constraints. The parity-check code represents the set of binary vectors satisfying all constraints, such that $c * H^T = 0$. Furthermore, a low-density parity-check (LDPC) code is defined by a sparse parity-check matrix [71]. Aside from the requirement that H must be a sparse matrix, an LDPC code is no different to other block code. According to [72], finding a sparse parity-check matrix for an existing code is not practical, but LDPC codes are designed by constructing a sparse parity-check matrix first and then determining a generator matrix for the code.

Discovered by Gallager in 1962, during his studies at MIT [72][77] and disregarded until the work of MacKay in 1995 [78], LDPC codes represent a class of capacity approaching codes, with increased error correction capability for both Binary Symmetric Channel (BSC) and Additive White Gaussian Noise (AWGN) channel. LDPC codes are used for a wide range of wired or wireless modern communication standards, like IEEE 802.3an (10 Gbps Ethernet), IEEE 802.3ba (40/100 Gbps Ethernet), IEEE 802.11n (Wi-Fi), IEEE 802.16e (Wi-Max) and DVB-S2 (digital video) [79][80], and lately in NAND based Flash memories [81]. Furthermore, the LDPC codes have the capability to outperform the Turbo codes in various applications and are preferred over other types of codes. As far back as 1962, LDPC codes have been associated with a well-defined iterative decoding scheme, whose complexity grows linearly with block length.

LDPC codes belong to the category of forward error-correction codes. Considering a binary message transmitted over a channel, the concept of forward error control coding means increasing the number of message bits, by deliberately introducing redundancy in the form of extra check bits, producing the binary vector called codeword. The class of resulted codewords is large enough and the transmitted message can be correctly deduced by the receiver, even if some bits of the codeword have been affected by errors [72].

A bi-partite graph called Tanner graph is a visual representation of the parity check matrix H and can conveniently represent an LDPC code. The Tanner graph contains two types of nodes: variable nodes, corresponding to the columns of a sparse parity check matrix denoted by H and the check nodes, corresponding to the rows of the same matrix. The decoding process is performed using message passing algorithms, which consist in the continuous exchange of messages between variable node units (VNUs) and check node units (CNUs).

According to [71], “a regular (j,k) LDPC matrix is an M x N binary matrix, having exactly j ones in each column and exactly k ones in each row, where j<k and

both are small compared to N". By contrast, "an irregular LDPC matrix is still sparse, but not all rows and columns contain the same number of ones" [71]. Consequently, every parity-check equation of a regular LDPC code involves exactly k bits, and every bit is involved in exactly j parity-check equations. The LDPC codes involved in this PhD thesis are all regular. The number of branches starting from a bit node is always j because each bit is involved in j parity checks, while the number of branches starting from each check node is always k, because each parity check uses k bits.

An example of a LDPC matrix with wordlength N=10, j=2 and k=4 is the following:

$$H = \begin{bmatrix} 1\,1\,0\,1\,0\,0\,1\,0\,0\,0 \\ 1\,0\,1\,0\,1\,0\,0\,1\,0\,0 \\ 0\,1\,1\,0\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,1\,1\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,1\,1\,1\,1 \end{bmatrix}$$

The Tanner graph associated with this 5 x 10 LDPC matrix is represented in fig. 7.1:

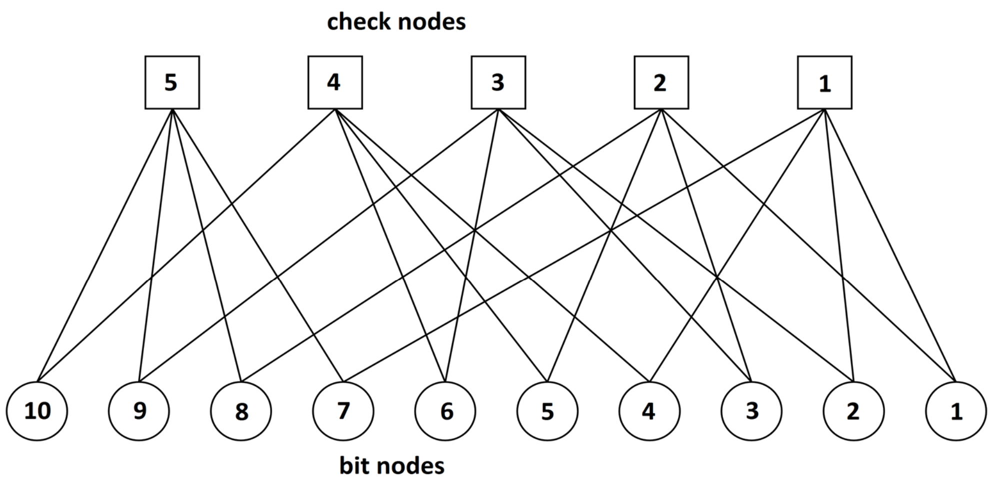


Fig. 7.1 The Tanner graph associated with the LDPC matrix H

The most widely used decoding technique is based on message-passing algorithms, which consist in the transfer of messages between nodes in the Tanner graph. Each node acts as an independent processing unit, receiving incoming messages and computing outgoing messages. Each bit or check node sends a message as soon as all necessary incoming messages have been received. For a cycle-free graph, the message-passing algorithms are recursive and they are always converging to the true a-posteriori log-likelihood ratios after a certain number of messages have been passed between the nodes [71]. On the other hand, most codes used in practice have cycles in their associated Tanner graphs and the message-passing algorithms applied for these codes become approximate, instead of being exact.

LDPC decoding can be performed using two different scheduling strategies: the flooded scheduling and the layered scheduling. The layered strategy requires a reduced number of stored messages, therefore a low number of memory bits and it displays a good convergence due to the high number of updates on the a-posteriori log-likelihood messages. On the other hand, the LDPC decoders based on flooded scheduling present high reliability to hardware faults, which transforms the flooded approach in the preferred one for applications with high fault tolerance requirements.

The majority of LDPC decoders hardware implementations rely on the Min-Sum (MS) algorithm or some enhanced versions of it, namely: Normalized MS (NMS) or Offset MS (OMS) [83]. These algorithms have proven to be the most suitable for hardware implementations, because they are based on operations such as additions and comparisons on a small number of bits.

Regarding modern LDPC decoder implementations, table 7.1 gives a brief overview of the performance characteristics achieved by some of them.

Communication standard	Silicon area	Technology node	Throughput	Dissipated power
WiMAX	4.84 mm^2	130 nm	955 Mb/s	340 mW
IEEE 802.15.3c	1.56 mm^2	65 nm	5.79 Gb/s	360 mW
10 Gb Ethernet	5.35 mm^2	65 nm	47 Gb/s	2.8 W

Table 7.1 – Modern LDPC decoders performance characteristics (data courtesy of [79])

In the case of flooded MS decoding, the corresponding processing consists in the following steps: the CNUs compute the check node messages, denoted as β, based on the incoming messages, denoted as α, which are received from the VNUs; the updated β messages are passed to the VNUs, which will compute the next set of α messages. The decoder uses the channel input log-likelihood ratios (LLRs), denoted as γ, as inputs. These steps are described by the following equations:

1. Initialization $\alpha_{i,k} = \gamma_i$ (1)
2. Check node update:

$\beta_{i,k} = \prod \operatorname{sign}(\alpha_{k,j}) \times \min(|\alpha_{k,j}|)$ (2)

3. Variable node update:

$\alpha_{i,k} = \gamma_i + \sum \beta_{k,j}$ (3)

4. A-posteriori update $\check{\gamma}_i = \gamma_i + \sum \beta_{k,i}$ (4)

$\alpha_{i,k}$, $\beta_{i,k}$ represent the variable node/check node message from node i to node k, while $\alpha_{k,j}$, $\beta_{k,j}$ represent all variable node and check node messages to node i except $\alpha_{k,i}$, $\beta_{k,i}$. An iteration consists of the steps 2, 3 and 4. Decoding stops whether a codeword was found, if a stopping criteria is implemented, or the maximum number of iterations has been reached.

7.2 RTL Saboteur-Based SFI for Fault Tolerance Analysis of a Flooded Min-Sum LDPC Decoder

Paper [84] proposes a methodology for timing errors analysis of complex circuits described at RTL, which consists in three main phases: (i) statistical static timing analysis (SSTA) for standard cell components, (ii) estimation based on probability density function (PDF) propagation for characterization of combinational blocks and (iii) simulated fault injection (SFI) performed at RTL. The work carried out by me for the experiments described in this paper is situated at the RTL, namely for the simulated fault injection step. All the steps of the methodology are included in fig. 7.2.

The analysis has been performed at three layers of abstraction as follows:

1. At circuit level, the authors belonging to our research group have performed SSTA based on Monte-Carlo SPICE simulation, in order to extract the propagation delay distribution for PVT variations for each standard cell component.
2. At gate level, the worst propagation path is determined for each primary output of each combinational block; for each primary output, the delay distribution is derived using a linear composition of PDFs corresponding to standard cell gates.
3. At RTL, probabilistic saboteurs are inserted in the RTL description of the circuit, on each primary output of the combinational blocks.

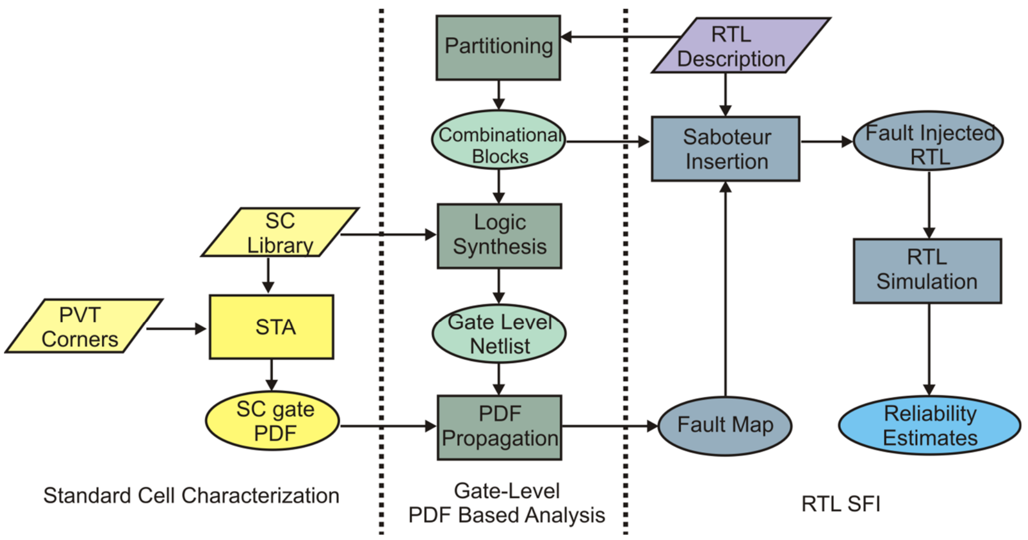


Fig. 7.2 Three layers reliability assessment flow, [84]

For the first step of the proposed methodology, Monte-Carlo SPICE simulations have been carried out in order to derive an Inverse Gaussian distribution for standard cell components. This way, the probability delay characteristic of the standard cell components has been modeled. The inverse Gaussian based PDF has been proved to fit accurately with respect to Monte-Carlo simulations results for both sub-powered circuits and circuits functioning at the nominal supply voltage. Additionally, linear composition of inverse Gaussian distribution of gates which compose a combinational circuit can be used for deriving the PDF of the entire circuit.

During second phase, gate level analysis is used to derive the error probabilities for each primary output of the combinational block. The PDF of each primary output is derived by using a linear composition of the components on the worst delay path for that specific output. The error probability of the primary output for a given delay constraint is determined using the Cumulative Distributed Function (CDF) of the inverse Gaussian distribution.

The error probabilities for different combinational blocks which operate in the same clock domain are obtained from their primary output's CDF, applying the same timing constraint.

For the RTL SFI step, the value or timing characteristics of one or more signals are altered using saboteurs, which are applied at the inputs of sequential / memory components. The proposed saboteurs perform the following functions:

- signal switch detection – timing errors manifest when transitions occur at the outputs, so it is essential to detect them;
- generation of random numbers, which are used to simulate the probabilistic nature of the timing errors;
- logic XOR modules, which is used to selectively alter the desired signals.

The proposed methodology has been applied for the reliability assessment of a Min-Sum (MS) LDPC decoder, which implements a flooded scheduling. The flooded MS decoding scheme consists in permanent exchange of messages between the CNU and the VNU processing units of the LDPC decoder. This message passing takes place for several iterations, until a codeword is found or the maximum number of iterations is reached. The MS decoder under test disposes of serial processing for both variable node messages (denoted as α) and check node messages (denoted as β). The input of the decoder is the log-likelihood ratios (LLR), denoted as γ. The hard decision bits, which represent the output of the decoder, are the signs of the a-posteriori LLR, denoted as γ.

The decoder architecture has been built for a quasi-cyclic (QC) (3,6)-regular LDPC code, with code length of 1296, code ratio of ½ and circulant matrix size of 54. The base matrix for this code contains 24 columns and 12 rows, while the parity check matrix contains 1296 columns and 648 rows.

The modules which compose the LDPC design appear in fig. 7.3 and are described as follows:

1. The Input Log Likelihood Ratio (LLR) memory is used to store the input messages, which will be used in the decoding process for VNU computations; one memory word stores a number of circulant size (54) γ messages, each one represented with a quantization of 4 bits.
2. The VNU processing block contains 54 individual VNU units, which compute the corresponding variable-to-check messages (α) for a column in the base matrix.
3. The α messages memory stores the variable check messages, which will be used in the check node computations; one memory word from this

memory stores a number of circulant size (54) ɑ messages, each one represented with a quantization of 4 bits.

4. The ɑ message barrel shifter represents the routing network between the VNU outputs and the CNU inputs; it consists of 6 multiplexer levels.
5. The CNU processing block is similar to the VNU processing block, containing 54 individual CNU units, which compute the corresponding β for a row in the base matrix.
6. The β messages memory stores the messages used in the VNU processing; one memory word stores 54 x 15 bits, which is equal to the circulant size multiplied by compressed β message size.
7. The β messages barrel shifter represents the routing network between the VNU outputs and the CNU inputs.
8. The hard-decision memory contains the hard-decision bits which are obtained after each iteration and contribute to the LDPC decoder output generation.
9. The global control unit provides the appropriate sequence of operations that must be performed according to the MS flooded algorithm implemented.

The proposed methodology was used to evaluate the error correction capability of an overclocked sub-powered flooded MS LDPC decoder. Considering a Binnary Additive White Gaussian Noise (BiAWGN) channel model and signal-to-noise ratio (SNR) values of 1 to 3 dB, the simulations indicate that increasing the operating frequency by a factor of 2 with respect to the maximum frequency allowed by the fault-free decoder, will not affect the error correction capability.

Taking into account the similarities between the methodology depicted in chapter 6 and this one, we can claim that both of them use a multi-level hierarchical approach, which uses the behavioral data at circuit level in order to determine the higher abstraction levels results. Both of them rely on SPICE simulations performed at circuit level, which represent the starting point for the higher levels methodologies.

The main difference between the two methodologies is that the one described in chapter 6 treats the logic gates independently, without considering how the propagation of errors influences the results on the critical paths. On the other side, the methodology presented in this chapter is based on analytical methods which combine the PDFs of the gates situated on each primary output’s chain. In the case of the second methodology, CDF is used to determine the error probability for each primary output, for a given timing constraint.

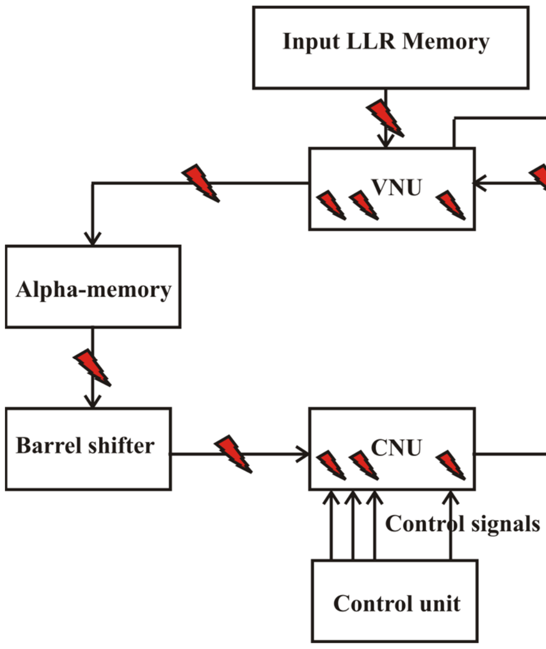


Fig. 7.3 Flooded MS LDPC decoder architecture, [84]

7.3 Increasing the Efficiency of BRAM Utilization in a Memory-Oriented LDPC Flooded Architecture

Various LDPC decoder implementations in hardware have been proposed in the literature. The simplest implementation is represented by the fully-parallel approach, where each check node and variable node in the Tanner graph has a corresponding processing unit in hardware. The main advantage of this approach is the high throughput, but the high overhead brought by the implementation of the interconnection network makes it less suitable in practice. In contrast, partially parallel decoders use a fixed and relatively small number of processing units, they achieve lower throughputs than fully-parallel implementations and they require less hardware resources [80]. In order to reduce the amount of resources required by the LDPC decoder, serialization at processing unit level is used, which affects the throughput of the decoder. Due to serialization, the usage of memory units is required, for storing the messages exchanged by the nodes.

In order to quantify the amount of resources required by a LDPC decoder found in practice, we can consider the following example, described in [82]: the (8176,7156) LDPC code used in NASA LANDSAT and cruise exploration shuttle mission. Each iteration consists of two phases (the check node and variable node processing), while each phase requires reading and writing messages associated to each edge. The Tanner graph associated with this LDPC code has 32704 edges, which means approximately 32704 x 4 = 131086 messages have to be read and written during each iteration in order to decode the codewords. Therefore, the bandwidth of the memory modules becomes the limiting factor of the decoding throughput. Modern implementations of LDPC decoders are tailored for FPGA devices, which dispose of hundreds of embedded memory blocks. In a Xilinx FPGA, these embedded memory blocks are called block RAMs (BRAMs), while in an Altera FPGA, they are called embedded array blocks [82].

Most of the LDPC decoder implementations for FPGAs, found in the literature, use the embedded memory blocks for storing messages, but don't succeed to take full advantage of the BRAMs' aspect ratio configurability feature. In a Xilinx Virtex 4 FPGA, for example, each 18 Kb size BRAM can be configured to operate as a 512 x 36, 1K x 18, 2K x 9, 4K x 4, 8K x 2 or 16K x 1 memory block. Most FPGA implementations store only one message per memory word, so they waste a large percentage of the memory bandwidth.

In order to improve the memory usage of the architectures found in literature, several optimizations have been discussed. One of them is represented by overlapping the processing of alfa and beta messages, in order to have only one memory block for storing both check-node messages and variable-node messages. Memory conflicts can be avoided by using waiting time minimization algorithms, which require the computation of a critical parameter called the waiting time, which finds the certain moment when the overlapping message passing can occur. The authors in [82] claim that overlapped message passing can double the throughput of the decoder, with respect to the baseline reference. However, the papers dealing with this technique suggest that not all LDPC codes exhibit good performance by employing it, so the technique is believed to be code dependent.

Another optimizations discussed in the literature are represented by vectorization and folding. Vectorization means packing multiple messages into a single memory word, while folding represents the technique for which messages corresponding to several circulants in the base matrix are packed into the same BRAM. The first technique, vectorization, takes advantage of the configurable width of the BRAM, while the second one, folding, exploits the configurable depth of the BRAM [82].

In order to optimize the BRAM blocks utilization, I have proposed a new multi-codeword LDPC architecture which has the following particularity: it stores multiple messages corresponding to multiple codewords in the same memory word. For example, in the case of a 36-bit BRAM memory width and a quantization of 4 bits, messages belonging to a maximum of 9 codewords can be stored in the same BRAM memory word. This architecture employs parallel processing units (CNUs and VNUs) in order to process the messages corresponding to different codewords. The number of VNUs contained by the entire architecture is equal to the number of columns in the base matrix multiplied by the number of codewords that must be processed in parallel, while the total number of CNUs is equal to the number of rows in the base matrix multiplied by the number of codewords. The block schematic of the proposed architecture is shown in fig. 7.5. The proposed BRAM memory usage methodology is graphically represented in fig. 7.4.

Gamma and HD memory blocks are used to store the input LLR messages and the hard decision bits; the number of such modules contained by the architecture is equal to the number of rows in the B matrix. The width is *gamma_quant* bits for gamma memory and 1-bit for HD-memory.

A single control unit is used to generate the start signals, the memory address and the read / write enable signals for each VNU processing unit, regardless of the number of codewords processed in parallel, because the sequence of signals is the same, no matter how many codewords are desired to be processed simultaneously. The same applies for the CNUs.

I have synthesized the proposed LDPC decoder architecture for several message quantization values and several number of codewords, as follows: 1, 4, 6, 9 and 18 codewords for a 4-bit message quantization and 1, 6, 12 and 24 codewords for a 3-bit message quantization. The target device has been a Xilinx Virtex 7 FPGA, model XC7VX485T, using the Xilinx ISE 14.7 tool. Two LDPC matrices were considered: a (3,6) LDPC code with a regular base matrix with 3 rows, 6 columns, expansion factor m=256 and codeword size of 1536, and a (3,6) regular LDPC matrix with 12 rows, 24 columns, expansion factor m=54 and codeword size of 1296.

The synthesis results are included in tables 7.2 and 7.3. The number of LUT flip-flop pairs used display a linear increase with the increase of the number of codewords, while the number of BRAMs used remain the same for 1, 4, 6 or 9 codewords. The number of BRAMs doubles in the case of 18 codewords due to the type of primitives used by Xilinx in the synthesis process.

The experiments and results presented in this section have been published in conference paper [85].

Tables 7.2 and 7.3 also contain the decoding throughput of the LDPC decoder, which has been computed using the following formula:

$T = N_C \, x \, L_C x \, {}^{f_{max}}/_{N_{iter}} \, x \, N_{cycles}$. N_C represents the number of codewords processed in parallel, L_C represents the code length, N_{iter} is the number of iterations and N_{cycles} is the number of cycles / iteration.

LDPC code: (3,6) regular; code length = 1296; circulant size = 54				
Decoder type	LUT-FF Pairs	Block RAMs	Frequency [MHz]	Throughput [Mbps]
NC=1; Q=4 bits	4117	49	169.68	101.80
NC=4; Q=4 bits	14996	50	159.20	382.08
NC=6; Q=4 bits	22442	50	159.20	573.12
NC=9; Q=4 bits	33341	51	159.20	859.68
NC=18; Q=4 bits	66569	102	180.73	1951.92
NC=1; Q=3 bits	3827	49	186.75	112.05
NC=6; Q=3 bits	19659	50	185.82	668.94
NC=12; Q=3 bits	35881	52	180.71	1301.04
NC=24; Q=3 bits	78299	104	233.04	3355.68

Table 7.2 – Synthesis results for the proposed decoder for code length 1296 and circulant size 54

LDPC code: (3,6) regular; code length = 1536; circulant size = 256				
Decoder type	LUT-FF Pairs	Block RAMs	Frequency [MHz]	Throughput [Mbps]
NC=1; Q=4bits	1206	13	194.96	29.24
NC=4; Q=4bits	4392	13	195.36	117.20
NC=6; Q=4bits	6502	13	196.17	176.52
NC=9; Q=4bits	9638	13	195.52	263.88
NC=18; Q=4bits	21096	26	231.33	624.60

Table 7.3 – Synthesis results for the proposed decoder for code length 1536 and circulant size 256

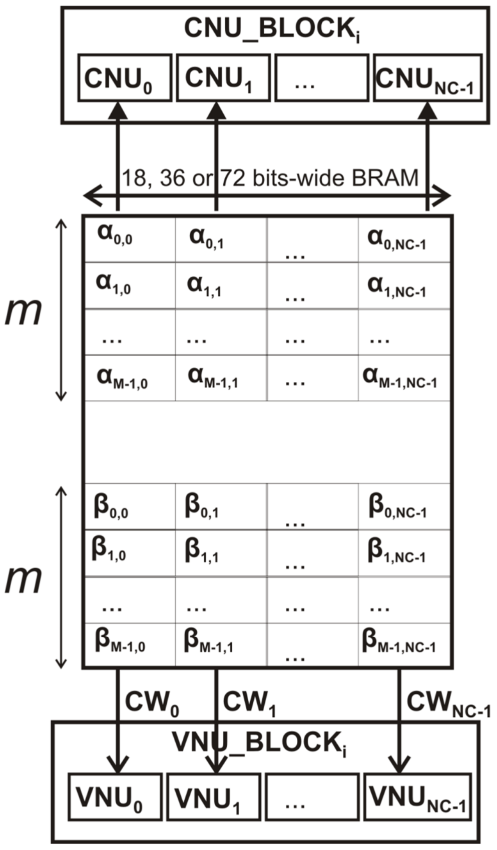


Fig. 7.4 Memory organization for proposed LDPC decoder, [85]

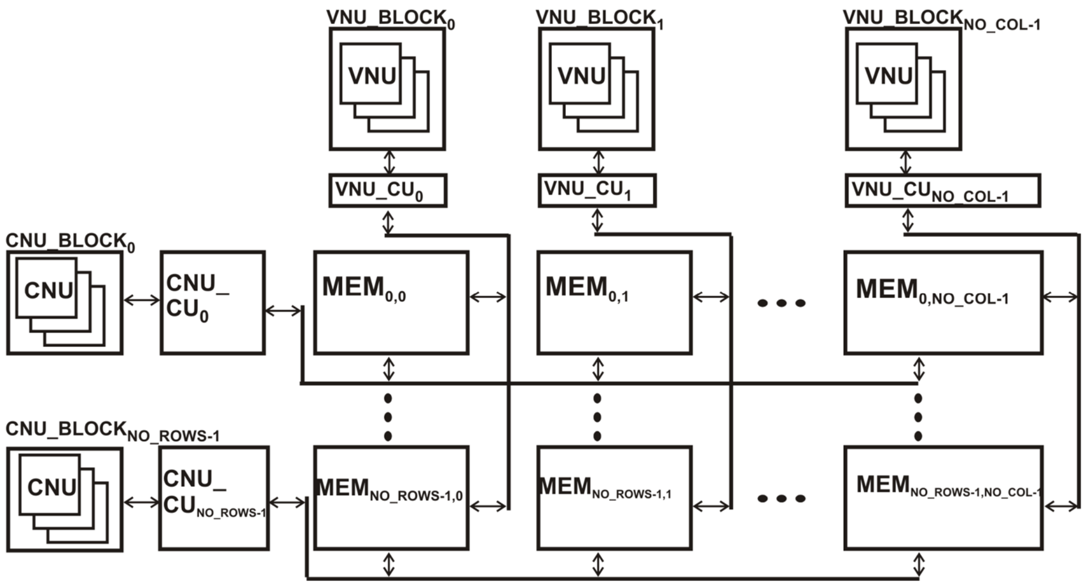


Fig. 7.5 Multiple codeword partially parallel LDPC Decoder, [85]

7.4 Memory-Oriented Simulated Fault Injection for LDPC Architectures

An LDPC decoder has a complexity that exceeds the one of a processing core. This is justified by the fact that a standard flooded architecture of an LDPC decoder is composed of at least three types of memory modules, each type having tens of instances, two types of processing units, usually one for each non-zero element of the base matrix and multiple control units. In the conventional model of communications, the channel on which data is transmitted is considered to be affected by different noise disturbances, while the hardware performing the forward error correction (FEC) or the modulation / demodulation is considered to be fault free. However, the reliability issues that affect today's nanoscale devices supplied at low voltages, rise another important question: what happens if the coder / decoder itself is based on low-powered components? How is the capability of error correction affected? Are the LDPC decoders capable of correcting errors affecting memory modules within the decoder? In order to answer these questions, the reliability assessment of LDPC decoders based on probabilistic sub-threshold or near-threshold circuits becomes a topic that has to be tackled. Different fault models can be considered and different results may be obtained depending on the module of the decoder which is being injected and the frequency of occurrence of faults.

For the previously discussed architecture, we have considered that the memory blocks are implemented using D flip-flops and we have run several SFI campaigns in order to assess the reliability characteristics of the LDPC decoder under probabilistic storage errors. It is known that conventional SRAM-based memories don't function at low supply voltages, so we have taken into account the probabilities of failure for D flip-flops extracted using SPICE simulations in [48] and [50], in order to simulate the faulty memories. The considered fault model is represented by a probabilistic fault generated by the timing violations that occur in a flip-flop based memory. I have applied an equal error rate per memory bit to all three memory types: alfa, beta and gamma.

The probabilities of failure considered for this experiment correspond to LDPC clock frequencies of 400 MHz, 450 MHz and 500 MHz, respectively. The values of the error rates are: 1.25×10^{-3} , 2.4×10^{-3} and 4×10^{-3} per clock cycle, per memory bit. The estimated number of errors for each decoding iteration is shown in fig. 7.6 as follows: the blue column corresponds to the entire decoder, the red column corresponds to the case when only the gamma memories are injected and the green column corresponds to the case when alfa or beta memories are injected.

The simulation environment for this LDPC architecture consists of Modelsim commercial simulator and a C++ transmission channel model composed of: a random word generator, an encoder for LDPC encoding the random word, a BiAWGN channel error model and a results analyzer module, which compares the output of the decoder with the input of the encoder. A System Verilog simulation framework is used for interfacing the LDPC RTL model under test with the C++ transmission channel model. We have computed the bit error rate (BER) and the frame error rate (FER) metrics, which indicate the error correction capability of the faulty LDPC decoder. The signal-to-noise ratio (SNR) considered in this experiment varies from 1 to 3. For each scenario, more than 300.000 frames have been simulated.

The FER performance of the faulty LDPC decoder, with all the memories injected, is depicted in fig. 7.7. A graceful degradation of the decoding performance is obtained with the increase in operating frequency. The error correction capability of a fault-free decoder is plotted in yellow, while the reduced error correction capability of faulty decoders operating at 400 MHz, 450 MHz and 500 MHz is represented with blue, green and purple, respectively.

Figures 7.8, 7.9 and 7.10 show the FER performance of the faulty LDPC decoder, when each of the three types of memory blocks are injected (alfa, beta and gamma), as follows: fig 7.8 corresponds to a frequency of 400 MHz, fig. 7.9 is plotted for a frequency of 450 MHz and fig. 7.10 corresponds to 500 MHz.

The results show that faults injected in the alfa-memory lead to a slightly lower decoding performance than faults injected in the beta-memory. Furthermore, an LDPC decoder with a faulty LLR input memory (gamma memory) has a insignificant improved error correction capability with respect to a decoder with faulty alfa or beta memories. These results are consistent with the ones obtained in [87], as well as the analytical based reliability analysis [88].

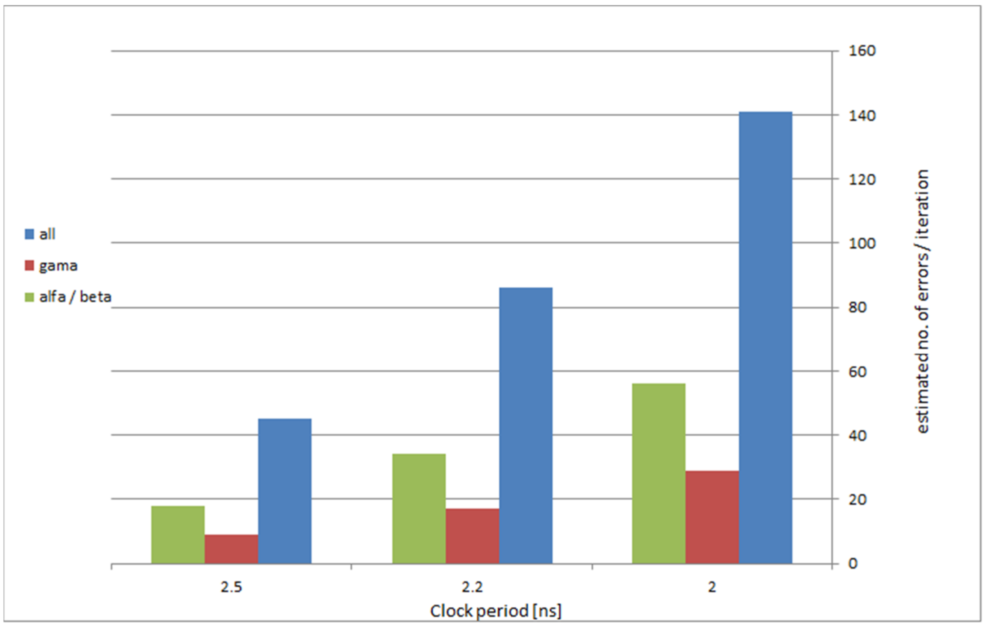


Fig. 7.6 Estimated number of errors per decoding iteration

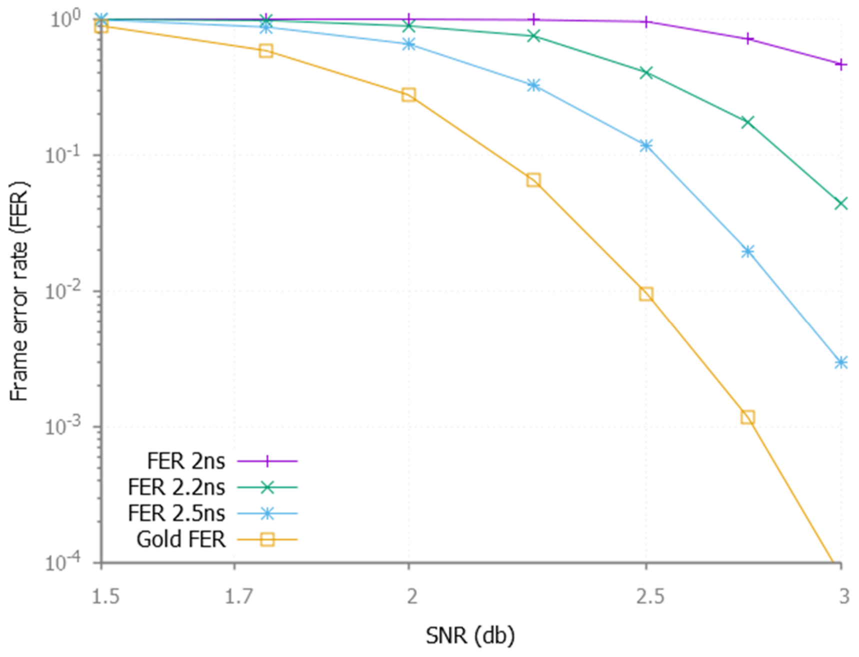


Fig. 7.7 FER performance of faulty LDPC decoder

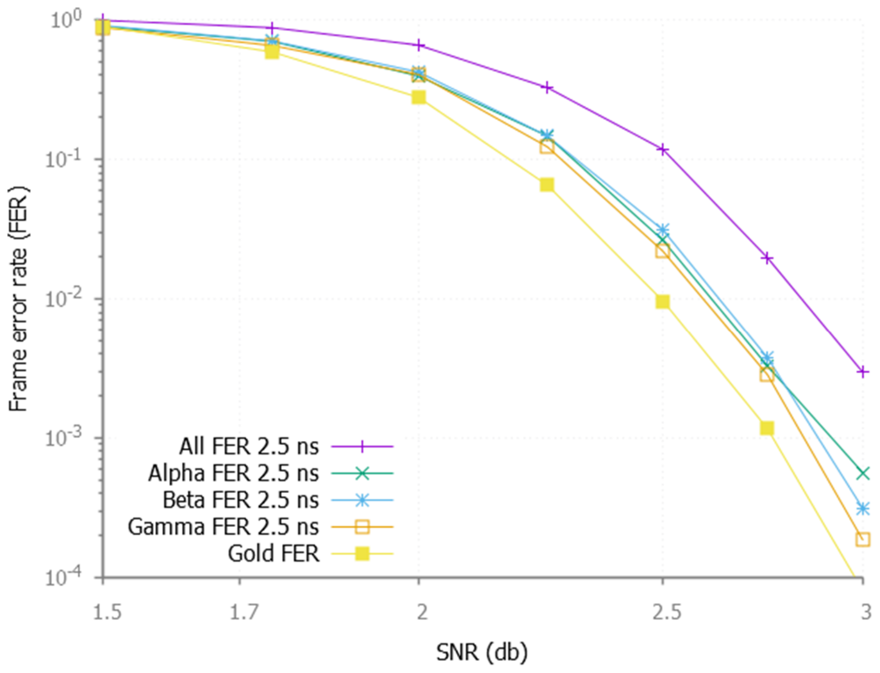


Fig. 7.8 FER performance for 2.5 ns clock cycle period

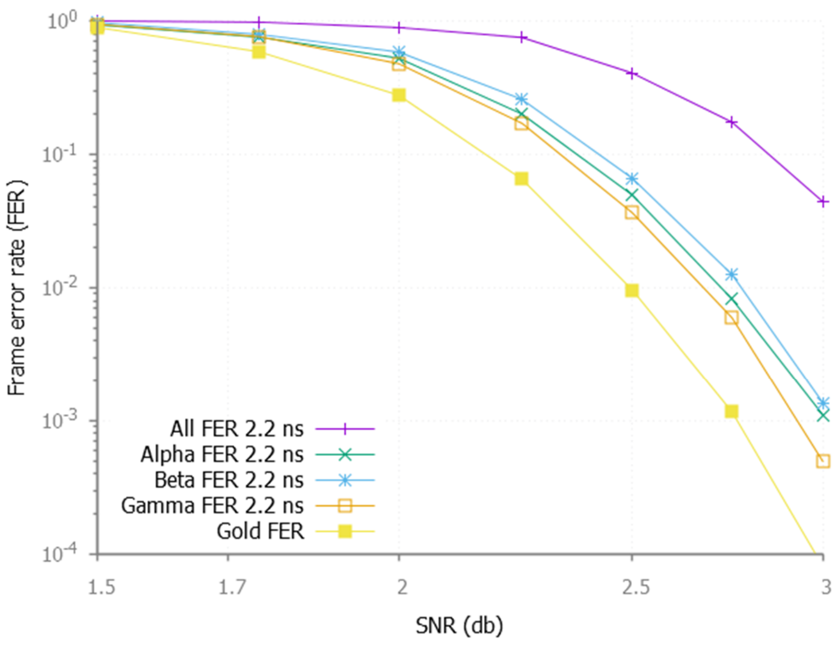


Fig. 7.9 FER performance for 2.2 ns clock cycle period

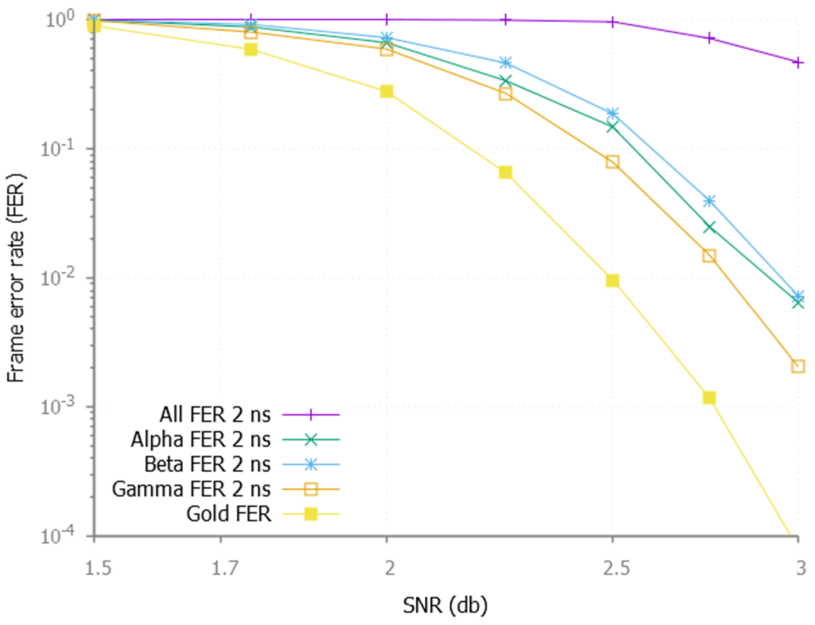


Fig. 7.10 FER performance for 2 ns clock cycle period

7.5 Reliability Assessment of LDPC Decoders Using the Hierarchical Hybrid Methodology

Using the principles of the hybrid hierarchical methodology described in chapter 6, for assessing the reliability of a parallel comparator and an AES crypto-core, in this chapter I propose a new set of experiments for determining the reliability of the previously described LDPC decoder.

The set of experiments detailed in the previous paragraph assume, in a simplistic manner, that faults affect only the memory modules of the decoder, implemented using D flip-flops. For a complete reliability analysis of the decoder, the experiments must be also extended to the processing units. The methodology used in this paragraph comprises the following steps:

1. The division of the LDPC processing units (VNUs and CNUs) into combinational and sequential sub-blocks; each pipeline stage represents a sequential sub-block, while the logic contained between two pipeline registers represents a combinational sub-block.
2. The logic synthesis of the combinational sub-blocks, using Synopsys Design Compiler.
3. Critical and non-critical path extraction from synthesis timing reports for each combinational sub-block.
4. A delay constraint is applied for the processing units and the appropriate delay is associated for each gate, depending on its appurtenance to a critical path (the path with the maximum number of gates) or a non-critical path (paths with fewer gates than the critical one). In order to achieve this, a C program is used for processing the timing report (generated by Synopsys Design Compiler) and the fault-free netlist and for generating the faulty netlist, with one mutant inserted for each gate, according to the required delay. The probabilities of failure for each gate are the ones determined in [49], as a function of supply voltage, temperature and delay.
5. Parallel simulation of the fault-free processing unit and the faulty gate-level processing unit, the approach being described in fig. 7.11. For this step, a dedicated CNU testbench and a dedicated VNU testbench are employed. The faulty processing unit is composed of faulty combinational sub-blocks and correct sequential sub-blocks; the faulty combinational sub-blocks receive only correct inputs, in order to measure the probability of failure of each sub-block independently, regardless of the faults that might propagate from one sub-block to another.
6. Extraction of the probability of failure for each bit of each primary output of the sub-blocks of both CNUs and VNUs.
7. RTL saboteur-based simulated fault injection of the entire decoder and FER plotting, using the previously determined probabilities of failure for each combinational sub-block.

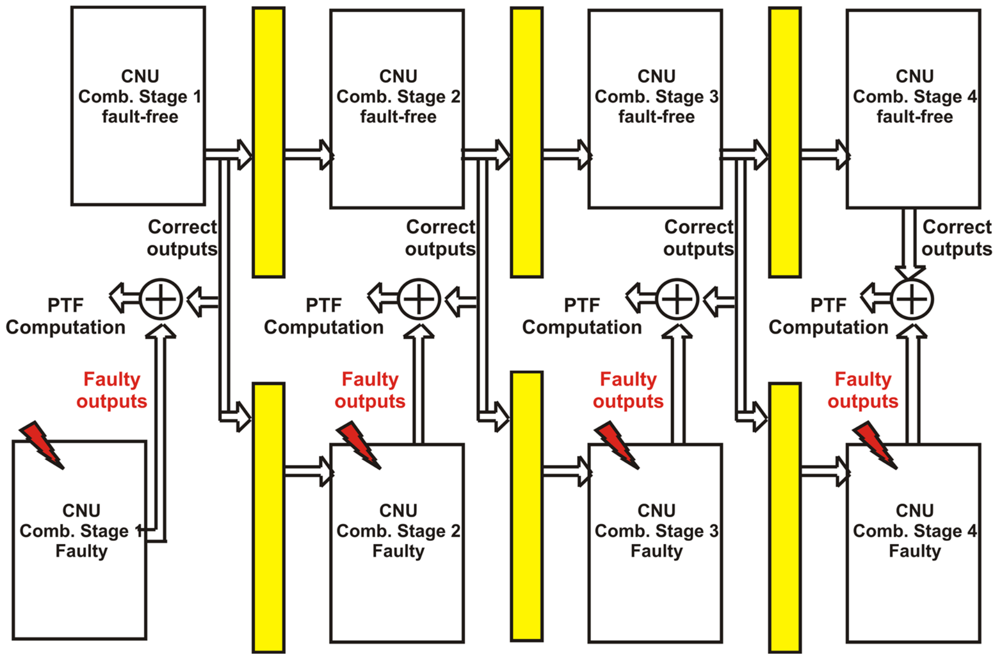


Fig. 7.11 Probability of failure computation for each sub-block of the CNU processing unit

The CNU processing unit has been divided in four combinational sub-blocks. The longest critical path belongs to the third combinational sub-block and it consists of 37 gates. Similarly, the VNU processing unit has been divided in three combinational sub-blocks.

These sets of experiments have been carried out using Modelsim commercial simulator and the C++ transmission channel model previously described. The gate level simulation has been performed for a supply voltage of 0.35 V, a temperature of 50 degrees Celsius and the delay constraint for one combinational block has been chosen to be equal to 200 ns, 133 ns and 100 ns, respectively. One set of experiments consisted in simulating the entire LDPC decoder with faults injected only in the CNU modules, the second set targeted the VNU modules and the third set considered faults injected in both CNUs and VNUs.

Figures 7.12, 7.13 and 7.14 show the FER performance of the faulty LDPC decoder, under the scenarios explained above, as follows:

- fig. 7.12 contains the FER of the LDPC decoder with CNU processing units injected for delay constraints of 200 ns, 133 ns ans 100 ns, respectively;
- fig. 7.13 contains the FER of the LDPC decoder with VNU processing units injected for the same delay constraints;
- fig. 7.14 contains the FER of the LDPC decoder with both types of processing units injected simultaneously, for delay constraints of 200 ns and 133 ns.

From fig. 7.12 we can see how the error correction capability degrades when lowering the delay constraint for the combinational modules from 200 ns to 100 ns. The scenario is realistic because each gate in the design has a different delay, depending on its position within or outside the critical paths. For a delay constraint of 100 ns, the error correction capability degrades significantly. Considering a FER value of 10^{-2} , the error correction capability of the faulty LDPC decoder, for a constraint of 200 ns on CNU, degrades with approximately 0.1 dB, with respect to the FER of the fault-free decoder. The degradation is equal to about 0.3 dB for a constraint of 133 ns on CNU and it increases to over 0.5 dB for a constraint of 100 ns. The error correction capability degradation tendency follows the one specified in theory [86].

Regarding the VNU results, we notice from fig. 7.13 that faults injected in the VNU processing unit affect in a different way the error correction capability of the LDPC decoder. Precisely, an error floor is obtained at a FER of 10^{-1} (the violet curve in the graph). The error floor induced by the VNU is even more pronounced for the case of fault injection in both VNU and CNU, depicted in fig. 7.14.

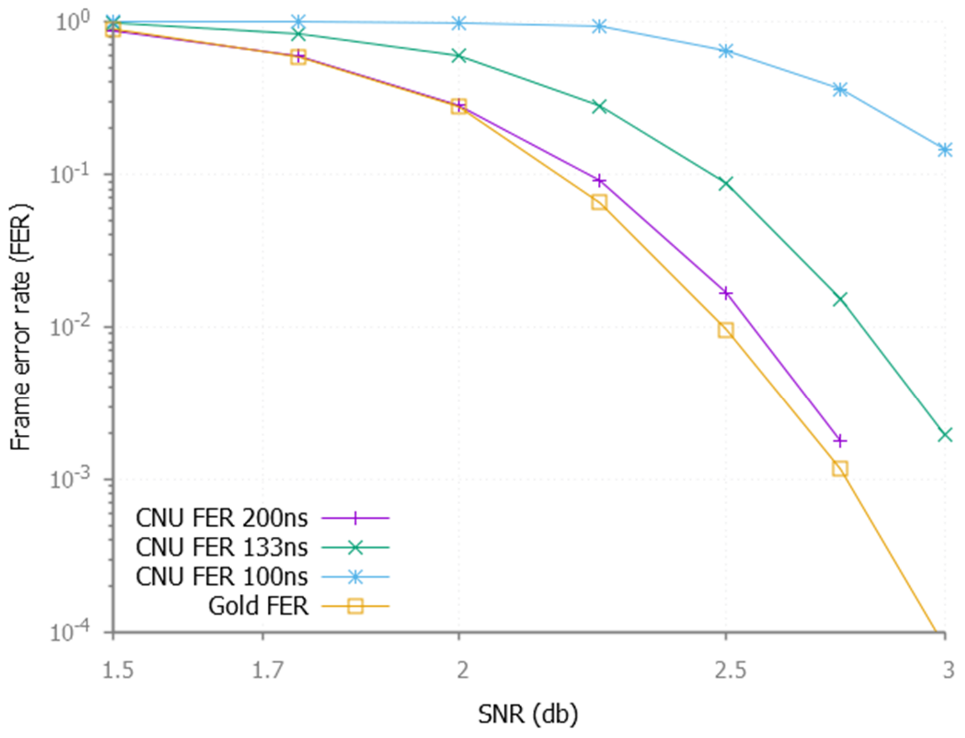


Fig. 7.12 FER performance with CNU injected

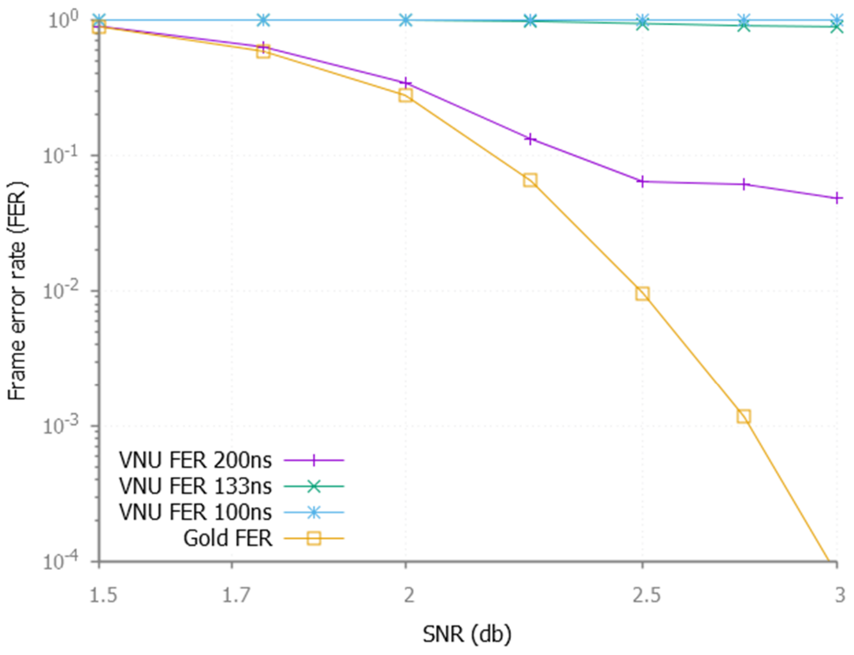


Fig. 7.13 FER performance with VNU injected

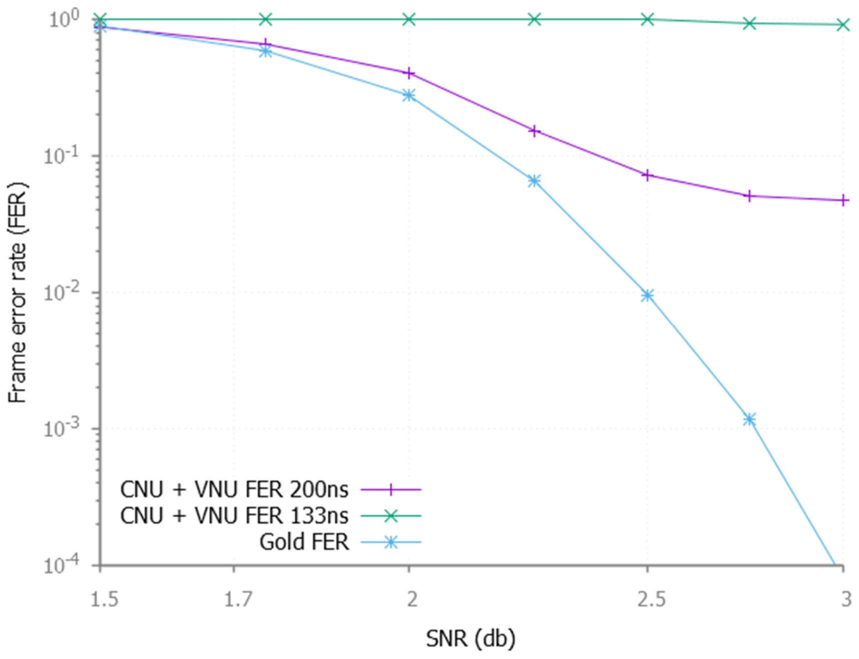


Fig. 7.14 FER performance with both CNU and VNU injected

7.6 Conclusions and Contributions for LDPC Reliability Analysis

Section 7.2 described a novel RTL saboteur-based SFI technique, which relies on analytical models in order to determine the CDF of each primary output. The saboteur-based SFI campaigns carried out in section 7.2 demonstrate important aspects about the reliability of overclocked LDPC decoders supplied at very low voltages and affected by timing errors: the frequency of the decoder can be doubled with almost no decoding capability loss.

Section 7.3 introduces a new LDPC decoder flooded architecture, with the following characteristics:

- efficient memory utilization is obtained by packing multiple messages corresponding to multiple codewords into the same BRAM word;
- up to 9 codewords can be processed in parallel for 4-bit quantization and up to 12 codewords for 3-bit quantization, without introducing significant memory overhead;
- with respect to other LDPC decoders, we use one order of magnitude less BRAM blocks per processed codeword.

Throughout sections 7.4 and 7.5, low-density parity-check (LDPC) decoders are injected with faults according to different patterns and the error correction capability is analyzed. One set of simulations targeted the different types of memory modules of the LDPC decoders, while other set targeted the processing units.

The experiments depicted in paragraphs 7.4 and 7.5 of this thesis bring the following contributions:

- reliability assessment using saboteur-based SFI is performed for the memory modules of an LDPC decoder, which has a greater complexity than a processor core;
- reliability assessment using gate-level mutant-based and RTL saboteur-based SFI is performed for the processing units of an LDPC decoder, in a hierarchical manner;
- the FER performance of the LDPC is monitored under different fault injection assumptions;
- a graceful degradation of the error correction capability is noticed, with the decrease of the delay constraint set for each combinational module; the error correction capability depends also on the module which is being injected;
- the VNU units show a greater vulnerability to transient faults than the CNU units;
- an error floor is obtained at a FER of 10^{-1} for the case of VNU injection.

8. CONCLUSIONS

8.1 The research performed for the PhD

My PhD work addresses one of the hot topics in today's digital era: building reliable circuits from unreliable low-power components. The preferred method for reducing the power consumption of digital CMOS integrated circuits, which are prevalent in nowadays digital devices, is represented by the aggressive scaling of the supply voltage. However, the gain obtained in the energy consumption of the device is counterbalanced by the increased susceptibility to random and systematic variations. Dealing with reliability issues becomes one of the main problems in this context. Therefore, it is imperative to develop efficient reliability assessment techniques at multiple levels of abstraction of a digital system.

Chapter 3 included the research performed at circuit level, in order to extract the probability of failure for sub-powered CMOS logic gates. The gates' behavior has been analyzed under different noise model assumptions. This work has been published in one conference paper [35].

Chapter 4 presented two gate-level simulated fault injection methodologies for reliability assessment of small and medium combinational circuits, one based on mutants insertion and the other one based on simulator commands and scripts. The accuracy of the simulator commands methodology was validated by confronting the results with the ones of the mutant-based approach. This research has been published in conference papers [49] and [51].

Chapter 5 addressed the probabilistic nature of low-powered interconnects and focused on a saboteur-based reliability assessment methodology. This approach was especially created to capture the probabilistic nature of crosstalk induced faults, taking into account the impact of transitions that occur on the lines situated in the vicinity of a certain line. The methodology and the experimental results have been discussed in detail in conference paper [59].

Chapter 6 presented a hierarchical simulated fault injection methodology for reliability evaluation of RTL circuit descriptions, which combined the accuracy of the gate level analysis with the low simulation overhead of the RTL analysis. The methodology has been demonstrated for a complex circuit (an AES crypto-core), which cannot be simulated entirely at gate-level due to the very large number of instances, which require high memory resources. The accuracy of the new methodology was proved for a smaller circuit, a parallel comparator used in the processing units of the LDPC decoders, which was simulated entirely both in the GL manner and the new hierarchical manner. This research has been published in the first journal elaborated during my PhD [76].

Chapter 7 was dedicated to LDPC decoders reliability analysis. In the first part, I presented a new multi-level fault tolerance analysis methodology developed by the researchers in our group, which was applied for a large flooded LDPC decoder architecture. This work has been published in conference paper [84] and my contribution was situated at the third level of analysis, RTL. In the second part, a new FPGA-tailored LDPC decoder flooded architecture has been presented, with the purpose of increasing the BRAM utilization efficiency. The paper describing the characteristics of this architecture has been published in 2016 [85]. This architecture has been further employed for running two categories of fault injection reliability assessment experiments: the first one pointed the memory modules contained by the LDPC decoder, while the second one pointed the processing units.

8.2 Contributions

The contributions at transistor level are:

- the effects of the amplitude and pulse width typical to transient errors in the CMOS circuits operating at low supply voltages are analyzed;
- from the noise amplitude point of view, a decrease in reliability is noticed, with the decrease of the supply voltage, for different noise assumptions, with normal distributions of the amplitude;
- regarding the propagation of transient faults, gates operating at low supply voltages show increased resilience to glitches, despite the fact that the noise margins of the circuits are diminishing; simulations have proven that glitches with shorter duration propagate better for higher supply voltages.

The methodology based on mutant insertion, presented in the first part of chapter 4 can be easily applied for small and medium sized circuits, described at gate-level in hardware description languages. The main advantage is that each logic gate of the design can be tuned independently, using the desired voltage, delay and temperature characteristics and faults can be injected according to one of the 4 fault models presented. The most complex fault model has the advantage of capturing the data dependency, so the circuit under test can be simulated under different dataflow realistic scenarios. The methodology can be used to analyze how the frequency of transitions' occurrences in a certain dataflow may impact the probability of failure of a certain bit of a primary output. Furthermore, the mutant-based reliability assessment methodology has the following advantages:

- low simulation overhead for small and medium complexity netlists;
- high level of accuracy: each gate can have its own parameters; circuits with critical and non-critical paths can be simulated easily;
- the possibility to monitor the probability of failure for each bit of each primary output of a module, independently.

The methodology based on simulator commands, presented throughout the second part of chapter 4 comes as a complement of the first one and has the following main advantages:

- high level of accuracy, validated by confronting it with other techniques;
- easiness in the simulation set-up process;
- high degree of flexibility;
- no overhead for the HDL code of the circuit under test.

The saboteur-based reliability assessment methodology presented throughout chapter 5 is tailored for capturing the probabilistic nature of low-powered interconnects, especially under crosstalk-induced faults and stands out with the following characteristics:

- it provides a low simulation time overhead, with respect to the time required by the simulation of the fault-free circuit;
- it captures in a realistic manner the probabilistic behavior of wires affected by crosstalk faults, by employing 4 types of saboteurs, among which 2 are data-dependent;
- the possibility to take into account the influence of the probability of failure of each wire in a selected vicinity;
- the simulations indicated which are the most critical signals of a Wishbone bus.

The hierarchical RTL reliability assessment strategy proposed throughout chapter 6 has the following advantages:

- it provides high accuracy, characteristic to lower abstraction levels, due to the GL simulation performed for each sub-block;
- it maintains a reasonable total simulation time, characteristic to upper abstraction levels: our approach requires three orders of magnitude less time than the classical GL SFI;
- it is a data-dependent hybrid methodology; the data dependency is ensured by the use of the most accurate fault model presented in paper [49], named GISP, which takes into account different probabilities for each distinct input combination that triggers a 0-to-1 or a 1-to-0 switch of the gate.

The contributions of the RTL saboteur-based SFI technique described in the first part of chapter 7 are related to the analytical models used in order to determine the CDF of each primary output. The saboteur-based SFI campaigns demonstrate important aspects about the reliability of overclocked LDPC decoders supplied at very low voltages and affected by timing errors.

The new LDPC decoder flooded architecture described in section 7.3 has the following characteristics:

- efficient memory utilization is obtained by packing multiple messages corresponding to multiple codewords into the same BRAM word;
- up to 9 codewords can be processed in parallel for 4-bit quantization and up to 12 codewords for 3-bit quantization, without introducing significant memory overhead;
- with respect to other LDPC decoders, we use one order of magnitude less BRAM blocks per processed codeword.

The experiments carried-on in sections 7.4 and 7.5 of this thesis bring the following contributions:

- reliability assessment using saboteur-based SFI is performed for the memory modules of an LDPC decoder, which has a greater complexity than a processor core;
- reliability assessment using gate-level mutant-based and RTL saboteur-based SFI is performed for the processing units of an LDPC decoder;
- the FER performance of the LDPC is monitored under different fault injection assumptions;
- a graceful degradation of the error correction capability is noticed, with the decrease of the delay constraint set for each combinational module; the error correction capability depends also on the module which is being injected;
- the VNU units show a greater vulnerability to transient faults than the CNU units;

- an error floor is obtained at a FER of 10^{-1} for the case of VNU injection.

In conclusion, considering an arbitrary chosen digital system, based on sub-powered CMOS circuits, for which the reliability assessment must be performed, the appropriate methodology should be selected according to the following criteria:

- if we have a small or medium system and the required degree of precision in reliability evaluation is high, the preferred method will be the mutant-based SFI described in the first part of chapter 4;
- if we have a small or medium system and the required degree of precision is not so high, but we target short simulation time, the SFI method based on simulator commands and scripts, described in the second part of chapter 4, should be appropriate;
- if the chosen digital system is a complex one, the entire gate-level simulation may be prohibitive, so the appropriate methodology is the one described in chapter 6, which combines RTL and gate-level simulations, using both mutants and saboteurs.

8.3 Future work

Regarding the research directions that may be tackled in the future, I intend to perform the following tasks:

- to extend the experiments performed in sections 7.4 and 7.5 of chapter 7 to another type of LDPC decoder architecture: a flooded Min-Sum architecture which processes uncompressed β messages;
- the FER performance of the new architecture with memory modules injected and, also, with processing units injected, will be compared with the FER performance obtained in sections 7.4 and 7.5; these experiments may explain which architecture elements increase the vulnerability of the LDPC decoder to transient faults;
- to perform system-level analysis using tools like System C for complex digital systems based on sub-powered circuits, composed of processing cores, memories and I/O interfaces.

List of Publications

- ISI indexed publications
    1. Sergiu Nimara, Alexandru Amaricai and Mircea Popa, "Analysis of transient error propagation in sub-powered CMOS circuits," Microelectronics Proceedings - MIEL 2014, 2014 29th International Conference on, Belgrade, 2014, pp. 375-378, doi: 10.1109/MIEL.2014.6842168.
    2. Sergiu Nimara, Alexandru Amaricai, Oana Boncalo and Mircea Popa, "Probabilistic saboteur-based simulated fault injection techniques for low supply voltage interconnects," Ph.D. Research in Microelectronics and Electronics (PRIME), 2014 10th Conference on, Grenoble, 2014, pp.1-4. doi: 10.1109/PRIME.2014.6872654.
    3. A. Amaricai, S. Nimara, O. Boncalo, J. Chen and E. Popovici, "Probabilistic Gate Level Fault Modeling for Near and Sub-Threshold CMOS Circuits," Digital System Design (DSD), 2014 17th Euromicro Conference on, Verona, 2014, pp. 473-479, doi: 10.1109/DSD.2014.92
    4. L. Dobrescu and S. Nimara, "Predictive models for short channel MOS CASCODE current references simulation," 2014 International Semiconductor Conference (CAS), Sinaia, 2014, pp. 291-294, doi: 10.1109/SMICND.2014.6966464.
    5. Sergiu Nimara, Alexandru Amaricai and Mircea Popa, "Sub-threshold CMOS circuits reliability assessment using simulated fault injection based on simulator commands," Applied Computational Intelligence and Informatics (SACI), 2015 IEEE 10th Jubilee International Symposium on, Timisoara, 2015, pp. 101-104, doi: 10.1109/SACI.2015.7208179.
    6. Alexandru Amaricai et al., "Multi-level probabilistic timing error reliability analysis using a circuit dependent fault map generation," Design of Circuits and Integrated Systems (DCIS), 2015 Conference on, Estoril, 2015, pp. 1-6. doi: 10.1109/DCIS.2015.7388580.
    7. Sergiu Nimara, Alexandru Amaricai, Oana Boncalo, Mircea Popa, “Multi-level simulated fault injection for data-dependent reliability analysis for RTL circuit descriptions,” Advances in Electrical and Computer Engineering (AECE) Journal, vol. 16, issue 1, pp. 93-98, 2016, DOI: 10.4316/AECE.2016.01013.

- IEEE indexed publications:
    1. S. Nimara, O. Boncalo, A. Amaricai and M. Popa, "FPGA architecture of multi-codeword LDPC decoder with efficient BRAM utilization," 2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Kosice, 2016, pp. 1-4, doi: 10.1109/DDECS.2016.7482452.

References

[1] L. B. Kish: “End of Moore’s law: thermal (noise) death of integration in micro and nanoelectronics”, Physics Letters A, vol. 305, pp. 144-149, December 2002

[2] T. Rueckes et al: “Carbon nanotube-based nonvolatile random access memory for molecular computing”, Science, vol. 289, pp. 94-97, 2000

[3] Ethan Mollick, „Establishing Moore’s Law”, IEEE Annals of the History of Computing, issue 3, vol. 28, 2006

[4] Shekhar Borhar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10-16, Nov.-Dec. 2005

[5] C. Constantinescu: “Trends and challenges in VLSI circuit reliability”, IEEE Micro, vol. 23, no. 4, pp. 14-19, July-August 2003

[6] Ronald G. Dreslinski, Michael Wieckowski, David Blaauw, Dennis Sylvester and Trevor Mudge: „Near Threshold Computing: Overcoming Performance Degradation from Aggressive Voltage Scaling”, Workshop Energy-Efficient Design, 2009

[7] S. Borkar, „Designing reliable systems from unreliable components: the challenges of transistor variability and degradation”, IEEE Micro, vol. 25, issue 6, 2005

[8] Zebo Peng, „Building reliable embedded systems with unreliable components”, Proceedings of the International Conference on Signals and Electronic Systems (ICSES), 2010

[9] Lav R. Varshney, „Toward limits of constructing reliable memories from unreliable components”, IEEE Information Theory Workshop (ITW), 2015

[10] K. Palem, A. Lingamneni, C. Enz, C. Piguet, „Why design reliable chips when faulty ones are even better”, Proceedings of the European Solid-State Circuits Conference (ESSCIRC), 2013

[11] Sergiu Nimara, „Sub-powered Probabilistic CMOS Error Modelling and Propagation”, Dissertation Thesis, „Politehnica” University of Timisoara, 2013

[12] Trevor Mudge, „Challenges and Opportunities for Extremely Energy-Efficient Processors”, IEEE Computer Society, 2010

[13] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw et al., „Ultralow-Voltage, Minimum-Energy-CMOS”, IBM Journal of Research and Development, vol. 50, issue 4.5, pp. 469-490, 2006

[14] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, J. Torrellas, „VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages”, 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2012

[15] L. Chang, W. Haensch, „Near-Threshold Operation for Power-Efficient Computing? It Depends...”, 49th ACM / EDAC / IEEE Design Automation Conference (DAC), 2012

[16] B. H. Calhoun, A. Wang, N. Verma and A. Chandrakasan: “Sub-Threshold Design: The Challenges of Minimizing Circuit Energy”, Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), Germany, 2006

[17] Alice Wang, A. P. Chandrakasan, Stephen V. Kosonocky: “Optimal Supply and Threshold Scaling for Subthreshold CMOS Circuits”, Proceedings IEEE Computer Society Annual Symposium on VLSI, 2002

[18] Fady Abouzeid, “Reduction of IPs Energy Consumption with Ultra-Low Voltage Supply”, 10th Conference on Ph.D. Research in Electronics and Microelectronics, Grenoble, France, 2014

[19] Himanshu Kaul, Mark Anders, Steven Hsu, Amit Agarwal et al: „Near-Threshold Voltage (NTV) Design – Opportunities and Challenges”, Design Automation Conference ’49 (DAC), 2012

[20] Alice Wang, Anantha Chandrakasan: “A 180-mV Subthreshold FFT Processor Using a Minimum Energy Design Methodology”, IEEE Journal of Solid State Circuits, vol. 40, no. 1, 2005

[21] Joyce Kwong, Yogesh Ramadass, Naveen Vermal, Markus Koeslerl, Korbinian Huber, Hans Moormann, Anantha Chandrakasan: „A 65 nm Sub-Vt Microcontroller with Integrated SRAM and Switched-Capacitor DC-DC Converter”, IEEE International Solid-State Circuits Conference, 2008

[22] Scott Hanson, Bo Zhai, Mingoo Seok, Brian Cline, Meghna Singhal et al: “Exploring Variability and Performance in a Sub-200-mV Processor”, IEEE Journal of Solid-State Circuits, vol. 43, no. 4, 2008

[23] Amit Agarwal, Steven Hsu, Sanu Mathew, Mark Anders, Himanshu Kaul, Farhana Sheikh, Ram Krishnamurthy: „A 32nm 8.3GHz 64-entry x 32b Variation Tolerant Near-Threshold Voltage Register File”, IEEE Symposium on VLSI Circuits, 2010

[24] Steven K. Hsu, Amit Agarwal, Mark A. Anders, Sanu K. Mathew et al. : “A 280 mV-to-1.1 V 256b Reconfigurable SIMD Vector Permutation Engine With 2-Dimensional Shuffle in 22 nm Tri-Gate CMOS”, IEEE Journal of Solid-State Circuits, vol. 48, no. 1, 2013

[25] Timothy N. Miller, Renji Thomas, Radu Teodorescu: „Mitigating the Effects of Process Variation in Ultra-low Voltage Chip Multiprocessors using Dual Supply Voltages and Half-Speed Units”, IEEE Computer Architecture Letters, vol. 11, no. 2, 2012

[26] Benton H. Calhoun, Sudhanshu Khanna, Randy Mann and Jiajing Wang: „Sub-threshold Circuit Design with Shrinking CMOS Devices”, IEEE International Symposium on Circuits and Systems (ISCAS), 2009

[27] Hendrawan Soeleman, Kaushik Roy: “Ultra-Low Power Digital Subthreshold Logic Circuits”, Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), San Diego – USA, 1999

[28] B. E. S. Akgul, L. N. Chakrapani, P. Korkmaz, K. V. Palem: “Probabilistic CMOS technology: a survey and future directions”, IFIP International Conference on VLSI, pp. 1-6, Nice, October 2006

[29] M. Srivastav, M. B. Henry, L. Nazhandali, “Design of low-power, scalable-throughput systems at near / sub threshold voltage”, 13th International Symposium on Quality Electronic Design, 2012

[30] J. Han, Y. Zang, S. Huang, M. Chen and X. Zeng, „An area-efficient error-resilient ultra-low-power subthreshold ECG processor”, IEEE Transactions on Circuits and Systems – II: Express briefs, 2015

[31] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, J. Torrellas, „VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages”, 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2012

[32] Rodney M. Goodman and Masahiro Sayano: “The reliability of semiconductor RAM memories with on-chip error-correction coding”, IEEE Transactions on Information Theory, vol. 37, pp. 884-896, 1991

[33] N. M. Zivanov and D. Marculescu: “Multiple transient faults in combinational and sequential circuits: a systematic approach”, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, vol. 29, no. 10, October 2010

[34] R. Baumann, “Soft errors in advanced computer systems”, *IEEE Design and Test of Computers*, vol. 22, no. 3, pp. 258-266, May/June 2005

[35] Sergiu Nimara, Alexandru Amaricai, Mircea Popa, „Analysis of Transient Error Propagation in Sub-powered CMOS Circuits”, Proceedings of 29th International Conference on Microelectronics (MIEL), May 2014

[36] G. S. Choi, R. K. Iyer, V. A. Carreno, „Simulated Fault Injection: AMethodology to Evaluate Fault Tolerant Microprocessor Architecturs”, IEEE Transactions on Reliability, vol. 39, no. 4, october 1990

[37] J. C. Baraza, J. Gracia, D. Gil, P.J. Gil, “Improvement of Fault Injection Techniques based on VHDL Code Modification”, Tenth IEEE International High-Level Design Validation and Test Workshop, 2005

[38] Oana Boncalo, „Simulation based reliability assessment of quantum circuits”, PhD Thesis, Ed. Politehnica, 2008

[39] S. Kim, A. K. Somani, „Soft Error Sensitivity Characterization for Microprocessor Dependability Enhancement Strategy”, Proc. Of the International Conference on Dependable Systems and Networks (DSN), 2002

[40] K. G. Shin, Y. H. Lee, “Measurement of fault latency: Methodology and experimental results”, Technical Report CRL-TR-45-84, 1984; Computing Research Laboratory University of Michigan, Ann Arbor

[41] S. Kim, R. K. Iyer, “Impact of device level faults in a digital avionic processor”, AIAA/IEEE *8th Digital Avionics Systems Conference,* 1988

[42] J. Cusick, R. Koga, W. A. Kolasinski, C. King, “SEU vulnerability of the Zilog 2-80 and NSC-800 microprocessors”, *IEEE Trans. Nuclear Science***,** vol NS-32, 1985 Dec, pp 4206-4211

[43] M. Rimen, J. Ohlsson, „A Study of the Error Behavior of a 32-bit RISC Subjected to Simulated Transient Fault Injection”, Proceedings of the International Test Conference, 1992

[44] B. Devlin, M. Ikeda, K. Asada, „Gate-level process variation offset technique by using dual voltage supplies to achieve near-threshold energy efficient operation”, IEEE Cool Chips XV, 2012

[45] J. Gracia, J. C. Baraza, D. Gil, P.J. Gil, “Comparison and application of different VHDL-based fault injection techniques,” Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2001.

[46] M. M. Ibrahim, K. Asami, M. Cho, “Evaluation of SRAM based FPGA performance by simulating SEU through fault injection,” 6th International Conference on Recent Advances on Space Technologies (RAST), 2013

[47] Oana Boncalo, Mihai Udrescu, Lucian Prodan, Mircea Vladutiu, Alexandru Amaricai, “Simulated fault injection for quantum circuits based on simulator commands,” 4th International Symposium on Applied Computational Intelligence and Informatics (SACI), 2007

[48] J. Chen, S. Cotofana, S. Grandhi, C. Spagnol, E. Popovici, “Inverse Gaussian distribution based timing analysis of sub-threshold CMOS circuits”,

Microelectronics Reliability, vol. 55, issue 12, December 2015, pp. 2754-2761

[49] Alexandru Amaricai, Sergiu Nimara, Oana Boncalo, Jiaoyan Chen, Emanuel Popovici, „Probabilistic Gate Level Fault Modeling for Near and Sub-Threshold CMOS Circuits", Proceedings of the 17th Euromicro Conference on Digital System Design (DSD), 2014

[50] FP7-ICT / FET-OPEN – 309129 / i-RISC Project Deliverable D2.1, "Circuit level fault models for sub-powered CMOS circuits for uncorrelated and correlated errors", available on-line at: http://www.i-risc.eu, January 2014

[51] Sergiu Nimara, Alexandru Amaricai, Mircea Popa, "Sub-threshold CMOS circuits reliability assessment using simulated fault injection based on simulator commands," IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics (SACI), 2015

[52] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, „Fault Injection into VHDL Models: The MEFISTO Tool", 24th Annual International Symposium on Fault Tolerant Computing (FTCS-24), pp 66-75, 1994

[53] Mark S. K. Lau, Keck-Voon Ling, Yun-Chung Chu and Arun Bhanu: „A General Mathematical Model of Probabilistic Ripple-Carry Adders", Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010

[54] L. N. Chakrapani, B. E. S. Akgul, S. Cheemalavagu, P. Korkmaz, K. V. Palem and B. Seshasayee: „Ultra-Efficient (Embedded) SOC Architectures based on Probabilistic CMOS (PCMOS) Technology", Proceedings of the conference on Design, Automation and Test in Europe (DATE), 2006

[55] Pong P. Chu, „RTL Hardware Design Using VHDL: Coding for Efficiency, Portability and Scalability", Wiley-Interscience, 2006

[56] Gordon W. Roberts and Adel S. Sedra, "SPICE – Second Edition", Oxford University Press, 1997

[57] Predictive Technology Models, available online at: http://ptm.asu.edu/

[58] Jiaoyan Chen, Christian Spagnol, Satish Grandhi, Emanuel Popovici, Sorin Cotofana, Alexandru Amaricai, "Linear Compositional Delay Model for the Timing Analysis of Sub-Powered Combinational Circuits", Proc. International Symposium on VLSI (ISVLSI), 2014

[59] Sergiu Nimara, Alexandru Amaricai, Oana Boncalo, Mircea Popa, „Probabilistic Saboteur-based Fault Injection Techniques for Low Supply Voltage Interconnects", Proc. 10th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), 2014

[60] FP7-ICT / FET-OPEN – 309129 / i-RISC Project Deliverable D2.2, "Higher abstraction fault models and their simulation methodology", available on-line at: http://www.i-risc.eu, November 2014

[61] D. Boning, S. Nassif, "Models of process variations in device and interconnect", Design of High-Performance Microprocessor Circuits, chapter 06, pp. 98-116

[62] P. A. Thaker, "Register-Transfer Level Fault Modeling and Test Evaluation Technique for VLSI Circuits", 2000

[63] M. Maniatakos, N. Karimi, C. Tirumurti, A. Jas, Y. Makris, "Instruction-Level Impact Comparison of RT- vs. Gate-Level Faults in a Modern Microprocessor Controller", 27th IEEE VLSI Test Symposium, 2009

[64] J. C. Baraza, J. Gracia, S. Blanc, D. Gil, P. Gil, "Enhancement of Fault Injection Techniques Based on the Modification of VHDL code", IEEE

Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 6, 2008

[65] D. Gil, L. J. Saiz, J. Gracia, J. C. Baraza, P. J. Gil, "Injecting Intermittent Faults for the Dependability Validation of Commercial Microcontrollers", IEEE International High Level Design Validation and Test (HLDVT) Workshop, 2008

[66] A. Evans, D. Alexandrescu, E. Costenaro, L. Chen "Hierarchical RTL –Based Combinatorial SER Evaluation" Proc. International On Line Testing Symposium (IOLTS), 2013

[67] M. Sonza Reorda, M. Violante "Fault List Compaction through Static Timing Analysis for Efficient Fault Injection Experiments" Proc. 17th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2002

[68] N. Foutris, M. Kaliorakis, S. Tselonis, D. Gizopoulos "Versatile architecture-level fault injection framework for reliability evaluation: A first report" Proc. 20th Int. On-Line Testing Symp. (IOLTS), 2014

[69] G. B. Hamad, O. Mohamed, Y. Savaria "Probabilistic model checking of single event transient propagation at RTL level" Proc. 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2014

[70] D. V. Kamath, "VLSI Design Flow", Manipal Institute of Technology, 2014

[71] John R. Barry, „Low-density parity check codes", 2001

[72] Sarah J. Johnson, „Introducing low-density parity-check codes", University of Newcastle, School of Electrical and Computer Engineering, 2010

[73] S. M. Yoo, D. Koturi, D. W. Pan, J. Blizard, "An AES crypto-chip using a high-speed parallel pipelined architecture," Microprocessors and Microsystems, vol. 29, issue 7, pp. 317-326, Elsevier, 2005

[74] M. Feldhofer, J. Wolkerstorfer, V. Rijmen, "AES implementation on a grain of sand", IEE Proceedings on Information Security, vol. 152, issue 1, pag. 13-20, 2005

[75] Wishbone bus and AES crypto-chip Verilog designs, available on Open Cores website: http://www.opencores.org

[76] Sergiu Nimara, Alexandru Amaricai, Oana Boncalo, Mircea Popa, "Multi-level simulated fault injection for data-dependent reliability analysis for RTL circuit descriptions," Advances in Electrical and Computer Engineering (AECE) Journal, vol. 16, issue 1, pp. 93-98, 2016, DOI: 10.4316/AECE.2016.01013

[77] R. G. Gallager, "Low-density parity-check codes," IRE Transactions on Information Theory, 1962

[78] D. J. C. MacKay and R.M. Neal, "Good codes based on very sparse matrices," Cryptography and Coding, 5th IMA Conference, 1995

[79] Y. S. Park, D. Blaauw, D. Sylvester, Z. Zhang, "Low-power high-throughput LDPC decoder using non-refresh embedded DRAM", IEEE Journal of Solid State Circuits, vol. 49, issue 3, 2014

[80] A. Balatsoukas-Stimming and A. Dollas, "FPGA-based design and implementation of a multi-Gbps LDPC decoder," 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012

[81] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, Z. Zheng "LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives" Proc. 11th USENIX conference on File and Storage Technologies, 2013

[82] X. Chen, J. Kang, S. Lin, V. Akella, “Memory system optimization for FPGA-based implementation of quasi-cyclic LDPC codes decoders”, IEEE Transactions on circuits and systems, vol. 58, no. 1, 2011

[83] T. Nguyen-Ly, K. Le, F. Ghaffari, A. Amaricai, O. Boncalo, V. Savin, D. Declercq, „FPGA design of high throughput LDPC decoder based on imprecise offset min-sum decoding”, 13th IEEE International New Circuits and Systems Conference (NEWCAS), 2015

[84] Alexandru Amaricai, Nicoleta Cucu-Laurenciu, Oana Boncalo, Joyan Chen, Sergiu Nimara, Valentin Savin, Sorin Cotofana, „Multi-level probabilistic timing error reliability analysis using a circuit dependent fault map generation”, Conference on Design of Circuits and Integrated Systems (DCIS), 2015

[85] Sergiu Nimara, Oana Boncalo, Alexandru Amaricai and Mircea Popa, "FPGA architecture of multi-codeword LDPC decoder with efficient BRAM utilization," 2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Kosice, 2016, pp. 1-4.

[86] C. L. K. Ngassa, V. Savin, E. Dupraz, D. Declercq, “Density evolution and functional threshold for the noisy min-sum decoder”, IEEE Transactions on Communications, vol. 63, issue 5, pp. 1497 – 1509, January 2015

[87] A. Amaricai, V. Savin, O. Boncalo, N. Cucu-Laurenciu, J. Chen, and S. Cotofana, “Timing error analysis of flooded ldpc decoders,” in Microwaves, Communications, Antennas and Electronic Systems (COMCAS), 2015 IEEE International Conference on, Nov 2015, pp. 1–5.

[88] E. Dupraz, D. Declercq, B. Vasi, and V. Savin, “Finite alphabet iterative decoders robust to faulty hardware: Analysis and selection,” in 2014 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Aug 2014, pp. 107–111.