**PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL**
**FACULTY OF INFORMATICS**
**COMPUTER SCIENCE GRADUATE PROGRAM**

# ANALYSIS OF VOLTAGE SCALING EFFECTS IN THE DESIGN OF RESILIENT CIRCUITS

**MATHEUS GIBILUKA**

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Dr. Ney Laert Vilar Calazans

**Porto Alegre**
**2016**

**Dados Internacionais de Catalogação na Publicação (CIP)**

G446a Gibiluka, Matheus
Analysis of voltage scaling effects in the design of resilient circuits / Matheus Gibiluka. – 2016.
103 f.

Dissertação (Mestrado) – Faculdade de Informática, PUCRS.
Orientador: Prof. Dr. Ney Laert Vilar Calazans

1. Informática. 2. Processamento de Imagens. 3. Circuitos Assíncronos. 4. Diagnóstico por Imagem. 5. Arquitetura de Redes. I. Calazans, Ney Laert Vilar. II. Título.

CDD 004.6

**Ficha Catalográfica elaborada por Loiva Duarte Novak – CRB10/2079**

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "*Analysis of Voltage Scaling Effects in the Design of Resilient Circuits*" apresentada por Matheus Gibiluka como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 4 de março de 2016 pela Comissão Examinadora:

Prof. Dr. Ney Laert Vilar Calazans– Orientador — PPGCC/PUCRS

Prof. Dr. Fernando Gehm Moraes – — PPGCC/PUCRS

Dr. Matheus Trevisan Moreira - — Chronos Tech

Prof. Dr. Edson Ifarraguierre Moreno - — FACIN/PUCRS

Homologada em...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

“You can’t connect the dots looking forward;
you can only connect them looking backwards.”
(Steve Jobs)

## ACKNOWLEDGMENTS

To my parents, Liamara and Moacir, for their continued support.

To my advisor, Dr. Ney L. V. Calazans, for the patience and guidance throughout the course of this research.

To Walter Lau Neto, for the assistance in the characterization tasks.

To my friends at GAPH and GSE for the helpful discussions and support throughout the Master's course.

To my girlfriend, Ana, for the love, support and patience.

# ANÁLISE DOS EFEITOS DE ESCALAMENTO DE TENSÃO NO PROJETO DE CIRCUITOS RESILIENTES

## RESUMO

Embora o avanço da tecnologia de semicondutores permita a fabricação de dispositivos com atrasos de propagação reduzidos, potencialmente habilitando o aumento da frequência de operação, as variações em processos de fabricação modernos crescem de forma muito agressiva. Para lidar com este problema, significativas margens de atraso devem ser adicionadas ao período de sinais de relógio, limitando os ganhos em desempenho e a eficiência energética do circuito. Entre as diversas técnicas exploradas nas últimas décadas para amenizar esta dificuldade, três se destacam como relevantes e promissoras, isoladas ou combinadas: a redução da tensão de alimentação, o uso de projeto assíncrono e arquiteturas resilientes. Este trabalho investiga como a redução de tensão de alimentação afeta os atrasos de caminhos em circuitos digitais, e produz três contribuições originais. A primeira é a definição uma técnica para garantir que circuitos sintetizados com um conjunto reduzido de células atinjam resultados comparaveis aos da biblioteca completa, mantendo a sua funcionalidade mesmo quando alimentados por tensões reduzidas. A segunda é a composição de um método para estender o suporte a níveis de tensão de alimentação para bibliotecas de células padrão providas por fabicantes de CIs, através de novas técnicas de caracterização de bibliotecas. A terceira é a análise dos efeitos do escalamento de tensão no projeto de circuitos resilientes, considerando tensões de alimentação superiores e inferiores à tensão de limiar dos transistores.

**Palavras-Chave:** circuitos assíncronos, circuitos resilientes, escalamento de tensão, projeto de circuitos digitais, caracterização de bibliotecas de células padrão.

# ANALYSIS OF VOLTAGE SCALING EFFECTS IN THE DESIGN OF RESILIENT CIRCUITS

## ABSTRACT

Although the advancement of semiconductor technology enable the fabrication of devices with increasingly reduced propagation delay, potentially leading to higher operating frequencies, manufacturing process variability grows very aggressively in modern processes. To cope with growing variability phenomena, significant delay margins need to be added to clock signal's periods, to ensure timing closure, which limits performance gains and constrains power efficiency. Among the several techniques that have been explored in the last decades to address these problems, three are quite relevant and promising either in isolation or combined: voltage scaling, asynchronous circuits and resilient architectures. This work investigates how voltage scaling affects circuit path delays, and produces three sets of original contributions. The first set establishes a technique to ensure that circuits synthesized with a reduced library achieve results comparable to the full library, while keeping functionality at low supply voltages. The second set of contributions composes a method to extend the voltage corners supported by standard cell libraries. This takes place through new library characterization techniques. The third set of contributions provides insights on the effects of voltage scaling in the design of resilient circuits. This analysis evaluates supply voltages in super- and sub-threshold levels.

**Keywords:** asynchronous circuits, resilient architectures, voltage scaling, digital circuit design, standard cell library characterization.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

ALF – Advanced Library Format

ASIC – Application-specific Integrated Circuit

BD – Bundled-data

CSM – Current Source Models

DI – Delay Insensitive

DSM – Deep Submicron

DSTB – Double Sampling With Time Borrowing

DVFS – Dynamic Voltage and Frequency Scaling

EDA – Electronic Design Automation

EDL – Error Detecting Latches

ELC – Encounter Library Characterizer

FF – Flip-flop

HDL – Hardware Description Language

IC – Integrated Circuit

IOT – Internet of Things

IP – Intellectual Property

LC – Library Comparator

MSFF – Master-Slave Flip-Flop

MVC – Multi-Voltage Characterization

NLDM – Non-linear Delay Model

PA – Path Analyzer

$\overline{PRP}$ – Average Percentage of Resilient Paths

PVT – Process, Voltage, and Temperature

PWL – Piecewise Linear

QDI – Quasi-Delay-Insensitive

RC – Resistance and Capacitance

STA – Static Timing Analysis

TDTB – Transition Detector With Time Borrowing

TRW – Timing Resilience Window

# CONTENTS

# 1. INTRODUCTION

The first wave of the mobile revolution led to the ubiquity of battery-powered portable devices. These devices, which are usually based on the synchronous circuit design paradigm, are required to deliver good autonomy while performing complex tasks that demand high performance computing. Wearable devices and Internet of Things (IoT) applications, which, according to [Zod15], are part of the next wave of the mobile revolution, have very different requirements. Unlike most mobile devices, which can be recharged on a daily basis, wearables and battery-powered IoT applications can have a very strict power budget, as these products could be subjected to energy-starved environments. Due to this characteristic, such applications are good candidates for ultra-low power sub-threshold operation [Vit15, CCGC13].

Albeit transistors become faster as technology nodes advance, allowing smaller gate delays and potentially leading to higher operating frequencies, manufacturing process variability also grows aggressively [JCDGH15]. In this way, delay uncertainties, which arise from such variations, begin to challenge synchronous designers. To cope with that, increasingly large delay margins need to be added to the period of clock signals to ensure timing constraints closure, limiting performance gains and constraining power efficiency. Low power requirements tend to further increase variations, which may lead to additional delay guardbands [JCDGH15]. Among the several techniques that have been explored in the last decades to address these problems, three are quite relevant and promising: voltage scaling, asynchronous circuits, and resilient architectures. The former is a classic technique, usually employed to reduce power consumption by diminishing the supply voltage of the circuit. It relies on the fact that power is directly proportional to the square of the voltage and, together with dynamic frequency scaling, is employed in many modern processors [KSSF10, ZGC+12, KKLR11]. Asynchronous circuits remove the need for a global clock signal, potentially reducing power consumption and electromagnetic interferences caused by periodic signal [BOF10]. Resilient architectures allow removing delay margins and tolerate timing errors to mitigate variability, which enables circuits to operate with relaxed timing constraints [EKD+03]. Due to the less strict timing margins, resilient circuits supporting voltage scaling could potentially provide higher performance at nominal voltage by allowing the circuit to operate close to average-case delay. These can also provide larger power savings when operating at supply levels lower than the minimum error-free voltage, which is possible due to the resilience to timing errors.

In the context of mobile devices, wearables, and IoT, resilient architectures supporting voltage scaling could bring benefits to each of these classes of application. The former class can leverage from the performance gains at nominal voltage. The remaining ones can benefit from operating at lower supply voltages – which can increase the energy efficiency

of the circuit. Yet, synchronous resilient architectures are prone for failures due to metastability [BCC+14] and can exhibit high penalties due to error recovery based on architectural replay [JCDGH15]. This issues can be overcome by asynchronous resilient architectures, such as Blade.

This work proposes as novelty to investigate how the reduction of supply voltage affects circuit path delays, providing insight on the effects of voltage scaling in resilient circuits. To enable this investigation, a method to extend the voltage corners supported by standard cell libraries was devised, along with a technique for establishing reduced cells libraries that present performance levels similar to the full libraries while ensuring proper operation at low supply voltages.

## 1.1 Motivation and Objectives

Blade [HMH+15, HHC+15] is a novel asynchronous resilient circuit architecture that seeks to overcome the problems with current synchronous resilient architectures, providing low error-recovery overhead and metastability-tolerant operation. Its development is a partnership between the University of Southern California (USC), in Los Angeles (USA), and the GAPH group at PUCRS. Part of the research work conducted by the Author during his MSc course targeted the development of Blade. Since Blade is a new approach for circuit design, there are many open problems and interesting research areas related to it. In this context, this work aims to understand how digital circuits behave under voltage scaling, providing insight on how reduced supply voltages affect resilient circuit design. Even though this study focuses on Blade, some analysis can be extended to synchronous resilient architectures.

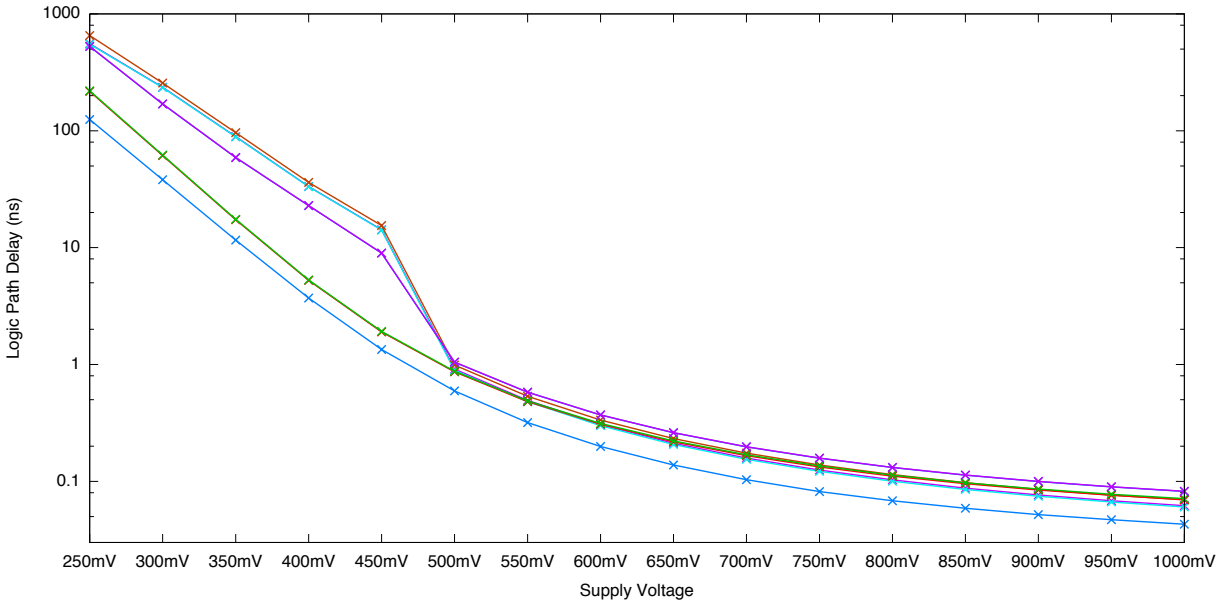



Figure 1.1 – Plot depicting the logic path delays of the c17 ISCAS'85 benchmark [HYH99] as a function of the supply voltage. The maximum delay of the eight paths that compose the circuit are shown.

It is well known that the delays of logic paths in a digital circuit increase as the supply voltage scales down. For instance, consider Figure 1.1, which plots the logic path delays of a circuit as a function of the supply voltage. Each line on the plot depicts the maximum delay of one of the eight logic paths obtained in a post-synthesis netlist of the c17 ISCAS'85 benchmark [HYH99]. In this plot, one can measure the rate at which the circuit delay increases as the supply voltage reduces. In addition, an interesting behaviour of the critical path can be observed: at the threshold region (around 450-500mV), the critical path of the circuit changes. From 1V to 500mV, the critical path is that represented by the purple line. For voltages below 500mV, the path depicted in brown becomes the critical one. Another way to visualize this behaviour is through the *path migration plot*, illustrated in Figure 1.2. This plot depicts the logic path delays of the circuit ranked by their criticality at each supply voltage – higher criticality levels indicate paths with larger delays. The change of critical path can be clearly seen by looking at the behaviour of the path depicted in brown. Its delay gradually increases in relation to the other paths, causing it to become the critical path when the threshold is crossed (i.e. for supply voltages of 450mV and below). The migration plot of this circuit shows that the logical path criticality can change as supply voltage scales. One of the goals of this work is to evaluate how these path fluctuations affect the design of resilient circuits. The data depicted in Figures 1.1 and 1.2 was extracted using the analysis environment detailed in Section 5.2.

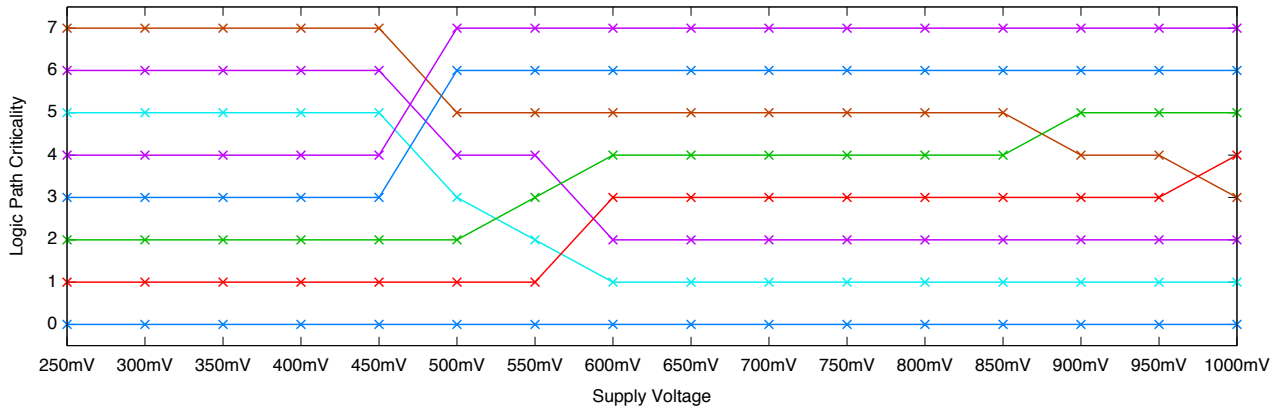



Figure 1.2 – Path migration plot illustrating the logic path fluctuation on the the c17 benchmark [HYH99]. For each supply voltage, the maximum delays of the eight paths that compose the circuit are ranked according their criticality.

To enable circuits to operate with relaxed timing constraints, resilient architectures implement specialized hardware to detect and correct timing violations in a transparent manner. To avoid prohibitively large area overheads, error detection logic is traditionally implemented only in critical and near-critical paths. In this context, the main objective of this work is to understand the impact of logic path fluctuations due to voltage scaling in resilient circuit design.

## 1.2 Contributions

The contributions of this work can be summarized in three items: i) a method to select reduced standard cell libraries; ii) the multi-voltage characterization flow; iii) the timing analysis of digital circuits under voltage scaling. These contributions are briefly sketched in this Section.

Method to select reduced standard cell libraries

This first contribution guarantees that circuits synthesized with the reduced library achieve timing and area results comparable to the full library, while keeping circuit functionality at low/very low supply voltages.

Multi-voltage characterization flow

This flow is a method designed to extend the voltage corners supported by standard cell libraries. It provides a systematic way to characterize libraries to a wide range of target voltages, ensuring the proper scaling of voltage-dependent parameters.

Timing analysis of circuits under voltage scaling

The main contribution of this work is the study of logic path behavior under voltage scaling. This investigation helps understand how supply voltage reduction affects path delays, providing insight on how changes in logic path criticality impacts the design of resilient circuits.

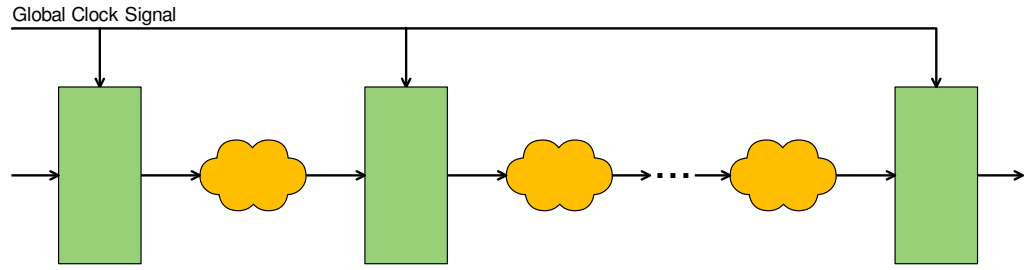
## 1.3 Document Structure

The remaining of the document is organized as follows. Chapter 2 provides relevant background information on asynchronous circuits and resilient architectures. The method used to select the reduced cell library that is used throughout this work is detailed in Chapter 3. Chapter 4 discusses standard cell characterization and describes the multi-voltage characterization flow. The main contribution of this work is explored in Chapter 5. Chapter 6 presents some final remarks and directions for future work.

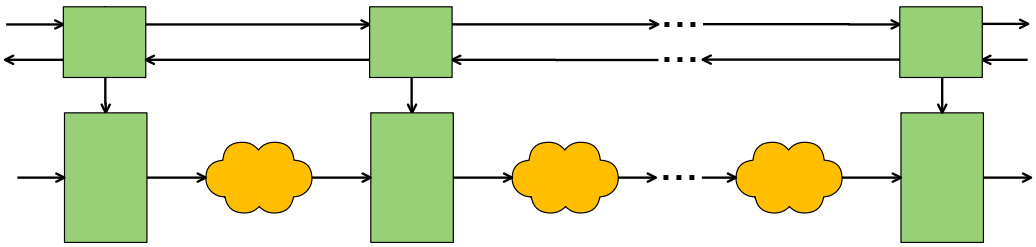
# 2. ASYNCHRONOUS CIRCUITS AND RESILIENT ARCHITECTURES

This Chapter presents basic concepts on asynchronous circuits and resilient architectures necessary to the understanding of this work in Sections 2.1 and 2.2. Section 2.3 presents and discusses the state of the art in synchronous resilient architectures. Section 2.4 introduces Blade, a novel asynchronous resilient architecture that seeks to overcome the problems with current synchronous resilient architectures. Finally, Section 2.5 provides a discussion about the design of resilient circuits.

## 2.1 Asynchronous Circuits




(a) Synchronous Pipeline



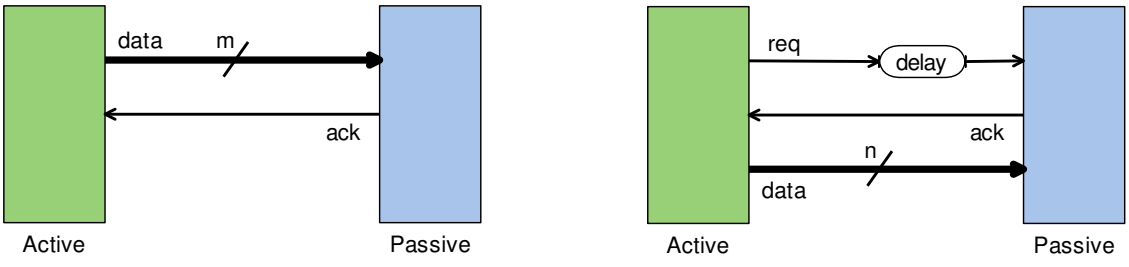
(b) Asynchronous Pipeline

Figure 2.1 – An abstract view of pipeline implementations of digital circuits: (a) a synchronous pipeline; (b) an asynchronous pipeline. Adapted from [NS11].

Events in sequential digital systems are traditionally synchronized by a global clock signal. In such implementations, as Figure 2.1(a) illustrates, the clock signal is designed to update all registers simultaneously – thus, creating a *synchronous* circuit. *Asynchronous* circuits, on the other hand, do not employ global synchronization. Instead, local handshakes between neighbour registers control the synchronization and sequencing of events [BOF10]. In an asynchronous pipeline like the one depicted in Figure 2.1(b), the operation of each register depends only on the previous and following registers – i.e. there is no control signal

commanding all registers. The discrete-time abstraction created by synchronous implementations helps simplifying the design, but removing it can grant several other benefits, like lower power consumption, higher operating speed, lower electromagnetic noise emission, and the elimination of clock distribution problems [Hau95].

Differently from synchronous circuits, asynchronous circuits can be designed using different data encoding schemes and handshake protocols [BOF10]. A choice of handshake protocol and data encoding scheme is called a *design template* or simply a *template*. Current practical asynchronous design templates can be classified in two main families: Quasi-delay-insensitive (QDI) and Bundled-data (BD) [BOF10]. The key characteristic of QDI designs is the use of multi-rail delay insensitive (DI) data encoding. An example is dual-rail, which uses two wires to represent one bit of data – and completion-sensing circuits to determine data validity [BOF10]. The example of Figure 2.2(a) uses the use of an n-of-m delay insensitive code, where each pattern with n bits at '1' in the m wires of the data channel represents a valid information. For example, if m=3 and n=2, the 2-of-3 resulting code can represent 3 different pieces of information (011, 101, 110) in a delay insensitive way. The passive block (on the right hand side) uses special circuitry to sense when the data is one of the valid 3-bit combination, thus inferring a request event from the active block. Even though QDI provides relaxed timing constraints, circuits from this class are usually power and area hungry. For this reason, this work restricts attention to BD circuits, which are implemented with single-rail encoding (i.e. the traditional Boolean encoding, where n bits can represent $2^n$ different pieces of information). This provides lower overhead and is less constraining with regard to the use of conventional standard-cell libraries and EDA tools for synthesis purposes.




(a) Quasi-delay-insensitive (QDI)

(b) Bundled-data (BD)


(c) 2-phase handshake protocol

(d) 4-phase handshake protocol

Figure 2.2 – Families of asynchronous circuits and handshake protocols: a) Quasi-delay-insensitive (QDI) circuit using an n-of-m DI code; b) Bundled-data (BD) circuit; c) 2-phase handshake protocol; d) 4-phase handshake protocol.

BD circuits often use an explicit pair of sideband control signals (request and acknowledge) to provide synchronization. Since single-rail data encoding is used, BD requires a number of timing assumptions to ensure proper circuit operation. These timing assumptions lead to constraints to guarantee that combinational circuits have enough time to compute, while respecting setup and hold times of registers. Ensuring constraints usually happens by adding delay lines in front of control signals to match the logic path delay. This has as goal to guarantee that handshake events only take place when data signals are stable at the input of the next stage's register(s). As an example, refer to Figure 2.2(b): the delay line inserted in the *req* signal ensures that the passive block will only receive the request event after the signals at its data input become stable. BD designs exist using both 2- and 4-phase handshake protocols, which are illustrated, respectively, in Figures 2.2(c) and 2.2(d). 4-phase protocols have a return to zero (RTZ) (or to one, RTO) phase (transitions 3 and 4 in Figure 2.2(d) for an RTZ protocol) that increases the number of events necessary to complete a handshake, potentially reducing the performance of the circuit. 2-phase protocols improve performance, by eliminating the return phase, at the expense of more complex control circuitry. BD templates can use different design styles for implementing their control blocks. 2-phase BD control circuits were pioneered by Ivan Sutherland, who introduced the concept of micropipelines [Sut89], implemented with special capture-pass latches capable of sensing transitions at its inputs. A more recent approach is Mousetrap [SN07], which implements 2-phase BD circuits using level-sensitive latches and XOR gates. Nowick and Singh propose a very good overview of these and other classical and modern asynchronous pipelines in [NS11].

## 2.2 Resilient Architectures

As silicon technology advances into the deep submicron (DSM) range, manufacturing variability becomes an increasing concern to circuit designers. To ensure good yield on contemporary processes, circuit designers add delay margins to account for process, voltage, and temperature (PVT) variations. Figure 2.3 illustrates how these margins, along with clock-related and data-dependent margins can increase the minimum cycle time of clocked logic. As an alternative to cope with PVT variations on synchronous circuits, resilient design techniques help removing the increasingly large delay margins, allowing circuits to operate with relaxed timing constraints. These techniques allow timing violations to occur, and rely on dynamic error recovery mechanisms to correct errors derived from such violations. Resilient architectures are usually implemented with special error detecting registers, capable of flagging when timing violations occur, and architectural mechanisms for error recovery, such as pipeline stalls or architectural replay. A number of resilient architectures have been proposed to date [EKD+03, DTP+09, FFK+13, KKFK13, HMH+15].

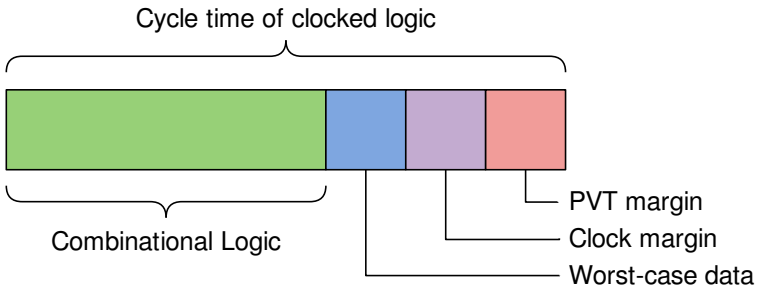



Figure 2.3 – Timing margins in contemporary VLSI designs.

Razor [EKD+03], also called Razor I, is a first successful family of resilient architectures and serves as a good case to explore how such architectures work. Figure 2.4(a) depicts the Razor I basic implementation, which relies on a special register called Razor Flip-flop (FF). This FF replaces selected FFs on delay-critical paths from an original, non-resilient architecture. A Razor FF combines a regular FF with a shadow latch controlled by a delayed clock signal. The delayed clock clk_del is designed to meet the latch setup time in worst-case situations, guaranteeing data validity at the latch input at any clk_del rising edge, even if and when the main flip-flop timing is violated. Timing violations are detected by comparing the values stored in the main FF and in the shadow latch. When delay failures occur, the value stored in the latch is used to correct the violation.

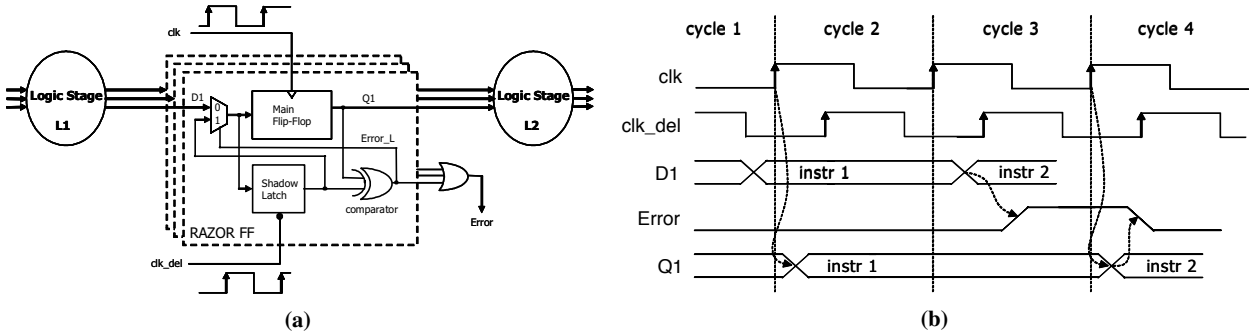



Figure 2.4 – Overview of the Razor I Architecture: (a) pipeline with Razor latches and control lines; (b) operation of a Razor I circuit. Adapted from [EKD+03].

Figure 2.4(b) illustrates the operation of a rising edge sensitive Razor FF. In the first clock cycle, the timing requirements of the main FF are met, and correct values are stored in both the main FF and in the shadow latch. In this scenario, the error signal coming from the comparator remains low, and the circuit operates without interruptions. In cycle 2, the combinational path time exceeds the allotted time and an incorrect value is stored in the main FF. Once the correct value is sampled in the shadow latch, at the start of cycle 3, the error signal is asserted. This causes the value stored in the shadow latch to be used as the input to the main FF. At the beginning of cycle 4, the correct value is stored in the main FF. Once a timing violation occurs in some clock cycle, the data on the following pipeline stage must be flushed in the next clock cycle, to avoid the propagation of incorrect data – this

accounts for the one cycle recovery penalty. In addition, the previous pipeline stage must be stalled for one cycle while the data from the shadow latch is restored into the main FF. Razor I operation relies on the assumption that worst-case situations are data-dependent and rarely occur. By allowing the circuit to tolerate timing failures, it is possible to reduce the circuit design margins, improving the average-case performance.

The error rate of a resilient circuit is a key design parameter, as it can determine if the circuit will deliver the desired performance improvements or if the error recovery overheads will dominate. The optimum error-rate is application- and circuit-dependent and can be set at design time, by post-silicon tuning, or during operation, using Dynamic Voltage and Frequency Scaling (DVFS) techniques.

## 2.3 State of the Art on Synchronous Resilient Architectures

This Section presents an overview of the current literature on synchronous resilient circuits, covering Razor-based architectures and alternative approaches.

### 2.3.1 Razor I

The Razor I architecture [EKD+03] was originally proposed as a technique to enable larger energy savings on circuits operating with dynamic voltage scaling (DVS). As mentioned in Section 2.2, the error rate is a key design parameter of a resilient circuit, specially considering the large error correction cost in a Razor I circuit, which can be about 18 times more expensive in terms of energy than regular operation [EKD+03], as in the case of a 64-bit Kogge-Stone adder. Based on experiments, a target error rate of 1.5% was selected for that circuit, allowing average energy savings of 41% with a maximum performance slowdown of 6% for the set of simulated benchmarks. A 64-bit Alpha processor employing the Razor I error detection and recovery mechanism was manufactured in 0.18 µm technology. The circuit was designed to operate at 200MHz, and the shadow latch clock was delayed by half clock cycle. Out of a total of 2408 flip-flops, 198 timing-critical flops were replaced with Razor FFs. The prototype presented an energy overhead of up to 3.1% due to the added logic when operating at nominal voltage.

### 2.3.2 Razor II

Razor II [DTP+09] simplifies the FF design proposed on the original Razor by removing the error recovery mechanism. The motivation behind this approach is that the

error rate at the point of first failure is in the order of 1 error in 10 million cycles, which makes error-correction energy negligible. Therefore, Razor II FF performs only error detection and error recovery takes place through architectural replay. The new FF, depicted in Figure 2.5(a), uses a single latch combined with a transition detector, operating as a positive-edge-triggered FF. Figure 2.5(b) exemplifies the Razor II FF operation: *i)* when data stabilizes before the rising edge of the clock no error is flagged, as illustrated by the scenario on the left side of Figure 2.5(b); *ii)* if a transition occurs after the rising edge of the clock, while the latch is transparent, the FF detects the transition and asserts the error signal, as shown on the right side of Figure 2.5(b). When an error occurs, the whole pipeline is flushed and the failing instruction is re-executed. In case of successive failures, the clock frequency is reduced by half during 8 cycles. A 64-bit 7-stage Alpha processor featuring Razor II was manufactured in 0.13 µm technology. Energy savings of up to 37.4% were obtained when the processor was kept with an error rate of 0.04%, compared to the energy consumption when the supply voltage is set to ensure correct operation with a 10% margin.

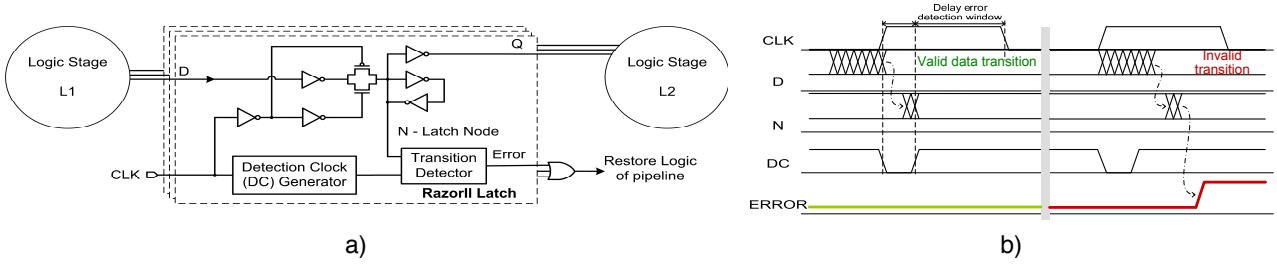



Figure 2.5 – Razor II operation: a) Example of a Razor II pipeline section and details of the Razor II flip-flop; b) Waveform exemplifying the operation of a Razor II circuit. Extracted from [DTP+09].

### 2.3.3 Razor-Lite

Razor-Lite [KKFK13] uses a side-channel transition detection approach, compatible with standard D FFs, as a way to reduce overheads due to error detection mechanisms. This approach allows the implementation of resilience on well-balanced pipelines, where a large number of FFs must be capable of detecting timing violations, with reasonable overhead. Figure 2.6(a) illustrates a conventional flip-flop design and the added logic (inside the red box) required by Razor-Lite. The added circuit connects to the virtual VDD (VVDD) and VSS (VVSS) rails of the FF, acting as a transition detector. Under normal operation (i.e. no error), as illustrated in the left box of Figure 2.6(b), the virtual rails voltage is kept stable after the rise transition of the clock and no error is flagged. When, after the rising edge of the clock, the input data '*D*' transitions to '0', the following takes place, as depicted in the center box of Figure 2.6(b): *i)* before the data transition, 'VVDD' is charged to VDD, 'VVSS' is grounded,

and the node 'DN' is also grounded; *ii)* once 'D' transitions to '0', 'VVDD' is discharged through the node 'DN', which is kept in '0' by the feedback inverter; *iii)* the transition on 'VVDD' is detected and flagged as an error; *iv)* on the falling edge of the clock, the virtual rails are restored to their original state, and the error signal is deasserted. The detection of zero-to-one transitions on the signal 'D' is analogous, as illustrated in the rightmost box of Figure 2.6(b). A 7-stage 64-bit Alpha processor was prototyped in 45nm SOI CMOS technology using the Razor-Lite error detecting mechanism. All registers in the decoding and execution stages of the pipeline (492 registers out of 2482), which are in the processor's critical path, were replaced by Razor-Lite registers, generating a core area overhead of 4.42% and a power overhead of 0.3%. Peak energy efficiency is improved by 83% when compared to the margined baseline processor.

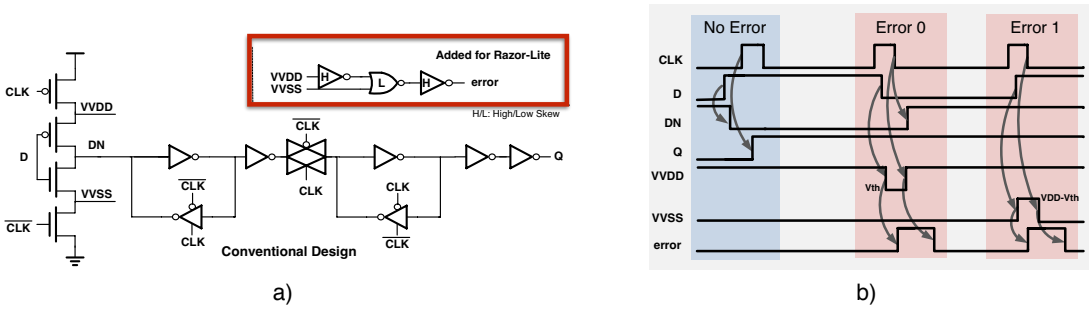


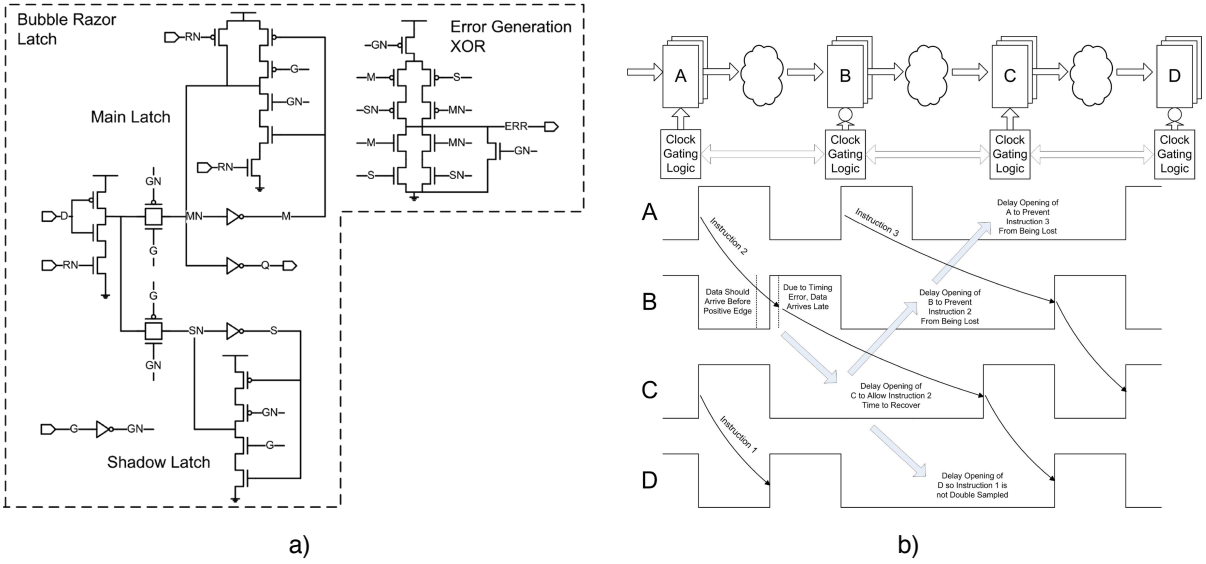

Figure 2.6 – a) Example of a Razor-Lite error detector; b) Waveform exemplifying the operation of a Razor-Lite error detection circuit. Extracted from [KKFK13].

### 2.3.4 Bubble Razor

Bubble Razor [FFK+13] is an architecture-independent error detection and correction mechanism that relies on a two-phase clock, and on a latch-based datapath. The error detection circuit, depicted in Figure 2.7(a), is similar to the one used on the original Razor. It is based on a shadow latch that captures the data before the main latch opens and flags errors if the value at the main register changes after it becomes transparent. Error correction is accomplished by local stalling. When a timing violation is detected, error signals (bubbles) are propagated to neighboring latches. It is important to notice that errors do not immediately corrupt the circuit, as they borrow time from later pipeline stages. A bubble gives an additional cycle for the correct data to arrive at the next latch. This is achieved by making adjacent latches skip the next transparent clock phase. Figure 2.7(b) illustrates how timing errors are corrected by propagating bubbles: *i)* a timing violation takes place at latch B; *ii)* the Bubble Razor latch detects the error and sends a bubble to latch C, making it skip the next cycle, which allows additional time to correct the violation; *iii)* latch C back propagates a bubble to latch B, to prevent it from overwriting instruction 2 before it is stored in the former latch; *iv)* in parallel with *iii*, latch C propagates a bubble forward to latch D, preventing it

from double sampling the data currently latched in C; *v)* latch B back propagates a bubble to latch A to prevent it from losing its contents. A bubble propagation algorithm is proposed to avoid indefinite bubble propagation through loops. An ARM Cortex-M3 microcontroller using Bubble Razor was implemented with target at a 45nm SOI CMOS technology. The flip-flop-based design was converted to a latch implementation using commercial retiming tools and error checking was added to all latches, resulting in 87% area overhead.

When operating with reduced margins, this implementation enables 100% throughput increase at nominal voltage and increased clock frequency, or 60% energy savings when compared to a processor operating with worst-case timing margins.




a) b)

Figure 2.7 – Details of the Bubble Razor approach: a) The design of the Bubble Razor latch; b) Timing diagram exemplifying the bubble propagation algorithm employed in Bubble Razor for error recovery. Extracted from [FFK+13].

### 2.3.5 TDTB and DSTB

Bowman et al. [BTK+09] propose two error-detecting registers and a instruction-replay-based error recovery mechanism. The Transition Detector with Time Borrowing (TDTB), illustrated in Figure 2.8(a), is a latch-based register that detects input transitions during the high phase of the clock. Each data input transition generates a pulse at the XOR gate output. The output of the XOR gate and the clock feed a 9-transistor dynamic (in fact, pseudo-static) gate that produces the ERROR signal. This circuit is in fact an asymmetric C-element, a well-known sequential component of asynchronous designs [MHBC15]. As illustrated on the left side of Figure 2.8(b), pulses that take place during the low phase of the clock, while the dynamic gate pre-charges, do not affect the error signal. If the input data arrives late, during the high phase of the clock, the pulse will discharge the output node of the dynamic gate,

causing a transition on the error signal. Once the clock signal transitions to low, the dynamic gate output restores its original state and the error signal is deasserted. Latch-based registers eliminate the risk of metastability in the data path, as the circuit is designed to guarantee that data signals will be stable before the falling transition of the clock. However, there are risks that the error signal (a control signal) becomes metastable. This may happen if the input data transitions slightly before the rising edge of the clock signal.

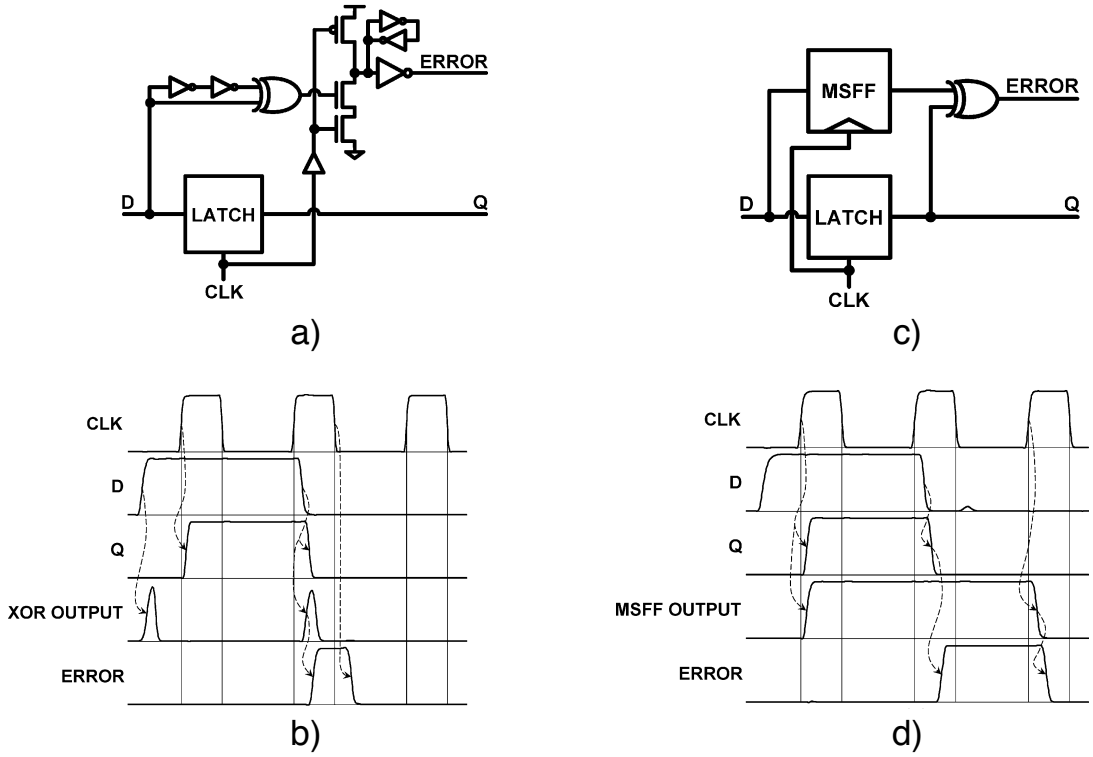



Figure 2.8 – Two options of error detector register architectures and their operation: a) Transition Detector with Time Borrowing (TDTB) circuit; b) Operation of TDTB; c) Double Sampling with Time Borrowing (DSTB)circuit; d) Operation of DSTB. Extracted from [BTK+09].

The second proposed register replaces the transition detector of the TDTB with a shadow flip-flop. The circuit design, called Double Sampling with Time Borrowing (DSTB), is depicted in Figure 2.8(c) and its operation is illustrated in Figure 2.8(d). On the rising edge of the clock, the latch becomes transparent and the input data is sampled by the Master-Slave Flip-Flop (MSFF). DSTB compares the data path latch and the MSFF output to generate the error signal – if the values differ, the input data arrived late and, thus, an error was detected. Similar to TDTB, this approach eliminates data path metastability, but introduces the risk of metastability on the error signal.

The error recovery mechanism proposed by Bowman and others is based on instruction replay. Once a timing violation is detected, the input buffer control logic invalidates the erroneous data. The controller determines the appropriate instruction to be replayed, based on the pipeline stage where the violation was detected. In parallel, the clock fre-

quency is halved, to ensure correct operation during replay. Once the replay is completed, the clock is scaled back to its original frequency.

A test chip using this technology was manufactured in a 65nm technology. Post manufacturing measurements indicate throughput gains of up to 32% when operating at nominal supply voltage and up to 37% reduction in power consumption when reducing VDD while maintaining the nominal throughput. However, due to the possibility of error signals becoming metastable, there is a risk of timing violations not being detected by the controller, compromising circuit operation.

### 2.3.6 Discussion

Section 2.3 presented some of the main proposed synchronous resilient architectures. Others exist, but their discussion is considered outside the scope of this work. In general, these approaches provide good performance or energy efficiency improvements by allowing the circuit to operate on clock periods smaller than the critical path or on supply values below the minimum error-free voltage. These techniques, however, are either prone to failure due to metastability [BCC+14] or exhibit high error recovery penalties. Timing violation recovery on the original Razor and Bubble-Razor is based on a one-cycle recovery mechanism that does not protect the main flop from injecting metastable signals in the data path. The drawback is that the propagation of such signals through the data path has unpredictable effects on the circuit behavior. Moreover, it is well-known [BCC+14] that the resolution time for metastability is unpredictable and unbounded – therefore, no safe assumptions can be made about the time required for a metastable signal to settle, even though statistical bounds exist. For this reason, there is a probability of the original Razor and Bubble-Razor registers to become metastable, compromising the integrity of the data flowing through the data path. Razor II and Razor-Lite FF provide only error detection mechanisms, as violation recovery is based on architectural replay. The problem with these architectures is that when an error is flagged, the pipeline is flushed and all instructions that were in the pipeline need to be re-executed. The approach presents a high error-recovery cost and is often based on extra architecture-dependent circuitry, which requires deep knowledge about the circuit architecture to be implemented. Both TDTB and DSTB have the advantage of restricting the occurrence of metastable events to the control path, where it is more easily dealt with. However, both TDTB and DSTB present the risk of not detecting an error if the register's error signal becomes metastable.

Table 2.1 summarizes the error detection and correction characteristics and issues of the synchronous resilient architectures previously discussed.

Table 2.1 – Summary of the state of the art in synchronous resilient architectures.

| | Error Detection | Error Correction | Issues |
|---|---|---|---|
| **Razor [EKD+03]** | Comparison with shadow latch | Allocation of extra cycle to correct error | Data-path metastability |
| **Razor II [DTP+09]** | Transition detection | Architectural replay | Error-recovery cost; Architecture-dependent error correction |
| **Razor-Lite [KKFK13]** | Transition detection | Architectural replay | Error-recovery cost; Architecture-dependent error correction |
| **Bubble Razor [FFK+13]** | Comparison with shadow latch | Allocation of extra cycles to correct error | Relies on metastability settling in, at most, one cycle |
| **TDTB [BTK+09]** | Transition detection | Architectural replay | Risk of not detecting errors under specific scenarios |
| **DSTB [BTK+09]** | Comparison with shadow flip-flop | Architectural replay | Risk of not detecting errors under specific scenarios |

## 2.4 Blade

The previous Section discussed a number of synchronous timing resilient architectures. It was pointed out that many of these are prone to failure due to metastability [BCC+14] or exhibit high recovery penalties. Approaches relying on architectural replay are often based on extra architecture-dependent circuitry to recover from timing violations, which requires deep knowledge about the circuit architecture to implement resilience and may present high performance penalties when recovering from violations. Resilient circuits are designed to work on clock periods smaller than the worst case ones, allowing timing violations to sporadically occur. When violations do take place, there is a probability of registers becoming metastable in either or both control and data paths. Also, in most of the reviewed approaches this can compromise the integrity of data or the very functionality of the circuit. According to [BCC+14], none of the architectures proposed to date describes a formal timing analysis procedure that can be used to guarantee timing correctness in the presence of metastability.

Blade [HMH+15] is a new asynchronous design style for 2-phase BD circuits that helps overcoming the problems with current resilient architectures. Its development is a partnership between the University of Southern California (USC), in the United States, and the GAPH group at PUCRS. Figure 2.9 illustrates a basic Blade architecture pipeline stage, which features single-rail logic followed by TDTB-based error detecting latches (EDLs) and two reconfigurable delay lines.

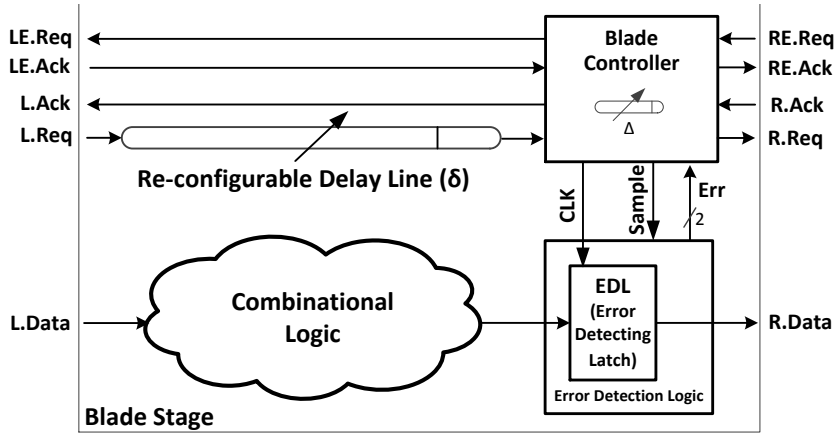



Figure 2.9 – The Blade architecture illustrated through the typical Blade stage organization. Extracted from [HMH+15].

In Blade, the first delay line, of length $\delta$, controls when the EDL first samples the data coming from the combinational logic block, assuming no errors occurred in the previous stage. The second delay line, of length $\Delta$, defines the timing resilience window (TRW), which is a window of time where errors are allowed to happen. If the output of the combinational logic changes during the TRW, the EDL asserts the dual-rail *Err* signal to flag the timing violation. Each Blade Controller (one per stage, typical in asynchronous handshake logic) has four asynchronous control channels [BOF10] that operate with 2-phase handshake protocols. Channels L and R are respectively input and output BD push channels[1], used to transfer data between registers. The request signal on these channels is associated with the $\delta$ delay line. Channels LE and RE on the other hand, are respectively input and output control-only channels (no data transfer is involved), used to check if a timing violation occurred in the previous stage.

Blade employs a novel asynchronous *speculative handshaking* protocol to communicate with neighbor controllers, reducing the timing violation recovery overhead when compared to synchronous resilience approaches discussed in Section 2.3. The request signal used to transfer data between registers (*L.req*) is *speculatively* asserted assuming the $\delta$ delay is sufficiently long and no timing violations occurs. The secondary *extend* channel (*LE*) is used to verify if the assumption was wrong and a timing violation on the previous stage was detected. This allows the previous stage to control how long the current stage will need to wait for a valid data input, in case of timing violations. When no errors occur, as illustrated in Figure 2.10(a), the acknowledgement of the extend channel (*LE.ack*) occurs immediately after *LE.req*. However, when a timing violation takes place on the previous stage, the acknowledgment will be delayed by $\Delta$, as Figure 2.10(b) shows, allowing enough time for the correct data to propagate through the datapath.

[1] A *push channel* is one where the producer generates data and a request control signal and accepts an acknowledge control signal from the consumer side. A *pull channel* is one where the consumer generates a request control signal and accepts data, and where an acknowledge control signal from the producer side. Push channels are data-driven, while pull channels are demand-driven [BOF10].

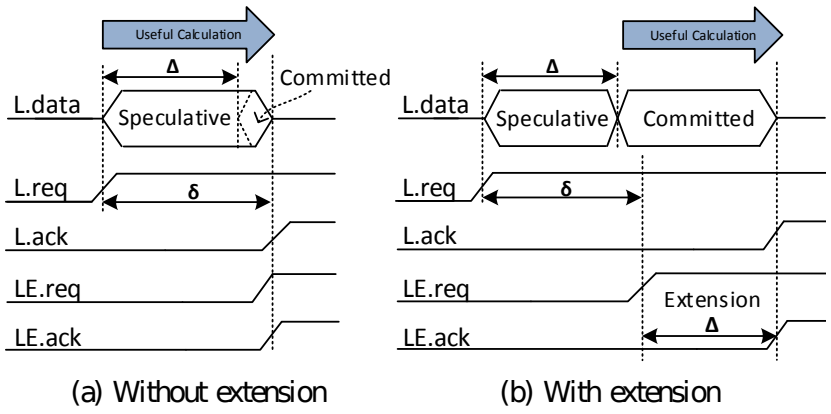



Figure 2.10 – The speculative handshaking protocol. Extracted from [HMH+15].

The speculative handshake controls the opening of the stage latch and the assertion of the output request signal (*R.Req*). Figure 2.11 illustrates the latch control signal over time when a timing violation occurs. In this example, the timing violation was identified at the falling edge of stage 2, delaying the opening of the latch on stage 3 by Δ. Note that the sum of the values of $\delta$ and Δ is always designed to be sufficiently large to cover the longest critical path in the stage.

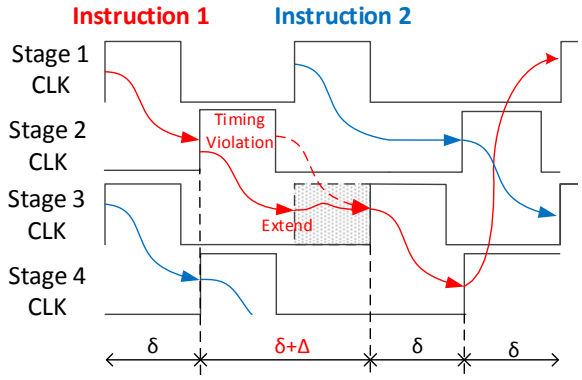



Figure 2.11 – Timing diagram of Blade showing the behaviour when a timing violation takes place. Extracted from [HMH+15].

Similar to Bubble Razor, Blade uses EDLs to flag errors if data signals are not valid when the latch becomes transparent. The circuit can recover from timing violations as long as the data signal becomes valid before the latch closes – this amount of time is determined by the TRW (Δ). To avoid the introduction of metastability on the datapath, the Blade EDL design [MHBC15], depicted by Figure 2.12, is based on the TDTB latches [BTK+09] discussed in the previous Section.

The EDLs avoid that possible metastable signals be propagated to the data path, constraining metastability to occur only in the control path, where it can be more easily treated. In Blade specifically, metastability is moved to the error detection control signals. There, a QFlop [RMCF88], which contains an internal metastability filter, generates the dual-rail encoded *Err* signal by sampling the error detector control signals. The internal metasta-

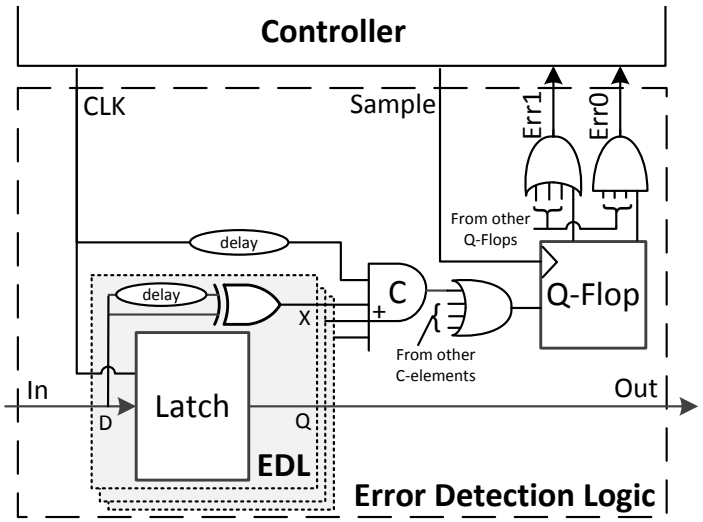



Figure 2.12 – General structure of the error detection logic. Extracted from [HMH+15].

bility filter guarantees that the dual-rail output remains with a spacer identifier [2] until metastability has resolved, i.e. metastability is contained in the QFlop and no metastable signal can propagate. In the rare cases where the metastability takes a long time to resolve, the Blade controller gracefully waits and stalls the opening of the next stage's latch, ensuring correct operation.

Blade includes a design flow that automates the generation of a Blade circuit from a synchronous RTL design. To evaluate the template, a 3-stage version of Plasma [Pla14], a MIPS OpenCore CPU, was converted from a 666MHz synchronous design to a Blade with a TRW of 30% targeting the ST-Microelectronics 28nm FDSOI technology. The overall area overhead of the Blade version is 8.4% when compared to the original synchronous design. The performance of a Blade circuit depends on which logic paths are excited during execution. If only non-critical paths are stimulated, the circuit can operate faster as no timing violations will take place. The Blade version of Plasma presented an average performance equivalent to a synchronous circuit operating on a frequency of 793MHz with a peak frequency of 950MHz, an increase of 19% and 42%, respectively.

## 2.5 Design of Resilient Circuits

This Section discusses design aspects of resilient architectures. The goal is to introduce the design space, providing insight on how design choices affect circuit performance.

The error rate of a resilient architecture is a key design parameter, as it can determine if the circuit will deliver the desired performance improvements or if the error recovery

[2] In a dual rail code, one bit is encoded in two wires. When these are at "01" this indicates a logic 0, "10" indicates a logic 1, and "00" indicate absence of data, also called a spacer.

overheads will dominate. The optimal error-rate is application- and circuit-dependent. In the case of the original Razor [EKD+03], a set of benchmarks were simulated to determine the error rate that resulted in the minimum energy consumption while still meeting the performance constraints. For Blade designs, the authors of [HHC+15] propose an stochastic approach to determine the optimal error rate based on the distribution of data-path delays of the circuit. The error rate of a resilient circuit can be set at design time, by post-silicon tuning, or during operation, using Dynamic Voltage and Frequency Scaling (DVFS) techniques.

The probability of errors occurring on a Blade circuit depends on the size of the timing resiliency window (TRW) [HHC+15]. As previously mentioned in Section 2.4, the operation of a Blade circuit is controlled by two delay lines: $\Delta$ and $\delta$. The length of $\Delta$ determines the TRW, which is the window of time where errors are allowed to happen. A Blade design can only operate correctly if, considering $d$ as the largest path delay in a circuit, the inequality $d \leq \delta + \Delta$ is satisfied. Therefore, there is a trade-off in selecting the length of the delay lines. Larger TRWs ($\Delta$) allow the system to operate faster when no timing errors occur. Shorter $\delta$, however, can result in more transitions being flagged as errors, which forces subsequent pipeline stages to be delayed by $\Delta$, reducing performance gains. Given the close relationship between TRW and the error rate of a resilient circuit, the methodology proposed in [HHC+15] to determine the ideal error rate can also be used to define the optimal TRW to achieve the largest performance gains.

Consider, for instance, a Blade circuit designed to fulfill the timing relationship $\Delta + \delta = d$, where $d$ is the worst-case path delay. In this circuit, all paths with worst-case delay greater than $\delta$ are susceptible to timing violations, and therefore, need to use error detecting latches (EDLs). In a scenario like this, where increasing $\Delta$ reduces $\delta$, the size of the TRW has a direct impact on the area overheads related to error detection hardware. In other words, a larger $\Delta$ leads to a greater number of paths inside the TRW (i.e. paths prone to timing violations), which results in a circuit with more EDLs. There is, therefore, a three-dimensional design space relationship between error rate, TRW, and area overhead.

The concept of a TRW and its association to area overhead can be extended to other resilient architectures as well. In the design of Razor circuits, for instance, only flip-flops in timing-critical paths are replaced by Razor FFs. A path is considered timing-critical if its delay surpasses a given threshold. The threshold, which is selected based on the target error rate of the circuit, delimits the TRW. The remaining of this work uses *TRW* as a general term to refer to the timing window where errors can gracefully take place in a resilient circuit.

# 3. SELECTION OF A REDUCED CELL LIBRARY

The main contribution of this work is the evaluation of circuits under voltage scaling. This analysis, which is detailed in Chapter 5, helps understand how supply voltage reduction affects critical and near critical paths, providing insight on how to design resilient circuits capable of operating under a wide range of supply voltages. Since the study relies on timing information collected through STA, the standard cell library employed on the analysis must be characterized for the range of supply voltages target of the evaluation. Most design kits, however, only provide libraries characterized for a few voltage corners – usually ranging from nominal voltage to 10-20% above and below it. Therefore, to enable a thorough analysis that encompasses near/sub-threshold voltages, a characterization effort to extend the voltage corners supported by the cell library is required.

Contemporary standard cell libraries contain hundreds of gates, many of which are not widely used on traditional synthesis flows – one example are scan flops, which are mainly used to replace regular flip-flops when assembling scan chains. In addition, process variation combined with low supply voltages can deeply degrade the noise margins of some standard cells, rendering them unsuitable for near/sub-threshold operation [KRV+08]. In this context, to avoid a massive characterization work that includes cells that are not commonly employed or that cannot operate correctly at low voltages, we propose the use of a reduced cell library, composed of a subset of the standard cell library. The reduced library was characterized using the flow proposed in Chapter 4, and the results were used to drive the analysis of circuits under voltage scaling. The first contribution of this work, which is detailed in this Chapter, is a method to select a subset of cells that is capable of achieving performance results comparable to the full library while keeping the overheads low and maintaining the circuit functional at low supply voltages.

Section 3.1 presents the criteria used to define a reduced cell library and metrics to compare its performance. The method developed to select and validate the reduced cell library is described in Section 3.2. Finally, Section 3.3 details the resulting library, which is used throughout the remaining of this work.

## 3.1 Reduced Cell Library Criteria and Evaluation Metrics

The goal of the reduced cell library is to achieve synthesis results comparable to the full library while ensuring the circuit remains functional even at low supply voltages. Therefore, a good reduced cell library must meet the following criteria: *i)* low overhead when compared to circuits synthesized with the full library; *ii)* every cell in the library must work properly on all the target supply voltages.

The metric used to evaluate criterion *i* is the area overhead between circuits synthesized with the reduced and full libraries, considering identical timing constraints for both. An automated synthesis environment was designed to allow the analysis of a large set of circuits – Section 3.1.1 details the 58 employed benchmark circuits. Synthesis was performed with the Synopsys Design Compiler using Topographical mode to achieve realistic wire delay estimates. The work targeted the ST-Microelectronics 28nm FDSOI technology using only cells from the ST CORE standard cell library for this technology – therefore, from now on, the set of all CORE library cells is referred as the *full library*. The target clock frequency for each circuit was set as the minimum frequency that ensures a slack near zero, so that cells would not be overused nor underused. Figure 3.1 illustrates the automated synthesis environment used to compare the overheads between the reduced libraries and the full library. The Analyzer is a Python script designed by the Author to compare a batch of synthesis results. The set of benchmark circuits is automatically synthesized for all reduced library candidates, and a comparison with the full library, performed by the Analyzer tool, is stored in the report file. This file contains area and timing results for each synthesis, along with a table summarizing each library candidate's average area overhead and number of slack violations.

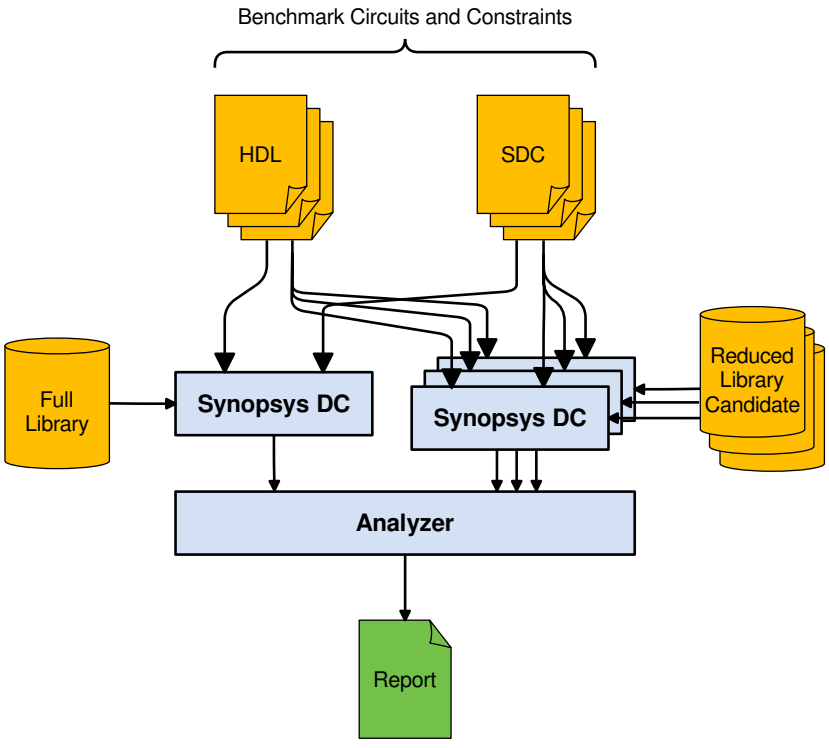



Figure 3.1 – Automated synthesis environment used to compare performance results between the full library and the reduced library candidates. The batch of benchmark circuits is synthesized targeting each library. The Analyzer compares area and timing results, and generates a Report file with the results.

Criterion *ii* relates to the electrical characteristics of each cell – and as such, it is dependent on the silicon technology and must be verified through electrical simulation. The proper operation of a standard cell can be verified by checking the correctness of its

output when subjected to an input vector that stimulates all possible logic states of the gate, considering each target supply voltage. During the cell electrical characterization process, exactly this same process takes place to build the cell delay models. For this reason, the final verification of criterion *ii* is left as part of the characterization task, which is detailed in Chapter 4.

### 3.1.1 Benchmark Circuits

With the goal of ensuring a good representativity of circuits and functions, a set of 58 circuits were selected to be used as benchmarks for the evaluation of reduced cell library candidates. This set includes 4 cryptographic circuits, 15 combinational circuits from the ISCAS'85 benchmark [BF85], 38 sequential circuits from the ISCAS'89 benchmark [BBK89], and a network-on-chip router [MHH+14]. The combinational and sequential ISCAS benchmarks are standard case studies in VLSI research. They contain test cases that stress a variety of functional characteristics of digital design, providing circuits with a large range of logic depth and width implementing unate and non-unate functions. Cryptography circuits also provide test circuits that stress different logic requirements. For instance, some algorithms are based on multiplication operations, such as the RSA, while others rely on look-up tables, like the DES and TDES. NoC routers is an example of applications that rely heavily on data-path and control logic. Table 3.1 shows the details about the set benchmark circuits selected. Finally, due to the computational complexity of the procedures executed in this work, larges benchmarks like 32- or 64-bit general purpose processors revealed to be intractable for the current technology of available computers. They were accordingly discarded in this work.

Table 3.1 – Description of the circuits used as benchmarks for evaluation of reduced library candidates. The number of cells was computed considering the results of the synthesis with the full library. The acronyms SEC/DED stand for “single-error-correcting" and "double-error detecting", respectively.

| Circuit | Frequency (GHz) | # of cells | Description |
|---|---|---|---|
| c17 | 11.10 | 3 | 6-NAND gate circuit [HYH99] |
| c432 | 2.13 | 112 | 27-channel Interrupt Controller [HYH99] |
| c499 | 2.00 | 201 | 32-bit SEC [HYH99] |
| c880a | 2.00 | 178 | 8-bit ALU [HYH99] |
| c1355 | 2.00 | 196 | 32-bit SEC [HYH99] |
| c1908 | 2.00 | 210 | 16-bit SEC/DED [HYH99] |

Continuation of Table 3.1

| Circuit | Frequency (GHz) | # of cells | Description |
|---|---|---|---|
| c1908a | 2.00 | 164 | 16-bit SEC/DED [HYH99] |
| c2670 | 2.00 | 304 | 12-bit ALU and Controler [HYH99] |
| c2670a | 2.00 | 281 | 12-bit ALU and Controler [HYH99] |
| c3540 | 2.00 | 599 | 8-bit ALU [HYH99] |
| c3540a | 2.00 | 477 | 8-bit ALU [HYH99] |
| c5315 | 2.00 | 639 | 9-bit ALU [HYH99] |
| c5315a | 2.00 | 638 | 9-bit ALU [HYH99] |
| c6288 | 1.18 | 3105 | 16-bit Multiplier [HYH99] |
| c7552 | 2.04 | 845 | 32-bit Adder/Comparator [HYH99] |
| des | 2.00 | 17378 | 16-stage Pipeline DES Criptography Circuit |
| des56 | 2.00 | 1763 | Single-stage DES Criptography Circuit |
| rsa | 2.00 | 3719 | RSA Criptography Circuit |
| s27 | 3.70 | 10 | 2-input ISCAS'89 Circuit [BBK89] |
| s298 | 2.78 | 60 | Circuit based on PLD Devices [BBK89] |
| s344 | 2.27 | 73 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s349 | 3.38 | 73 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s382 | 2.86 | 96 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s386 | 2.27 | 71 | Controller [BBK89] |
| s400 | 2.70 | 96 | Traffic Light Controller [BBK89] |
| s420 | 2.00 | 89 | Fractional Multipliers [BBK89] |
| s444 | 2.94 | 94 | Traffic Light Controller [BBK89] |
| s510 | 2.08 | 124 | Controller [BBK89] |
| s526 | 2.70 | 106 | Traffic Light Controller [BBK89] |
| s641 | 2.04 | 94 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s713 | 2.04 | 97 | 2-input ISCAS'89 Circuit [BBK89] |
| s820 | 2.04 | 146 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s820a | 2.04 | 146 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s832 | 2.00 | 142 | 2-input ISCAS'89 Circuit [BBK89] |
| s832a | 2.00 | 142 | 2-input ISCAS'89 Circuit [BBK89] |
| s838 | 2.00 | 199 | Fractional Multipliers [BBK89] |
| s953 | 2.56 | 237 | Controller [BBK89] |
| s953a | 2.56 | 237 | Controller [BBK89] |
| s1196a | 2.00 | 295 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s1196b | 2.00 | 295 | 4-bit Shift-and-Add Multiplier [BBK89] |
| s1238 | 2.00 | 294 | Combinational circuit pipelined [BBK89] |
| s1238a | 2.00 | 294 | 2-input ISCAS'89 Circuit [BBK89] |
| s1423 | 2.08 | 419 | 2-input ISCAS'89 Circuit [BBK89] |

| Continuation of Table 3.1 | | | |
|---|---|---|---|
| **Circuit** | **Frequency (GHz)** | **# of cells** | **Description** |
| s1423a | 2.08 | 419 | 2-input ISCAS'89 Circuit [BBK89] |
| s1488 | 2.00 | 317 | 2-input ISCAS'89 Circuit [BBK89] |
| s5378 | 2.00 | 772 | 2-input ISCAS'89 Circuit [BBK89] |
| s5378a | 2.00 | 772 | 2-input ISCAS'89 Circuit [BBK89] |
| s9234 | 2.04 | 531 | Circuit based on real-chip [BBK89] |
| s9234a | 2.04 | 531 | Circuit based on real-chip [BBK89] |
| s13207 | 2.00 | 694 | Circuit based on real-chip [BBK89] |
| s13207a | 0.50 | 694 | Circuit based on real-chip [BBK89] |
| s15850 | 2.00 | 2205 | Circuit based on real-chip [BBK89] |
| s15850a | 2.00 | 2205 | Circuit based on real-chip [BBK89] |
| s35932 | 2.00 | 5334 | 2-input ISCAS'89 Circuit [BBK89] |
| s38417 | 2.00 | 6537 | Circuit based on real-chip [BBK89] |
| s38584 | 2.00 | 6690 | Circuit based on real-chip [BBK89] |
| tdes | 2.00 | 52676 | Triple-DES Cryptography Circuit |
| yeah | 2.04 | 3484 | YeAH! Network-on-chip Router [MHH+14] |

## 3.2 Selection of Standard Cells

This Section presents and evaluates the two criteria proposed to select standard cells to compose a reduced cell library: cell usage and number of cell inputs.

### 3.2.1 Approach 1: Selection Based Statistical Analysis of Cell Usage

The initial approach to select cells for the reduced library is based on statistical analysis of cell usage. With the goal of having a reduced library that contains only the most often used gates, each standard cell from the full library was classified by the percentage of benchmark circuits that employ at least one instance of it. The idea is that if the synthesis tool is able to implement the functionality of the least used gates with the most common ones, there could be a drastic reduction on the number of characterizations without large overheads. To test this hypothesis, the cells were binned considering a minimum percentage of use criteria. Standard cells with identical logic function but different drive strength were grouped and are referenced by the function name. Three library candidates were considered: *i)* lib70, containing cells used in at least 70% of the benchmark circuits; *ii)* lib60, with cells present in at least 60% of the circuits; *iii)* lib50, consisting of cells employed in

at least 50% of the benchmarks. This analysis was based on the synthesis of benchmark circuits targeting the full library.

Table 3.2 shows a list of cell functions from the full library sorted by the percentage of use on benchmark circuits – unused functions were omitted. From the Table, we can determine which cells are present on each library candidate. For instance, lib70 contains 20 logic gates, 19 of them implementing combinational logic functions and one is a flip-flop. Lib60 extends lib70 with 5 combinational gates. Lib50 adds 6 combinational logic functions to Lib60. Thin horizontal lines in the table mark the end of each library candidate set. Because there is only one type of sequential logic function available (DFPQ), the synthesis tool cannot successfully synthesize all benchmark circuits as some of them require latches and flip-flops with asynchronous resets. Therefore, this initial analysis refutes the hypothesis that the functionality of the least used gates can be implemented with the most common ones, invalidating this approach to select the cells.

Table 3.2 – List of cell functions from the full library sorted by the percentage of use in the benchmark circuits.

| **Cell Function** | **% of Circuits** | **Description** |
|---|---|---|
| IV | 98.28 | Inverter |
| NAND2 | 98.28 | 2-input NAND |
| NOR2 | 98.28 | 2-input NOR |
| AOI12 | 96.55 | 2-input AND into 2-input NOR |
| OAI12 | 94.83 | 2-input OR into 2-input NAND |
| NOR3 | 93.10 | 3-input NOR |
| AOI22 | 87.93 | Double 2-input AND into 2-input NOR |
| AOI112 | 87.93 | 2-input AND into 3-input NOR |
| AOI13 | 84.48 | 3-input AND into 2-input NOR |
| OAI22 | 84.48 | Double 2-input OR into 2-input NAND |
| NOR2A | 84.48 | 2-input NOR with A input inverted |
| NAND2A | 82.76 | 2-input NAND with A input inverted |
| NAND3 | 81.03 | 3-input NAND |
| OR2 | 79.31 | 2-input OR |
| NOR3A | 75.86 | 3-input NOR with A input inverted |
| AND3 | 75.86 | 3-input AND |
| OAI211 | 74.14 | 2-input OR into 3-input NAND |
| AO12 | 72.41 | 2-input AND into 2-input OR |
| DFPQ | 72.41 | Positive edge triggered non-scan D flip-flop, having non-inverted output Q only |

Continuation of Table 3.2

| Cell Function | % of Circuits | Description |
|---|---|---|
| AND2 | 70.69 | 2-input AND |
| AOI211 | 68.97 | 2-input AND into 3-input NOR |
| OA12 | 65.52 | 2-input OR into 2-input AND |
| NAND4AB | 63.79 | 4-input NAND with A and B inputs inverted |
| MUXI21 | 62.07 | 2:1 inverting multiplexer with coded selects |
| NOR4AB | 60.34 | 4-input NOR with A and B inputs inverted |
| AO112 | 58.62 | 2-input AND into 3-input OR |
| OA112 | 58.62 | 2-input OR into 3-input AND |
| OAI21 | 56.90 | 2-input OR into 2-input NAND |
| XNOR2 | 55.17 | 2-input exclusive NOR |
| AOI21 | 51.72 | 2-input AND into 2-input NOR |
| XOR2 | 51.72 | 2-input exclusive OR |
| NAND3A | 46.55 | 3-input NAND with A input inverted |
| DFPQN | 44.83 | Positive edge triggered non-scan D flip-flop, having inverted output QN only |
| CB4I1 | 41.38 | 4-input multi stage gate implementing function (AB + C)D |
| NAND3AB | 37.93 | 3-input NAND with A and B inputs inverted |
| BF | 36.21 | Buffer |
| AO22 | 36.21 | Double 2-input AND into 2-input OR |
| OAI222 | 34.48 | Triple 2-input OR into 3-input NAND |
| CBI4I6 | 32.76 | 4-input multi stage gate implementing function !((A+B)C + D) |
| OA22 | 32.76 | Double 2-input OR into 2-input AND |
| XOR3 | 29.31 | 3-input exclusive OR |
| AOI222 | 25.86 | Triple 2-input AND into 3-input NOR |
| PAO2 | 22.41 | 2 bit programmable AND/OR logic |
| MX41 | 18.97 | 4:1 non-inverting multiplexer with individual selects |
| AND4 | 15.52 | 4-input AND |
| OAI112 | 15.52 | 2-input OR into 3-input NAND |
| MUX21 | 12.07 | 2:1 non-inverting multiplexer with coded selects |
| OR2AB | 10.34 | 2-input OR with A and B inputs inverted |
| AO212 | 6.90 | Double 2-input AND into 3-input OR |
| XNOR3 | 6.90 | 3-input exclusive NOR |
| AO222 | 5.17 | Triple 2-input AND into 3-input OR |
| DFPSQ | 5.17 | Positive edge triggered non-scan D flip-flop, with active low asynchronous preset, having inverted output QN only |
| OA222 | 5.17 | Triple 2-input OR into 3-input AND |
| FA1 | 5.17 | Full-adder having 1 bit input operand |

| Continuation of Table 3.2 | | |
|---|---|---|
| **Cell Function** | **% of Circuits** | **Description** |
| DFPRQ | 5.17 | Positive edge triggered non-scan D flip-flop, with active low asynchronous reset, having non-inverted output Q only |
| NOR4 | 1.72 | 4-input NOR |
| NAND4 | 1.72 | 4-input NAND |
| LDHQ | 1.72 | Active High transparent latch, having non-inverted output Q only |
| HA1 | 1.72 | Half-adder having 1 bit input operand |
| OR4 | 1.72 | 4-input OR |

### 3.2.2 Approach 2: Selection Based on the Number of Inputs

The former approach did not generate good reduced library candidates due to the lack of gates to implement sequential logic. The selection based on a maximum number of inputs constraint aims to overcome this limitation, producing a reduced library capable of successfully synthesizing all benchmark circuits without a large overhead. Three library candidates were considered in this analysis: *i)* 2-input Lib, containing cells with at most 2 inputs; *ii)* 3-input Lib, with all cells from the 2-input Lib plus 3-input cells; *iii)* 4-input Lib, consisting of all cells from the previous libraries with the addition of 4-input gates.

Table 3.3 – Synthesis results for reduced library candidates based on the number of inputs.

| | **2-input Lib** | **3-input Lib** | **4-input Lib** |
|---|---|---|---|
| **Average Area Overhead** | 1.1238 | 1.0074 | 0.9822 |
| **Area Overhead Std. Dev** | 0.1115 | 0.0799 | 0.0649 |
| **Number of Timing Violations** | 1 | 0 | 0 |
| **Number of Analyzed Circuits** | 55 | 58 | 58 |
| **Number of Logic Functions** | 16 | 38 | 59 |
| **Number of Cells (w/ drivers)** | 97 | 184 | 261 |

Table 3.3 shows the performance results of the reduced library candidates, obtained from the flow depicted in Figure 3.1. The average area overhead indicates the area difference between the synthesis with the reduced and full libraries. It is interesting to note that the circuits synthesized with the 4-input Lib are on average smaller than those synthesized with the full library. The 2-input Lib could not fulfill timing constraints for one circuit. In addition, since the 2-input Lib does not have flip-flops with asynchronous reset (such cells

have 3 inputs), only 55 out of the 58 benchmark circuits could be synthesized. Number of Logic Functions indicates how many distinct cells are in the reduced library, without taking into account the various driving strengths. Number of Cells accounts for all cells, including the various driving strengths – this indicates the number of characterizations required to enable STA in a given voltage level.

Based on the aforementioned results, the 3-input Lib is the library candidate selected to be characterized. This reduced library enables the synthesis of all benchmark circuits with a very low average area overhead of 0.07%, fulfilling the previously defined criterion *i*. Moreover, according to [KRV+08], libraries restricted to a maximum fan-in of three avoid excessive transistor stacking, which reduces noise margin-related issues when operating in low supply voltages. Noise margin degradation is one of the main problems of near/sub-threshold operation because, due to process variations, pull-up or pull-down networks may randomly turn on. Criterion *ii* is verified along with the library characterization step, which is approached in detail in Section 4.3.

## 3.3 Library of 3-input cells

A library of cells with up to 3 inputs was selected to be used throughout the development of this work. Table 3.4 details the 38 logic functions present in the library: 12 for sequential logic and 26 for combinational logic. In addition, the Table contains the number of cells implementing each function – each cell has a different drive strength.

Table 3.4 – List of cells that compose the library of 3-input cells.

| Logic Function | # of Cells | Description |
|---|---|---|
| AND2 | 5 | 2-input AND |
| AND3 | 4 | 3-input AND |
| AO12 | 3 | 2-input AND into 2-input OR |
| AOI12 | 4 | 2-input AND into 2-input NOR |
| AOI21 | 5 | 2-input AND into 2-input NOR |
| BF | 10 | Buffer |
| DFPHQ | 3 | Positive edge triggered non-scan D flip-flop, with active high data enable, having non-inverted output Q only |
| DFPHQN | 3 | Positive edge triggered non-scan D flip-flop, with active high data enable, having inverted output QN only |
| DFPQ | 4 | Positive edge triggered non-scan D flip-flop, having non-inverted output Q only |

| Continuation of Table 3.4 | | |
|---|---|---|
| **Logic Function** | **# of Cells** | **Description** |
| DFPQN | 4 | Positive edge triggered non-scan D flip-flop, having inverted output QN only |
| DFPRQ | 2 | Positive edge triggered non-scan D flip-flop, with active low asynchronous reset, having non-inverted output Q only |
| DFPRQN | 2 | Positive edge triggered non-scan D flip-flop, with active low asynchronous reset, having inverted output QN only |
| DFPSQ | 2 | Positive edge triggered non-scan D flip-flop, with active low asynchronous preset, having inverted output QN only |
| DFPSQN | 2 | Positive edge triggered non-scan D flip-flop, with active low asynchronous preset, having inverted output QN only |
| IV | 11 | Inverter |
| LDHQ | 2 | Active high transparent latch, having non-inverted output Q only |
| LDHQN | 1 | Active high transparent latch, having inverted output QN only |
| LDLQ | 3 | Active low transparent latch, having non-inverted output Q only |
| LDLRQ | 2 | Active low transparent latch, with active low asynchronous reset, having non-inverted output Q only |
| NAND2 | 11 | 2-input NAND |
| NAND2A | 6 | 2-input NAND with A input inverted |
| NAND3 | 13 | 3-input NAND |
| NAND3A | 4 | 3-input NAND with A input inverted |
| NAND3AB | 4 | 3-input NAND with A and B inputs inverted |
| NOR2 | 14 | 2-input NOR |
| NOR2A | 7 | 2-input NOR with A input inverted |
| NOR3 | 6 | 3-input NOR |
| NOR3A | 4 | 3-input NOR with A input inverted |
| OA12 | 3 | 2-input OR into 2-input AND |
| OAI12 | 4 | 2-input OR into 2-input NAND |
| OAI21 | 5 | 2-input OR into 2-input NAND |
| OR2 | 4 | 2-input OR |
| OR2AB | 4 | 2-input OR with A and B inputs inverted |
| PAO2 | 4 | 2 bit programmable AND/OR logic |
| XNOR2 | 5 | 2-input exclusive NOR |
| XNOR3 | 4 | 3-input exclusive NOR |
| XOR2 | 6 | 2-input exclusive OR |
| XOR3 | 4 | 3-input exclusive OR |

# 4. CELL LIBRARY CHARACTERIZATION

Static timing analysis (STA) [RCN08] is a widely used technique to perform timing verification of digital circuits analytically. Different from dynamic (i.e. simulation-based) timing analysis, which is often time consuming and input-dependent, this method computes timing information based on pre-existing delay models. These models are included in the standard cell libraries and are usually generated based on data acquired from timing characterizations performed by foundries. Even though advanced node libraries are characterized for several corners, they usually do not cover the voltage levels that will likely drive ultra-low power IoT and wearable applications in the upcoming years – which are the target of the analysis proposed in this work. For instance, the libraries available with the ST-Microelectronics 28nm FDSOI design kit, which is the target technology in this work, are only characterized for supply voltages of 1.0V (the nominal supply voltage), 0.9V, and 0.8V at the typical process corner. Therefore, in order to leverage STA to analyze circuit behaviour under other supply voltage levels (e.g. at near-threshold or even sub-threshold values), additional research is necessary to obtain methods able to produce characterization data for the voltages of interest.

Section 4.1 of this Chapter provides background information related to cell characterization. The multi-voltage characterization flow, which is an original contribution of this work, is detailed in Section 4.2. Section 4.3 presents the characterization results for the reduced cell library (defined in Section 3.3) using this flow.

## 4.1 Background

This Section presents basic concepts on electrical characterization, including an overview of the characterization process and the models obtained from it. The Liberty Format, which is the industry standard library format to hold characterization data, is also briefly described. In addition, the Section presents the characterization tool Encounter Library Characterizer, developed by the EDA vendor Cadence.

### 4.1.1 Electrical Characterization of Standard Cells and Delay in Circuit Paths

Cell-based design is a widely used method for the creation of digital integrated circuits (ICs). It relies on the reuse of pre-designed circuits, called standard cells, as the building blocks of a design [WH10]. Standard cell libraries are typically supplied by foundries, target a specific silicon technology and implement a wide range of commonly used logic

functions – from simple gates, like inverters, NANDs and NORs, to more complex functional blocks such as adders and flip-flops. One of the main steps of a cell-based IC design flow is the selection of which cells to use and how to interconnect them to assemble a circuit that performs a given functionality. This step is called logic synthesis and take as input the target functionality of the circuit, which is usually supplied as a behavioral specification written in a hardware description language (HDL). The optimization engine in synthesis tools uses timing models contained in the standard cell libraries to guide the cell selection, trying to create a circuit that will perform according to a set of design-specific timing constraints. Timing closure is verified after synthesis through STA, which also relies on the same timing models. These models, which represent relevant electrical characteristics of each cell (i.e. timing, power, noise margin), are the result of the electrical characterization process.

The electrical characterization of standard cells is a well established process that can be performed automatically by library characterization tools [Cad13]. These tools usually take as input characterization settings and a SPICE-level post-layout netlist containing reference to specific transistor models, resistances, and capacitances for each library cell. The main output is a database that can contain the logic function, timing, power, and noise models for each cell. Figure 4.1 illustrates a high-level view of a characterization flow. Initially, the tool analyzes the transistor circuits in the cell SPICE netlist to identify the logic function and type – such as combinatorial logic, sequential logic, pass transistor logic, among others – of each cell.

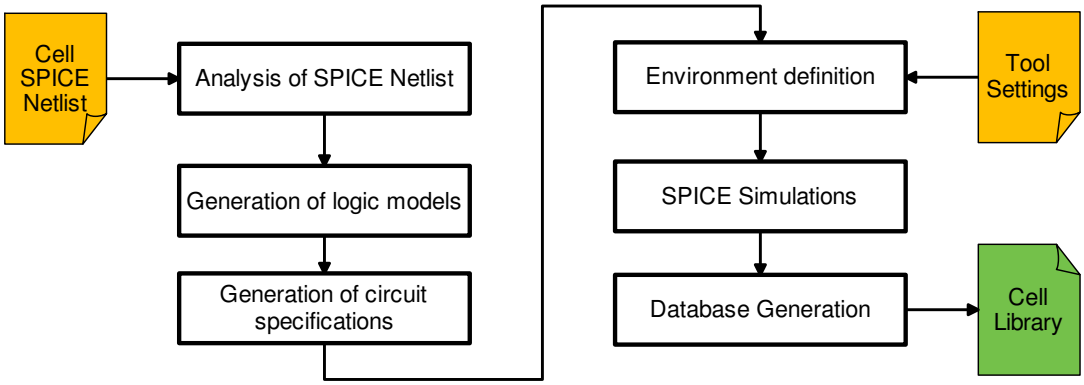



Figure 4.1 – High-level view of a cell characterization flow.

Based on the previous analysis, a logic model for each cell is created, followed by the generation of circuit specifications (i.e. pin direction, pin-to-pin delay, etc). Next, parameters, such as supply voltage, temperature, output load, and process corner, are read from a settings file and used to set the simulation environment. Finally, SPICE simulations are performed and the electrical characteristics of a cell are measured during simulation. The results are then extracted, processed, and stored on a database that can be exported to a library file.

Timing analysis is an integral part of the IC design process and ascertains important performance figures of the chip – for instance, the maximum operating frequency of synchronous systems, which is determined by the slowest logic path in the circuit. A logic path is defined as a path of logic stages where each of these connects a register output (or primary input) to a register input (or primary output) without any intervening registers, as Figure 4.2 illustrates.

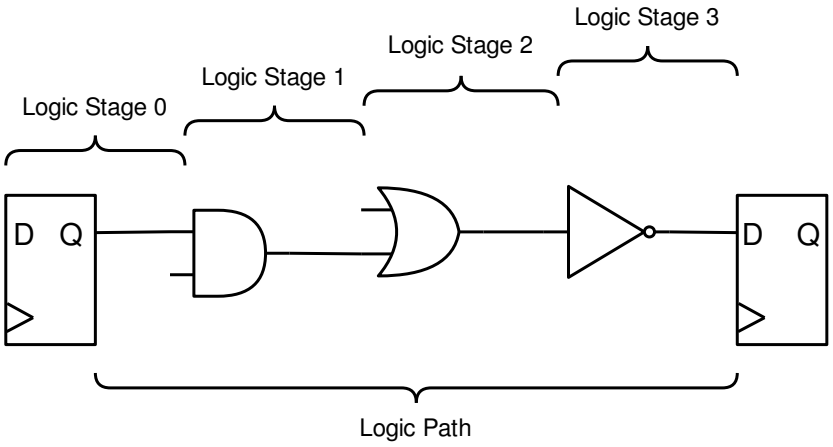



Figure 4.2 – Illustration of the logic path definition.

A logic stage represents one logic gate and the interconnect associated with its output. In this sense, the logic path delay ($d_{path}$) – that is, the time it takes for a signal to propagate from the start point to the end point of a logic path – is determined by the sum of the $n$ logic stages delay ($d_{stage}(i)$) that are part of the path, as Equation 4.1 shows[1]. The delay of a logic stage ($d_{stage}$) is defined as the delay between the input pin of a cell and the input pin of the next cell [Syn15], and, as Equation 4.2 shows, it has two components: the cell delay ($d_{cell}$), and the interconnect delay ($d_{connect}$).

$$d_{path} = \sum_{i=0}^{n} d_{stage}(i) \tag{4.1}$$

$$d_{stage} = d_{cell} + d_{connect} \tag{4.2}$$

The interconnect delay ($d_{connect}$) is the time it takes for a signal to propagate through the wire connecting the output pin of a cell to the input pin of the next cell. This delay depends on the resistance (R) and capacitance (C) characteristics of the wire, which is influenced by its length and width, among other factors. Realistic RC values can only be extracted after circuit layout; however, during logic synthesis, these characteristics are usually estimated from wire load models developed from statistical data provided by the foundry.

Cell delay ($d_{cell}$) is the time interval between a cell sensing an input transition and generating a corresponding output transition [Syn15] – that is, the propagation time from an

[1] This analysis disregards the setup time of the flip-flop sampling the last logic stage.

input to an output of a cell. This delay is computed as the time it takes for the output of a cell to reach a certain threshold voltage ($V_{th}$) after the input voltage goes above its $V_{th}$. The threshold voltage is a characterization parameter defined independently for each of input and output transition, resulting in up to four different values: input-rise, input-fall, output-rise, and output-fall. On the example depicted in Figure 4.3, for an input rise transition causing and output rise transition, $V_{th}$ is defined as 0.4V for both input and output rising transitions – therefore, the cell delay is measured from when the input pin has reached 0.4V until the output pin reaches the same voltage level. The cell delay is affected by both the slew rate of input signal (input slew) and the capacitance seen by the output pin of the cell (output load). The latter is primarily influenced by the output interconnect capacitance and the fan-out of the cell, while the former is determined by the output slew of the cell driving the input pin. In this way, large output loads or large input slews result in larger cell delays.

The output slew of a gate is determined by the output transition delay – that is, the time required for the output to switch from one logic state to the other. For rise transitions, this delay is measured as the time required for a gate to charge its output from a low-slew voltage ($V_{sl}$) to a high-slew voltage ($V_{sh}$). On fall transitions the opposite takes place: the transition delay is the time required to discharge the output from $V_{sh}$ to $V_{sl}$. These voltages are characterization parameters usually defined as 20% ($V_{sl}$) and 80% ($V_{sh}$) of the supply voltage – this means that voltages below $V_{sl}$ are interpreted as a logic low, and above $V_{sh}$ as logic high. On the example illustrated in Figure 4.3 (for a voltage supply of 1V), the transition delay is the time to charge the output from 0.2V to 0.8V. Similar to $d_{cell}$, the transition delay also depends on the output load and, in submicron technology, on the input slew [Syn15].

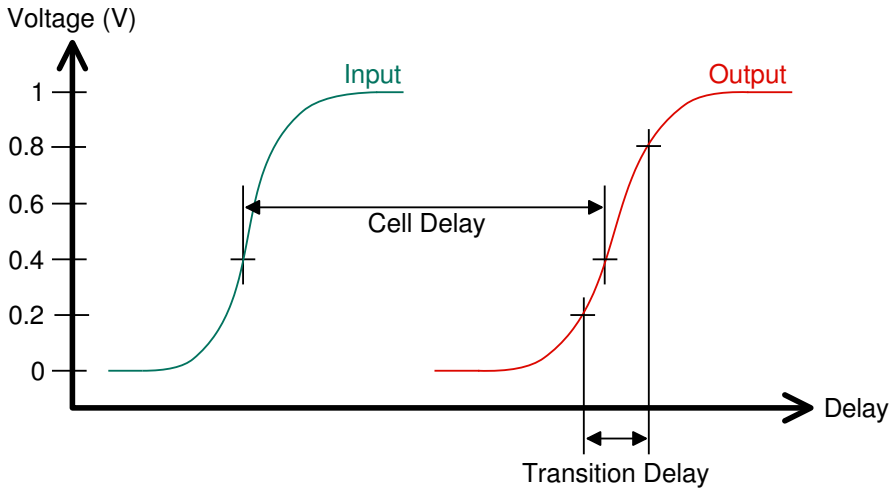



Figure 4.3 – Illustration of cell and transition delays of a positive-unate cell.

To efficiently handle the mentioned computations for circuits with millions of gates, the delay models extracted from the library characterization must be simple enough to compute timing analysis fast, yet accurate enough to yield realistic results [WH10]. Characterization tools usually support several types of delay models, and its up to the library designer to

choose the one that better suits the technology node of the library, considering the simplicity/accuracy trade-off. Three commonly used delay model classes can be identified below:

1. The Linear Model [DS89] - it uses an RC (resistor-capacitor) network and the input transition delay to estimate gate delay, disregarding the non-linear behaviour of transistors. This model is fast and memory efficient but not accurate enough to support the wide range of slews and loads found in contemporary ICs;

2. The Non-linear Delay Model (NLDM) [Syn15] - introduced in the early 1990s, this model overcomes the limitation cited for the Linear Model and has largely replaced linear models, becoming an industry standard in the early 2000s [WH10]. It employs tables indexed by input slew and output load to determine gate delay, providing a good simplicity/accuracy trade-off. This model, however, does not provide enough information to accurately compute delays through complex RC interconnect networks or to fully characterize noise events (i.e. crosstalk);

3. The Current Source Models (CSMs) [Syn06] - The limitations of the previous models drove the development of Current Source Models (CSMs), which express the output current of a cell as a non-linear function of its input and output voltages. Cell delays can be determined by integrating the output current to find the voltage as a function of time on a given RC network and solving for delay.

The NLDM is the timing model employed in the ST-Microelectronis FDSOI 28nm cell library, the target technology in this work. For this reason, other models will not be further addressed herein.

Characterization tools analyze the transistor topology of each cell to determine the number of delay tables required to cover all possible input events the cell may be subjected to. When using NLDM, four tables serve to model rising and falling cell and transition delays for each distinct timing path from a gate input to a gate output (i.e. a timing arc). Simple unate gates – such as inverters, NANDs, ANDS, NORs, and ORs – have one timing arc per cell input. A two-input NAND gate, for example, requires eight tables (two arcs, each represented with four tables), and an inverter (one arc) can be represented with four tables. Unlike simple gates, exclusive logic gates (i.e. XORs and XNORs) and complex gates (e.g. AND-OR-Invert) are state-dependent and require a larger number of arcs to cover every status combinations of other inputs [Syn15]. For instance, an XOR gate presents a non-unate behaviour – that is, a rising transition on one input may result on both a rising (positive unate) of falling (negative unate) output transition, depending on the status of other inputs. This gate, therefore, has two arcs per input, resulting in 16 tables to fully characterize the timing of a two-input XOR. The number of arcs per input for complex gates depends on the circuit topology used to implement. More details can be found in [Cad13].

So far, only path propagation delays (i.e. input-to-output delays) were addressed. However, state-holding cells such as latches and flip-flops also have input-to-input delay constraints that must also be characterized to ensure proper circuit operation. For instance, the setup time between data and clock ports of flip-flops determines how early a data signal must be stable before the active edge transition of the clock signal. Even though these constraints must be considered when designing a circuit, the analysis proposed in this work only deals with path propagation delays. For this reason, input-to-input delay constraints will not be further discussed – more details about these can be found in [WH10] and [Cad13].

As previously mentioned, the NLDM delay tables are indexed by input slew and output load. Load and slew vectors, which can be specified either for each gate or for groups of cells, are part of the characterization settings. The length of these vectors determine the table size – bigger tables offer more precision, but are more resource-intensive. Table 4.1 illustrates the cell delay of an inverter for falling outputs. This table is indexed by 8 slews (from 0.003ns to 1ns) and 6 output loads (from 0.2fF to 44.7 fF), resulting in 48 delays organized in a 6x8 matrix. These delays are measured from SPICE simulations performed using all combinations between the slew and load vectors attributed to this cell. Figure 4.4 illustrates the 28 simulations executed by the characterization tool to extract the information presented on the Table. Each of the eight different input slews, shown in red, was used to simulate 6 different loads – resulting in the 48 output curves, shown in blue. From each simulation, the characterization tool is able to extract both cell and transition delays for a given slew/load setting. Therefore, each timing arc requires two SPICE simulations for each slew/load setting to generate the rising and falling cell and transition delay models (4 tables).

Table 4.1 – NLDM delay table for a falling output cell delay of an inverter. Delays are in nanoseconds.

| | | **Input Slew (ns)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **0.003** | **0.017** | **0.033** | **0.065** | **0.13** | **0.25** | **0.5** | **1.0** |
| **Output Load (pF)** | **0.0002** | 0.004123 | 0.008446 | 0.011673 | 0.016671 | 0.024327 | 0.036017 | 0.056504 | 0.092996 |
| | **0.0019** | 0.008288 | 0.015279 | 0.02082 | 0.029736 | 0.042849 | 0.061754 | 0.09168 | 0.140273 |
| | **0.0037** | 0.012499 | 0.020347 | 0.027363 | 0.038431 | 0.055499 | 0.079175 | 0.117136 | 0.174379 |
| | **0.0075** | 0.021366 | 0.030135 | 0.038582 | 0.052206 | 0.07485 | 0.106028 | 0.154472 | 0.229839 |
| | **0.015** | 0.038849 | 0.047653 | 0.057604 | 0.073959 | 0.102733 | 0.143109 | 0.208558 | 0.306826 |
| | **0.0447** | 0.108063 | 0.116847 | 0.126705 | 0.146026 | 0.184704 | 0.244284 | 0.342094 | 0.496511 |

Once a cell library is fully characterized, it possible to use the delay models to determine the delays of logic paths. Timing analysis tools consider the logic behaviour to determine which timing models will be employed for each cell. For instance, consider the circuit in Figure 4.5, where the fall delay across the timing arc of gate *U1* needs to be computed using the NLDM. Since delay tables are indexed by input slew and output load,

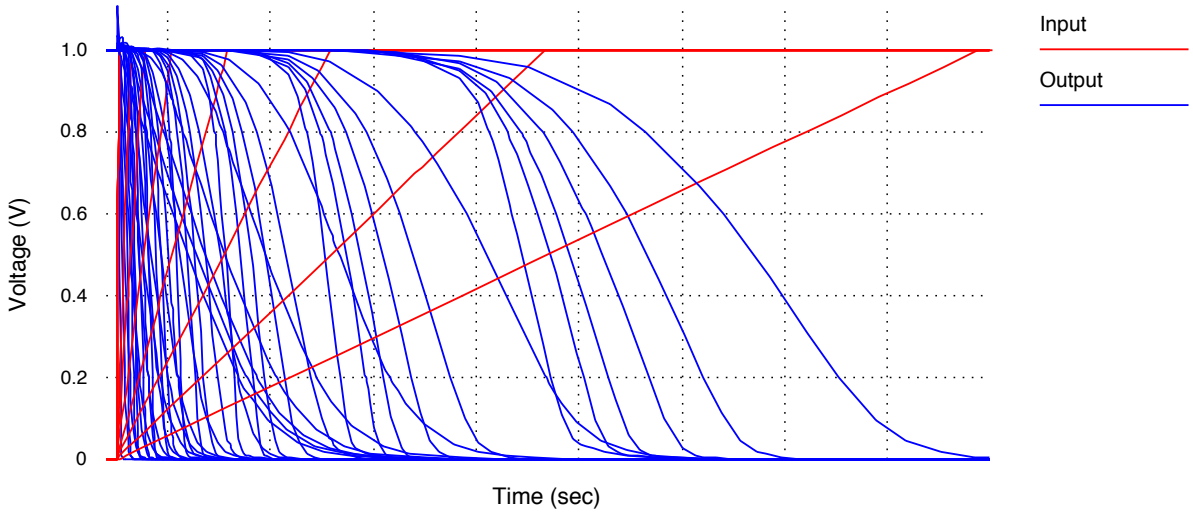



Figure 4.4 – Waveform showing the 48 SPICE simulations required to characterize a rising input of an inverter considering eight input slews and six output capacitances. Red lines show input stimuli and blue lines denote gate outputs.

these values must be determined in order to compute the cell delay. The input slew of *U1* is determined based on the transition delay of the gate driving its input pin: in this case, *U0*. Due to the negative unate behaviour of *U1*, the rise transition table of *U0* is used to determine *U1*'s input slew. If multiple timing arcs exist at cell *U0*, the maximum rise transition time of those arcs is selected [Syn15]. The output load is computed by addition of the capacitance introduced by the pins connected to net $N_1$ and the wire capacitance. The latter can be calculated based on wireload models or back-annotated from a post-synthesis layout. If the computed input slew and output load are not indices of the table, the delay will be either interpolated or extrapolated. Two-dimensional interpolation is performed in cases where the slew/load values are within table boundaries. Otherwise, the delay will be extrapolated based on the data available on the model. Generally, interpolated results are more accurate than extrapolated ones. For this reason, it is important to select realistic slew and load vectors when creating the characterization settings. This way, the NLDM tables encompass the slew/load values present in most circuits. More details about interpolation and extrapolation can be found at [Syn15].

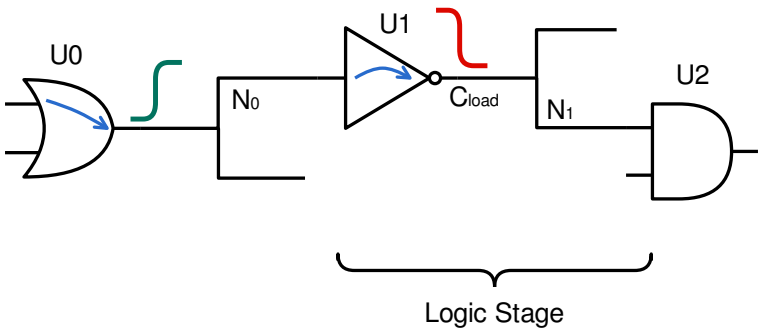



Figure 4.5 – Circuit schematic of an example circuit used to compute cell delay with NLDM.

In addition to timing, other aspects of standard cells are usually modeled by characterization tools. Among these, the most common ones are power and noise. These models are outside the scope of this dissertation and will not be discussed here. Details about noise and power characterization can be found in [Cad13] and [Syn15].

### 4.1.2 Technology Libraries

As mentioned in the previous Section and illustrated in Figure 4.1, the cell models created by characterization tools are exported to a library file. This file, which is usually one of the inputs to synthesis and sign-off tools, contains not only power, delay, and noise models, but also information about the logic functions of gates, about the specific CMOS technology and physical characteristics of cells – such as pin capacitance and area, among others. The current industry standard library format is the Synopsys Liberty format, which generates human-readable library files with the structure illustrated in Figure 4.6.

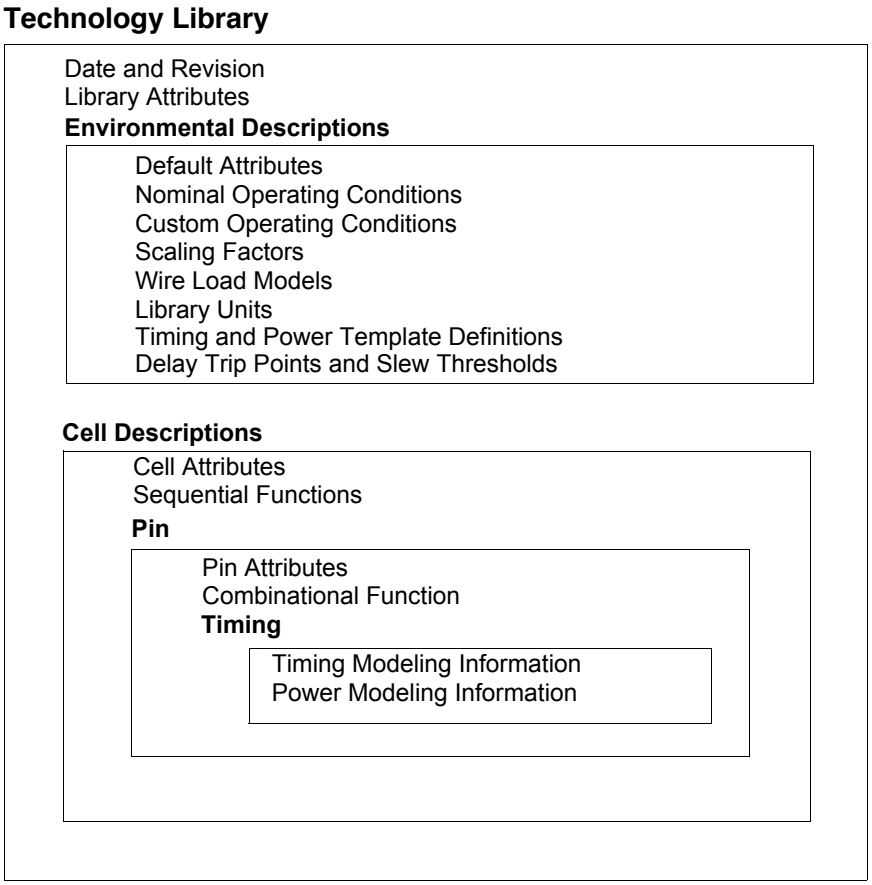



Figure 4.6 – High-level illustration of the structure of a technology library, containing environmental and cells descriptions. Based on the Symopsys Liberty format [Syn15].

The information contained in the Liberty format is divided in two groups: environmental descriptions and cell descriptions. The former contains technology information that

is not unique to individual cells – examples are the operating conditions, statistical data of interconnect estimation (i.e. wireload models), units, and default cell attributes. The latter describes the individual characteristics of each cell in the library – such as logic function of the cell, pin connectivity and timing information. The NLDM tables mentioned in the previous section are part of the timing group of a cell description.

### 4.1.3 Encounter Library Characterizer

Encounter Library Characterizer (ELC) is an automated characterization tool commercialized by the EDA vendor Cadence. It uses a flow similar to the one presented in Section 4.1.1 to generate timing, power, and noise models for standard cells in the Liberty format. As Figure 4.7 illustrates, the inputs to ELC can be grouped in three categories: standard cell library files, characterization settings and tool commands. The outputs are the characterization data in the Advanced Library Format (referred in Figure 4.7 as *ALF*) and Liberty format (*LIB*), along with a log file (*Report*) with information about the characterization process. The *ALF* file can latter be converted to other formats, such as Verilog and VHDL libraries or datasheets in HTML format.

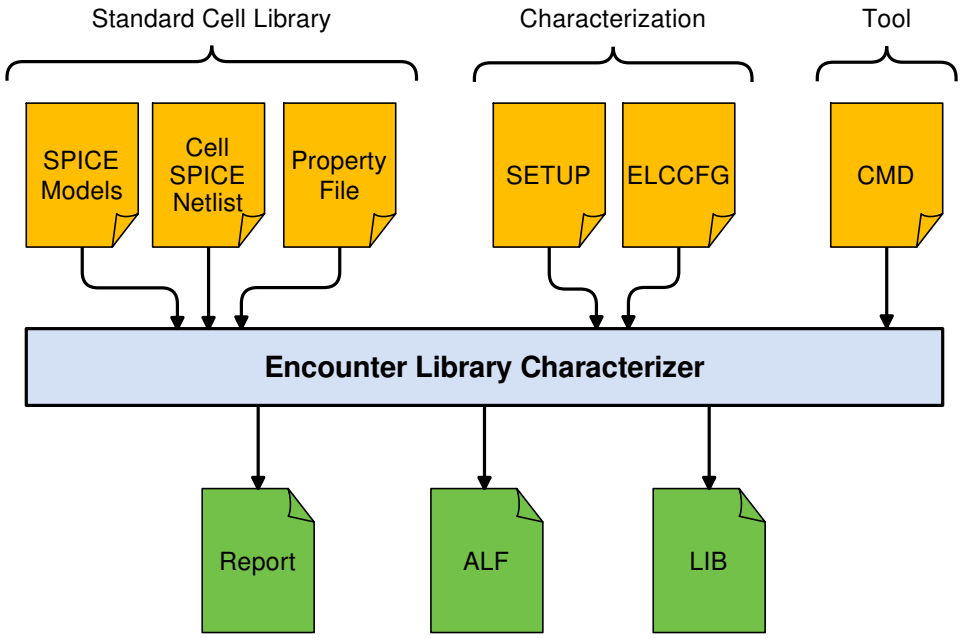



Figure 4.7 – Inputs and outputs of the Encounter Library Characterizer tool.

One of the most important inputs of a characterization tool is the electrical description of each standard cell, as this is imperative to generate delay, power, and noise models. ELC reads this information from two SPICE files: one, referred as *Cell SPICE Netlist*, describes the post-layout characteristics of each standard cell as a SPICE-format subcircuit; another, called *SPICE Models*, contains models for the devices (i.e. transistors, diodes) used

in the previous file. These files are usually part of the technology design kit and are provided by the library vendor. Additional cell information, such as area and footprints can be optionally supplied to the tool via a property file. Since property information is not used by the characterization engine, will simply be transferred as is to the output *LIB* and *ALF* files.

The characterization parameters are determined by two files: simulation setup (*SETUP*) and *ELCCFG*. The former specifies characterization parameters for each standard cell, while the latter contains environment variables, setup directives, and paths to other input files. *ELCCFG* is automatically loaded by ELC prior to the execution of any commands, to guarantee the characterization environment is properly set. Listing 4.1 shows an example of an *ELCCFG* file.

```
1  ###############################################################
2  # Settings to generate input slew from driver cell
3  ###############################################################
4  # Use a non-linear input slew signal instead of linear input
5  EC_INPUT_NONLINEAR=1;
6  # Use PWL information from driver cell as input waveform
7  EC_PWL_FROM_DRIVER=1;
8  EC_RECHAR_DRIVER=1;
9
10 ###############################################################
11 # Simulation Settings
12 ###############################################################
13 # Select simulator
14 EC_SIM_TYPE="Spectre";
15 EC_SIM_NAME="spectre";
16 # Specify power rail names
17 EC_SPICE_SUPPLY0_NAMES="gnd";
18 EC_SPICE_SUPPLY1_NAMES="vdd";
19 # Enable parallel simulation
20 EC_SIM_USE_LSF=1;
21 EC_SIM_LSF_CMD="";
22 EC_SIM_LSF_PARALLEL=8;
23 # Only load SPICE models that are actually used in the netlists
24 EC_SPICE_SIMPLIFY=1;
25
26 ###############################################################
27 # Models and Corner Settings
28 ###############################################################
29 # Device Models
30 MODEL="../models/stm28nm_fdsoi.lib";
31 # SPICE Models of Standard Cells
32 SUBCKT="../models/lib3in.sp";
33 # Simulation Setup File
34 SETUP="../settings/stm_setup.st";
35
```

```
36 ############################################################
37 # Characterization Settings
38 ############################################################
39 # Process Corner
40 PROCESS="tt28_1000mV_25C";
41 # Select all cells in the SUBCKT netlist for characterization
42 DESIGNS="*";
```

Listing 4.1 – Example of an ELCCFG file.

Initially, in lines 1-8, environment variables enable the use of non-linear input waveforms for cell characterization. These generate more realistic results, as non-linear slopes emulate the inputs that the gates will be subjected to during circuit operation in a better way. In fact, the use of ramped linear waveforms can by itself impact the accuracy of delay calculation by 5-10% [Cad13]. A piecewise linear (PWL) function can be automatically generated by ELC to approximate the non-linear output behaviour of a gate from the standard cell library. To do so, the simulation setup file must include a field selecting a standard cell to be the driver, which should be either an inverter or a buffer. This way, ELC can simulate the scenario of a gate input being connected to the output of a driver cell, increasing the accuracy of the characterization results. The waveforms in Figure 4.8 illustrate the impact of a driver cell in gate delay. The top chart shows a characterization scenario with linear input slews – that is, no driver cell is employed, while the bottom chart illustrates the scenario where a non-linear input slew is used – i.e. with a driver cell. When comparing both charts, it is possible to detect the delay difference in both input and output signals. The remaining lines of *ELCCFG* configure the SPICE simulation (lines 10-24), sets the paths to the other input files (lines 26-34), and selects the process corner and standard cells to be characterized (lines 36-42).

The simulation setup file (*SETUP*) specifies the characterization parameters for a given standard cell library. This file is written in a text-based format designed to describe how each standard cell should be characterized. The format supports many classes of parameters that can be combined to describe both global and cell-specific settings. Listing 4.2 exemplifies an ELC simulation setup file. Its parameters and classes can be sorted in two groups: library settings (lines 1-18) and cell-specific settings (lines 20-31). Library settings are applied to each standard cell and determine process corner, signal parameters and simulation settings. The circuit operating conditions, such as voltage and temperature, are determined by the process corner (lines 1-4). Signal parameters (lines 6-13) control how ELC interprets the signal voltage levels from the SPICE simulations to compute delay, power and noise models – for instance, these parameters define the threshold voltages employed in cell delay computation ($V_{sl}$ and $V_{sh}$), which were previously mentioned in Section 4.1.1 (refer to Figure 4.3). Simulation settings (lines 15-18) determine the duration and step of

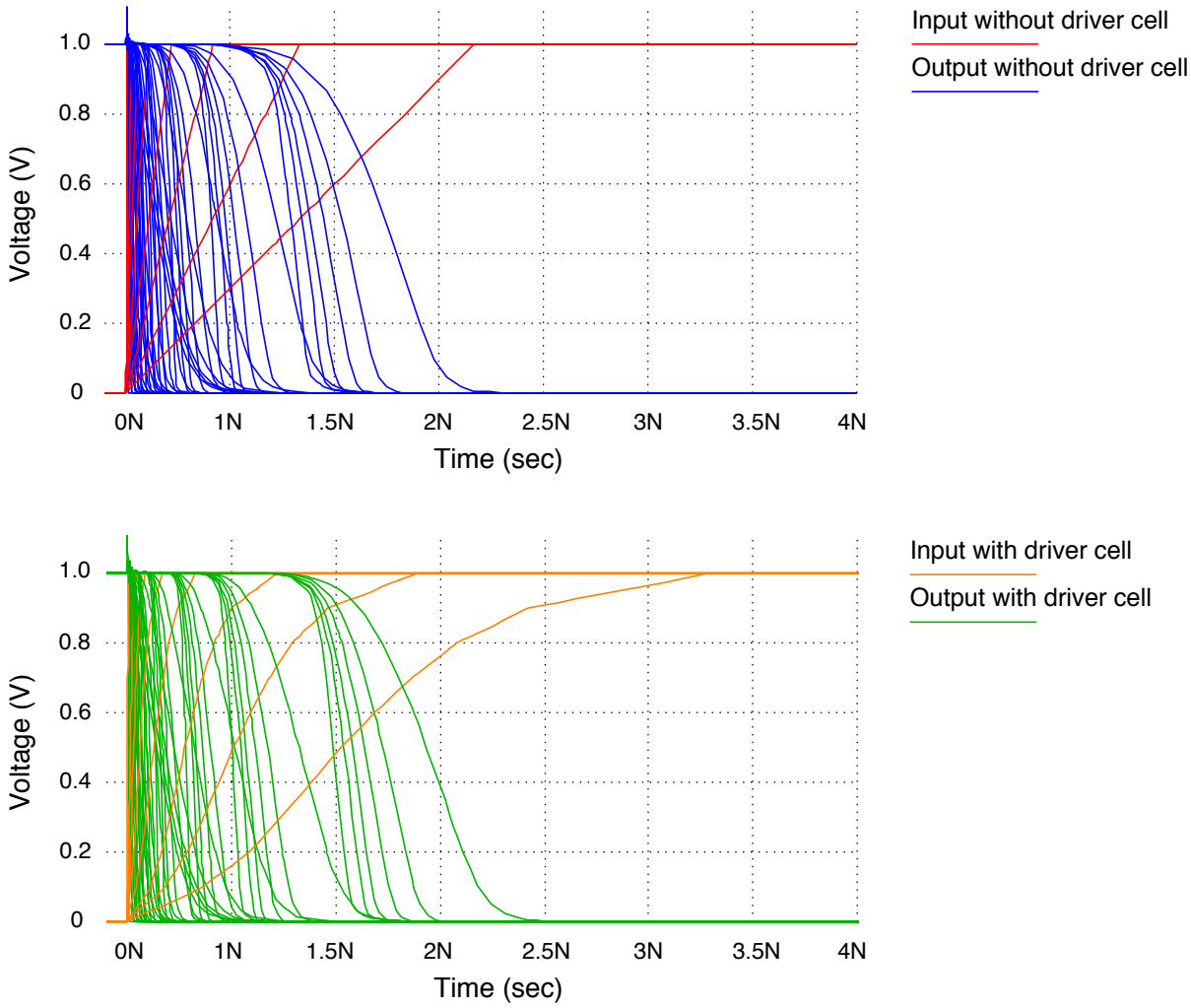



Figure 4.8 – Waveforms illustrating the rising input characterization of an inverter, considering two input scenarios: i) without driver cell (linear input), on the top chart with inputs in blue and outputs in red; ii) with driver cell (non-linear input), on the bottom chart with inputs in orange and outputs in green.

each SPICE simulation, and, among other parameters, the driver cell used to create the non-linear input waveform model for the library – *CELL_IVX3*, in this example (line 17).

Cell-specific settings determine, for each standard cell, the slew and load indexes that will be used to generate the characterization data. In ELC, index classes, which determine slew and load vectors, are associated with groups of cells. This way, it is possible to assign a unique slew and load vector to each individual gate, or to share vectors with a group of standard cells. For example, in lines 20-23 an index class is created with an input slew vector of eight values (line 21) and an output load vector of six values (line 22). Next, a group to hold cell whose names contain the characters "X3" is created in lines 25-27. In lines 29-31, the group is associated with the index class, assigning the slew and load vectors to each cell belonging to the group. The vector in this example creates the characterization example of Table 4.1 and Figure 4.4, previously shown in Section 4.1.1. The parameters in

lines 33-38 attach the signal and simulation parameters to the process settings. More details about the simulation setup file format can be found in [Cad13].

```
1  Process tt_1000mV_25C {
2    voltage    = 1.00 ;
3    temp       = 25   ;
4  } ;
5
6  Signal tt_1000mV_25C {
7    unit         = REL ;
8    Vh           = 1.0     1.0 ;
9    Vl           = 0.0     0.0 ;
10   Vth          = 0.4     0.6     0.4     0.6 ;
11   Vsh          = 0.8     0.8     0.8     0.8 ;
12   Vsl          = 0.2     0.2     0.2     0.2 ;
13 } ;
14
15 Simulation tt_1000mV_25C {
16   transient    = 0.5n 100n 3.0e-13 ;
17   incir        = CELL_IVX3 ;
18 } ;
19
20 Index X3_1000mV {
21   Slew  = 0.003N 0.017N 0.033N 0.065N 0.13N 0.25N 0.5N 1.0N ;
22   Load  = 0.0002P 0.0014P 0.0027P 0.0055P 0.011P 0.03278P ;
23 } ;
24
25 Group X3 {
26   CELL = *X3* ;
27 } ;
28
29 set index (tt_1000mV_25C) {
30   Group(X3)      = X3_1000mV ;
31 } ;
32
33 // Associating Process to other parameters
34 set process (tt_1000mV_25C) {
35   simulation = tt_1000mV_25C ;
36   signal     = tt_1000mV_25C ;
37   index      = X3_1000mV ;
38 } ;
```

Listing 4.2 – Example of an Encounter Library Characterizer Simulation Setup File (*SETUP*).

ELC provides a command line interface that can be accessed either interactively via a shell or through a TCL command file. Listing 4.3 illustrates an ELC command file (*CMD* in Figure 4.7) capable of automatically characterizing a standard cell library. Initially, user

variables are set with the process and library names, and the path to the library's property file (lines 1-5). Next, ELC commands are issued to create and prepare a characterization database (lines 8-9). Once the database is ready, a command initiates the batch of SPICE simulations required to characterize the standard cell library (line 10). When the simulations are complete, the *ALF* file is created (line 11), followed by the generation of a Liberty file (line 12). Finally, the database is closed and ELC is exited (lines 13-14). Further details about ELC can be found in [Cad13].

```
1  # Settings
2  set process_name "tt28_1000mV_25C";
3  set library_name "LIBRARY"
4  set property_file "./../settings/lib_properties.st"
5  set char_name "${library_name}_${process_name}"
6
7  # Perform Cell Characterization
8  db_open $library_name
9  db_prepare
10 db_spice -keep_log -keep_wave
11 db_output -state -alf ${char_name}.alf
12 alf2lib   -state -alf ${char_name}.alf -lib ${char_name}.lib -def ${
       property_file} -cfg elccfg
13 db_close
14 exit
```

Listing 4.3 – Example of an ELC command file (*CMD*).

## 4.2 Multi-voltage Characterization Flow

The multi-voltage characterization (MVC) flow is a method designed to extend the voltage corners supported by standard cell libraries. It provides a systematic way to characterize libraries to a wide range of target voltages, ensuring the proper scaling of voltage-dependent parameters – such as signal slew and voltage levels used to determine delays. This flow, which is the second original contribution of this work, provides the means to enable STA of circuits operating at virtually any supply voltage that is within the limits of the silicon technology used by the library. In fact, the timing analysis of circuits under voltage scaling (Chapter 5) was made possible by the characterization of the reduced cell library (Section 3.3) from nominal (1V) to sub-threshold (250mV) voltages using the proposed flow.

The MVC flow extracts voltage-dependent parameters from a reference library characterized at nominal voltage and scales them to create characterization environments compatible with the target voltages. These environments employ ELC to characterize the standard cell libraries and export the data in Liberty format. A tool designed to validate characterization data by comparing the generated libraries with the reference is also included in

the flow. As Figure 4.9 illustrates, the flow takes as input a reference library – composed by a set of Liberty files originated from cells characterized at nominal voltage, the technology models and cell netlists in SPICE format – and a settings file, containing the target supply voltages of the characterization. The outputs are Liberty libraries characterized at each target voltage. The flow consists of three tasks: nominal voltage environment creation, voltage-dependent parameter scaling, and multi-voltage characterization. Each of these tasks are detailed in the subsequent Sections.

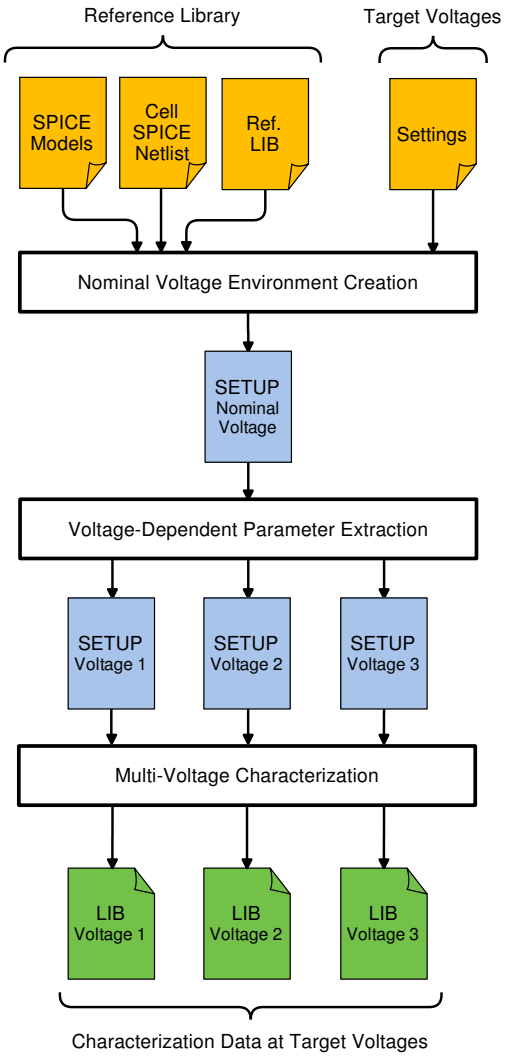



Figure 4.9 – Overview of the Multi-voltage Characterization Flow (MVC). Yellow boxes denote the main input files of the flow, green boxes designate outputs, and blue boxes indicate intermediate files.

### 4.2.1 Nominal Voltage Environment Creation

Nominal Voltage Environment Creation is the first task of the MVC flow. Its goal is to create a characterization environment at the technology nominal voltage, capable of

generating libraries with data equivalent to the reference library. Since this environment will be the basis to create the characterization environments for each target supply voltage, it is vital to ensure it can produce correct results. The input to this task is the reference library and the output is an ELC simulation setup file (similar to the one illustrated in Section 4.1.3) tuned to the nominal voltage. Figure 4.10 details the four steps this task encompasses.

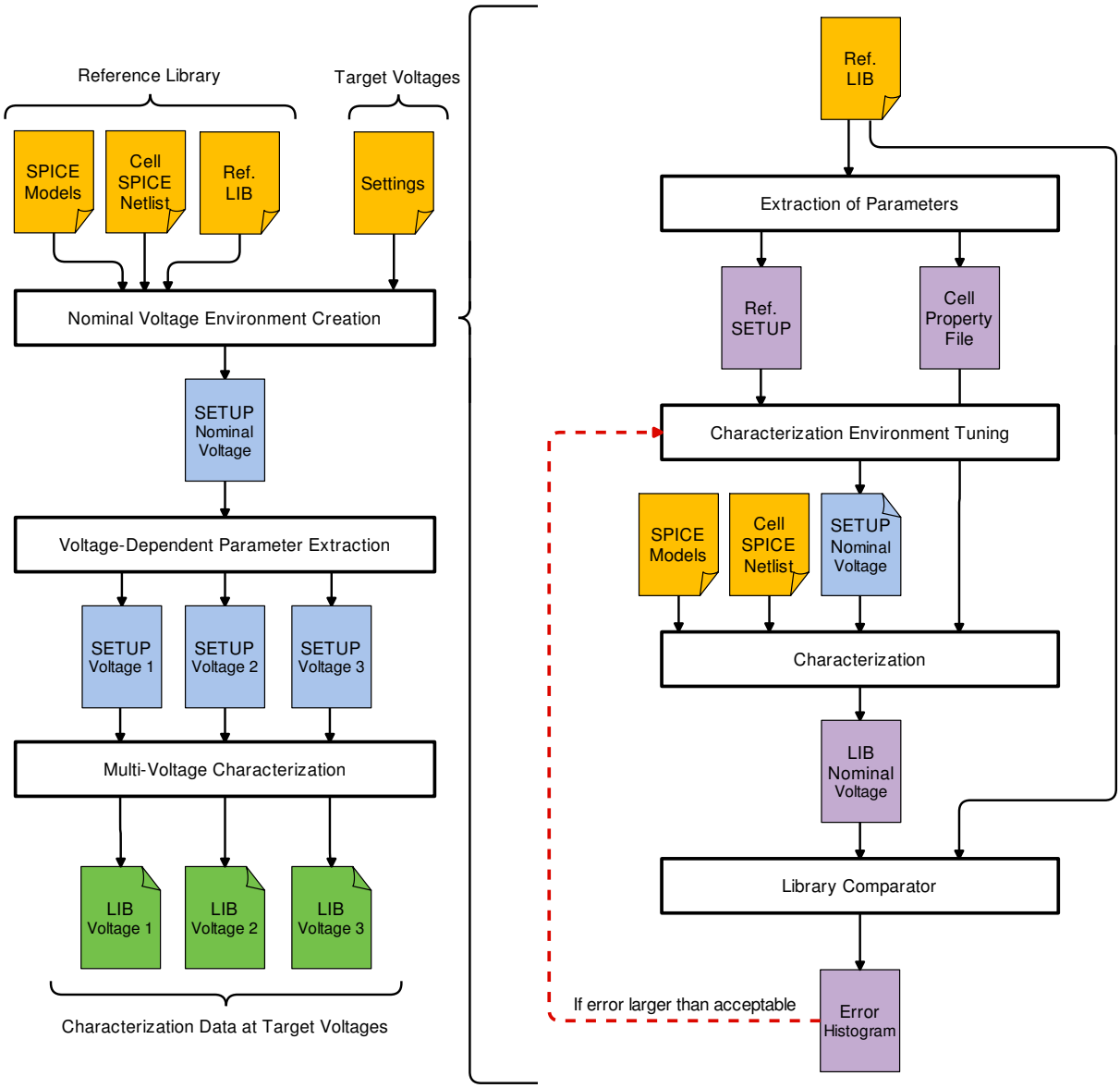



Figure 4.10 – Detailed view of the Nominal Voltage Environment Creation task of the MVC flow. Yellow boxes denote the main input files of the flow, green boxes designate outputs, and blue and violet boxes indicate intermediate files of MVC and of the task in question, respectively.

The initial step is the extraction of parameters from the reference library characterized at the nominal voltage. The goals are to create a *Cell Property File*, containing information about the cells that are not used by the characterization tool (i.e. cell area and footprint), and a reference ELC simulation setup file (*Ref. SETUP*). The creation of the former can be done using text manipulation tools, while the latter is automatically extracted using ELC. The

*Ref. SETUP* file contains the input slew and output capacitance vectors used in the reference library, along with simulation and corner settings – the details about the contents of the ELC simulation setup file were previously discussed in Section 4.1.3. The next step consists in creating an ELC simulation setup file (*SETUP Nominal Voltage*) based on the *Ref. SETUP* file, with the parameters grouped by drive strength of the cells. This eases the process of scaling voltage-dependent parameters that is executed in the second task of the MVC flow. Next, the cell library is characterized using *SETUP Nominal Voltage* as the simulation setup file. The resulting library is compared with the reference library using a custom Library Comparator (see details on the next Section below), and the delay differences are plotted as an error histogram. If the difference is not within the acceptable bounds, *SETUP Nominal Voltage* must be tuned to reduce the error. ELC offers several approaches for characterization environment tuning. For instance, the delay values extracted from SPICE simulations can be scaled or offset through specific parameters supported by the *SETUP* file. Also, the input slew and capacitance vectors and the driver cell can be adjusted to reduce the error. Details about environment tuning can be found in [Cad13]. Once the environment is tuned, the output *SETUP Nominal Voltage* file becomes the input of the next step of the flow.

The Library Comparator

Library Comparator (LC) is an automated environment for comparing timing models of Liberty libraries. The creation of a custom in-house environment for this purpose was necessary as the library comparator tool bundled with ELC (*libdiff*) does not support NLDMs. As Figure 4.11 shows, LC works by extracting logic path delays from a set of netlists using the provided libraries and comparing the obtained results. It relies on Synopsys PrimeTime to extract path delays and on an in-house tool called Timing Report Comparator to compare the results. The Timing Report Comparator encompasses two Python scripts of around 500 lines each.

To ensure that each netlist is thoroughly analyzed, PrimeTime generates unique timing reports for each path that connects a register output (or primary input) to a register input (or primary output) without any intervening registers – thus, guaranteeing that data from all timing paths are acquired. In addition, PrimeTime evaluates up to eight types of delay for each path, considering the combinations of: *i)* minimum and maximum delays; *ii)* falling and rising input signals; *iii)* falling and rising output signals. The Timing Report Comparator parses each timing report and, using Equation 4.3, computes cell and path delay error between reference and target libraries. LC combines the error data obtained from all netlists as error histograms. From the error histograms one can determine if both libraries have equivalent timing models – that is, both libraries generate similar STA reports.

$$error = (d_{target} - d_{reference})/d_{reference} \tag{4.3}$$

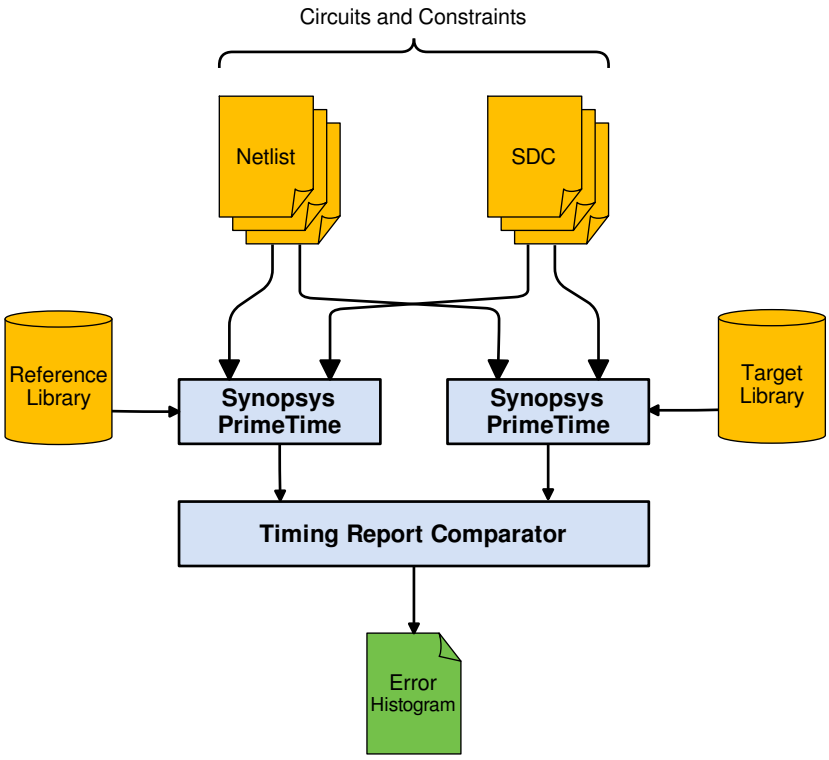



Figure 4.11 – Automated environment used to compare data models from Liberty libraries. STA is performed on a set of circuits using Synopsys PrimeTime. The Timing Report Comparator parses the STA reports, compares the data between reference and target libraries, and generates an error histogram with the results.

The reason behind the development of LC is the necessity to validate the characterization environments created by the MVC flow. In other words, it is necessary guarantee that libraries generated with this flow are accurately characterized. To assess this, the MVC-generated libraries are compared with the ones supplied in the design kit (i.e. reference library) using LC. If the error distribution shown in the histogram presents a small standard deviation and average error close to zero, it is assumed that the characterization is good. This validation approach is based on two assumptions: *i)* reference libraries are accurate; *ii)* libraries that present near-zero error when compared to the reference are also viewed as accurate. Based on observations of the delay difference seen between SPICE simulations of logic paths and STA reports, which can be in the order of 5-10%, a target error and standard deviation of up to 0.05 (5% error) were selected for this work. In this way, libraries within this target are considered accurate.

### 4.2.2 Voltage-Dependent Parameter Extraction

Characterization environments are collections of settings that establish how characterization tasks should take place. Some of these settings, such as temperature of oper-

ation and output load vectors, do not depend on supply voltage and, consequently, do not need to be modified to support other voltage corners. On the other hand, voltage-dependent parameters need to be carefully scaled when the target supply voltage of a characterization changes. Therefore, to enable accurate characterizations, each target supply voltage of the MVC flow requires a unique characterization environment with voltage-dependent parameters appropriately scaled. The goal of this task is to create these environments.

The voltage-dependent settings contained in the environments produced by the MVC flow can be categorized in two classes: *i)* signal parameters and *ii)* simulation parameters. The first class determines how signal voltage levels are interpreted by the characterization tool. These parameters are defined relatively to the voltage corner and are, consequently, scaled automatically by ELC. Class *ii* specifies the input slews that each standard cell will be subjected to during characterization. These values are expressed in absolute delay and need to be scaled to reflect the slew that a circuit would be exposed to when operating under a specific target supply voltage. As previously discussed, timing analysis tools use the transition delay of one gate to determine the input slew of the next one. If the input slew computed by the tool is outside the range defined in the delay model (i.e. values specified in class *ii*), it cannot reliably estimate cell and transition delays. Therefore, scaling these parameters in a realistic manner is imperative to avoid inaccurate delay computations. A technique based on SPICE simulations was devised to scale input slew vectors in a way that reflects the slew range found when operating on a given target supply voltage. The first step consists in determining the minimum and maximum signal slews that library cells produce when operating in a certain voltage corner. To do so, a SPICE deck was designed to reproduce these extreme values using the standard cells that exhibit the smallest and largest transition delay. As Figure 4.12 illustrates, this deck is composed of a step function generator, a pair of input inverters, a test cell, and load capacitances.

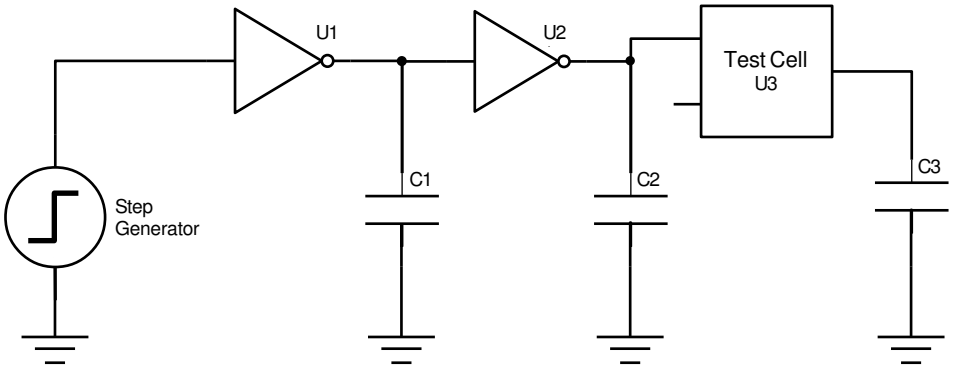



Figure 4.12 – Generic SPICE deck employed in parameter scaling. Test cell is selected according to the characteristics of the parameter to be scaled, and capacitances are tuned to reflect the expected slews.

When analyzing minimum slew, the test cell is replaced by the gate that presents the smallest transition delay. Similarly, the standard cell with the largest transition delay

is used as test cell to analyze maximum slew. This way, when the supply voltage scales, the minimum/maximum slew characteristics are maintained. The simulation stimulus, which is produced by the step generator, goes through a pair of inverters to generate a realistic non-linear input signal to the test cell. Once the load capacitances are calibrated to generate the minimum and maximum slew values employed in the reference environment, SPICE simulations are performed in all target supply voltages, to obtain the scaled values. These results determine the two extreme slew values for each voltage. With the extreme values determined, intermediate values can be interpolated using cross-multiplication. This is exemplified in Table 4.2, where *min* and *max* values were determined using the SPICE-based technique. Reference values represent the input slew vector at nominal voltage obtained from the reference library. The example assumes a slew vector of 5 values. Larger input vectors can be supported by adding additional columns.

Table 4.2 – Example of input slew scaling through cross-multiplication between reference and scaled values.

| | **Input Slew (ns)** | | | | |
|---|---|---|---|---|---|
| **Reference** | $a$ | $b$ | $c$ | $d$ | $e$ |
| **Scaled** | $min$ | $f = g * b/c$ | $g = h * c/d$ | $h = max * d/e$ | $max$ |

The second task of the MVC flow uses the scaling technique previously described to create the environments required to enable library characterization on all target supply voltages. This task, as Figure 4.13 details, takes as input the characterization environment created in the previous task and, after four steps, generates a collection of new environments, each tuned to a specific target supply voltage. The first step within this task consists in extracting the voltage-dependent parameters from the input environment. As previously mentioned, only the slew indexes used to create the NLDMs need to be explicitly scaled and, therefore, only these are extracted. Next, SPICE decks representing each of these parameters are created and simulated in all target supply voltages to obtain scaled values of minimum and maximum slews. These values are interpolated using the cross-multiplication technique exemplified in Table 4.2 to generate scaled input slew vectors. Finally, the original slew vectors are replaced by the interpolated ones, thus creating unique simulation setup files scaled for each target supply voltage.

### 4.2.3 Multi-voltage Characterization and Validation

The final step of the MVC flow consists on the cell library characterization at each target supply voltage. As Figure 4.14 illustrates, this step takes as input the characterization environments created in the previous task and outputs a set of Liberty libraries, one for each

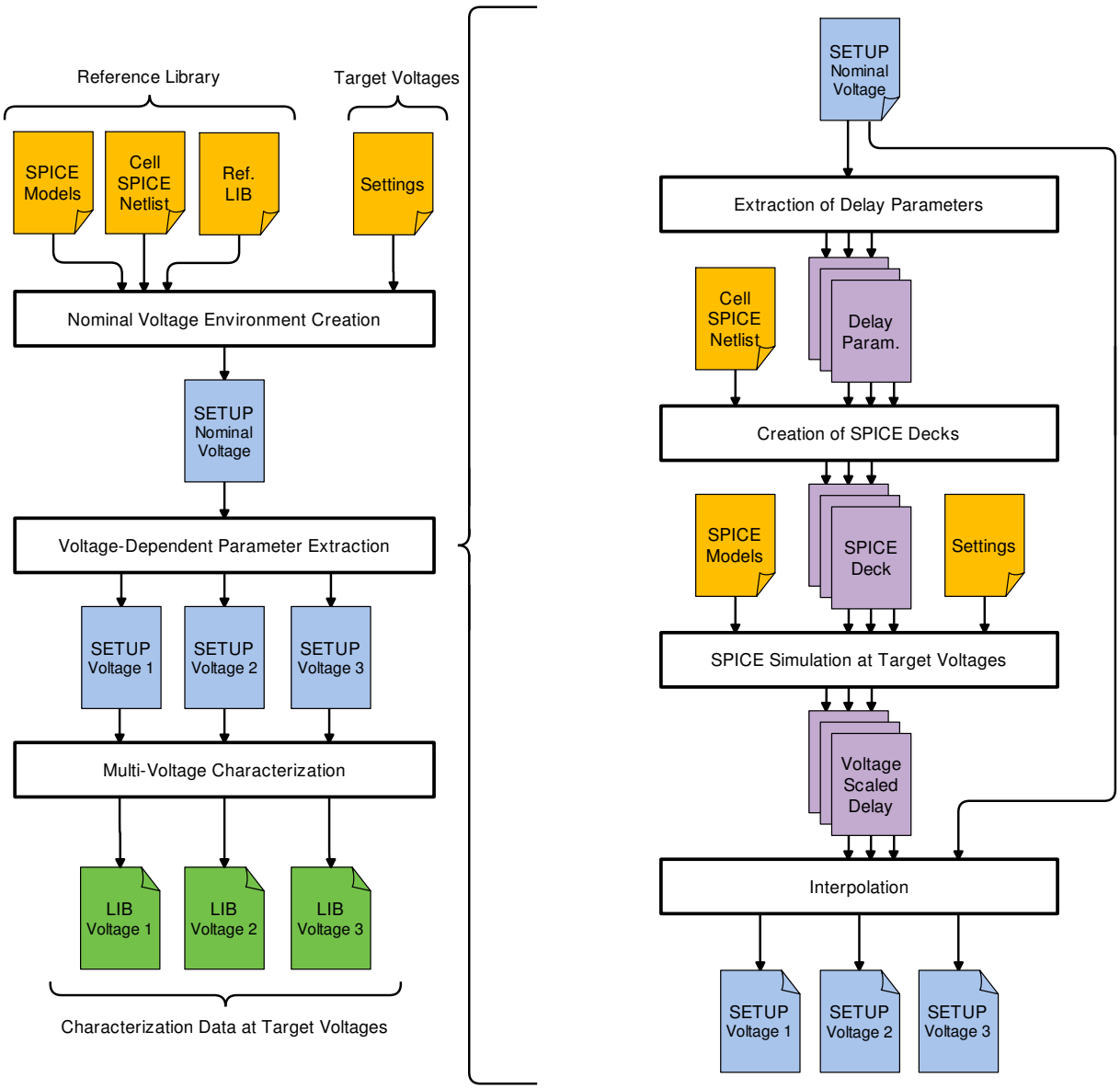



Figure 4.13 – Detailed view of the Voltage-Scaled Parameter Extraction task of the Multi-voltage Characterization Flow. Yellow boxes denote the main input files of the flow, green boxes designate outputs, and blue and violet boxes indicate intermediate files of MVC and of the task in question, respectively.

target supply voltage. If the design kit includes libraries characterized at any of the target supply voltages of the MVC flow, these can be employed on an additional validation step using the LC environment.

## 4.3 Case-study: Multi-voltage Characterization of Reduced Cell Library

This Section details the characterization effort performed to extend the voltage corners of the reduced cell library defined in Section 3.3 to support the timing analysis of circuits

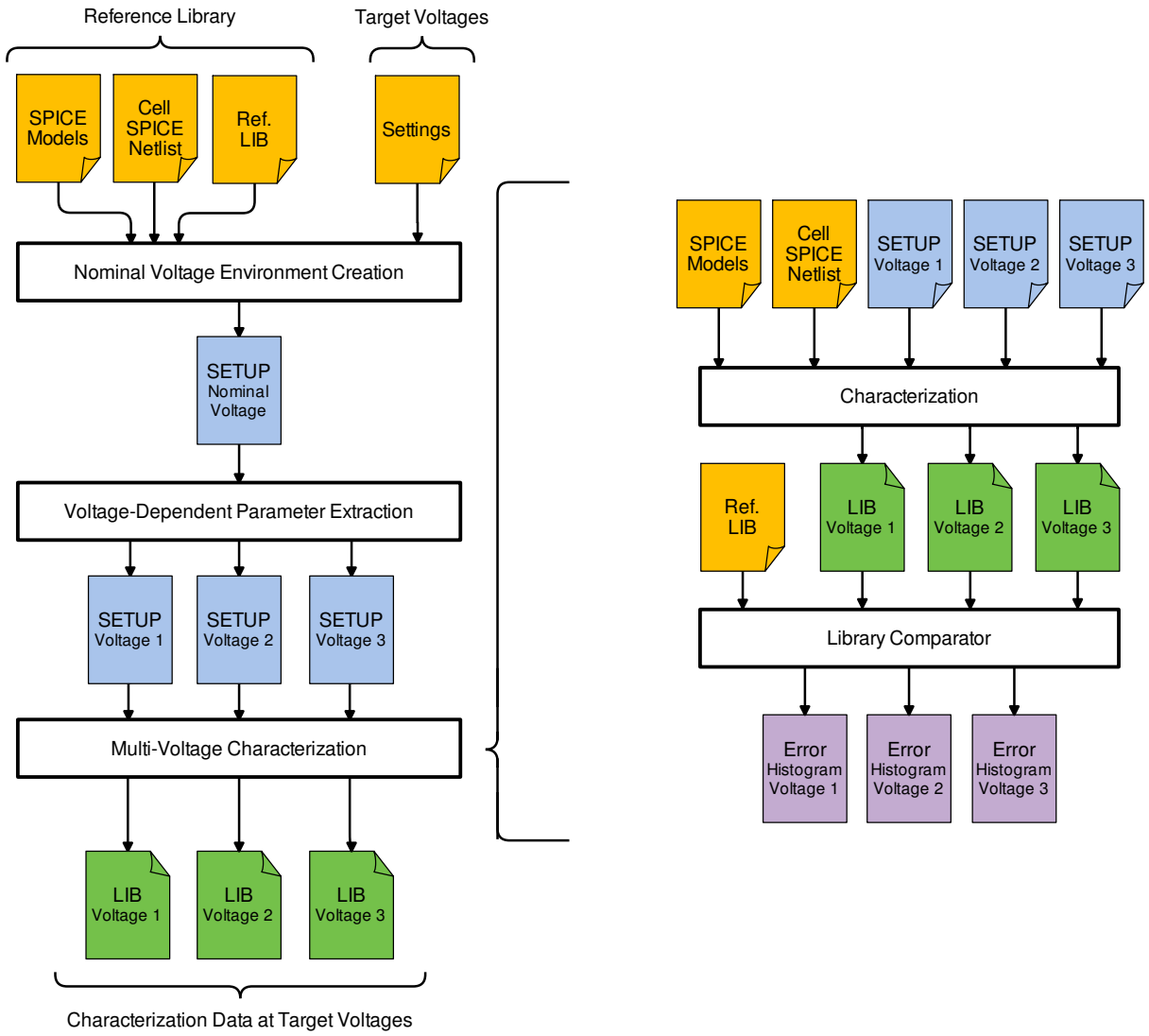



Figure 4.14 – Detailed view of the Characterization task of the Multi-voltage Characterization Flow. Yellow boxes denote the main input files of the flow, green boxes designate outputs, and blue and violet boxes indicate intermediate files of MVC and of the task in question, respectively.

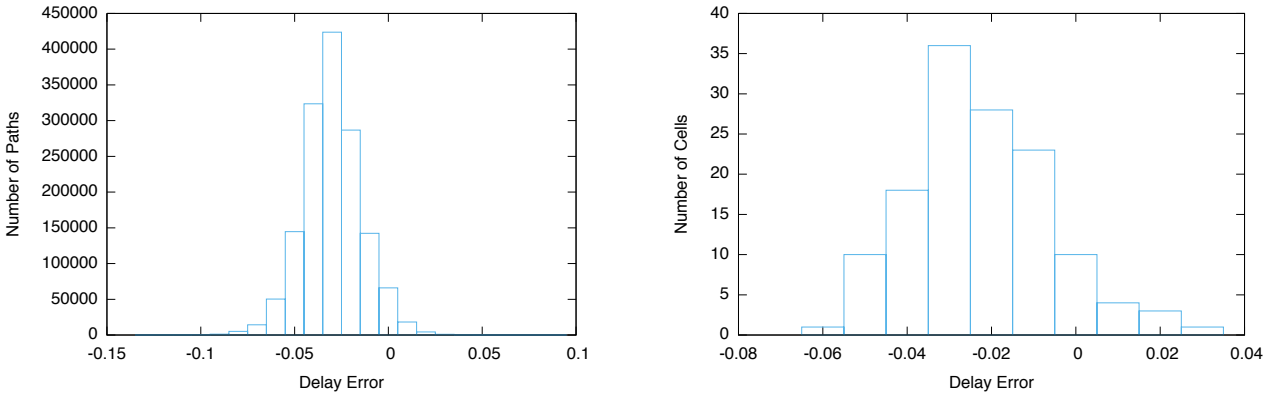
under voltage scaling detailed in Chapter 5. The goal is to characterize the library in the range from 1V (nominal) to 250mV in steps of 50mV. The reduced library employ cells from the ST-Microelectronics 28nm FDSOI CORE library. Gates from this library cannot operate correctly[2] when subjected to supply voltages under 250mV. When operating below this voltage level, some standard cells are not able to output signals with voltage levels above $V_{sh}$, which is the lower limit for a logic high signal.

As previously defined in Section 4.2, the first step of the MVC flow is the creation of a characterization environment tuned for nominal voltage using the method detailed in Section 4.2.1. A modified version of ST-Microelectronics 28nm FDSOI CORE library that contains only the set of gates that are part of the reduced library was used as the reference

[2]In the context of this work, the notion of *correct operation* refers to the standard cell being able to generate output signals within the bounds defined by the cell library. This analysis disregards other effects, such as noise and crosstalk.

library. From this library, the input slew and output capacitance vectors used in the characterization setup file were extracted. To increase the characterization accuracy, a midsized inverter was selected to be used as the driver cell. With this environment, the reduced cell library was characterized at nominal voltage (1V), and the results compared with the reference. The benchmark circuits described in Section 3.1.1 were used by LC to validate the library.

Figure 4.15(a) shows a histogram of path delay error grouped in bins of 0.01 (i.e. 1% error). The histogram, which includes the errors computed for each path of each circuit analyzed by LC, points to a mean path error delay of -0.025, with a standard deviation of 0.015. A similar approach was taken to evaluate the delay error of individual cells. For this analysis, shown in Figure 4.15(b), delay errors were grouped by logic cell function and binned using the same criteria as before. In other words, each value accounted in the histogram is the average error for a given logic function. The results indicate a mean cell delay error of -0.026, with a standard deviation of 0.043. When compared to the reference library, the mean error measured for both path and cell delay are under the 5% error limit established for this work. The statistical data, therefore, attests the accuracy of the characterization environment tuned for nominal voltage.




(a) Path delay error

(b) Cell delay error

Figure 4.15 – Error histograms comparing the reduced cell library characterization at nominal voltage to the reference library. Histogram (a) shows a mean path delay error of -0.025, with a standard deviation of 0.015. Histogram (b) indicates a mean cell delay error of -0.026, with a standard deviation of 0.043.

The second step of the MVC flow takes as input the characterization environment validated for the nominal voltage. In this step, voltage-dependent parameters are scaled according to the method defined in Section 4.2.2. Table 4.3 illustrates the results of the scaling process, which is based on cross-multiplication interpolation. Each row of the table determines the input slew vector for the characterization environment of a given supply voltage. As previously mentioned, some standard cells from this library cannot operate correctly when subjected to supply voltages below 250mV. For this reason, only voltages of 250mV or above are considered. An inverter with the maximum drive strength was used as test

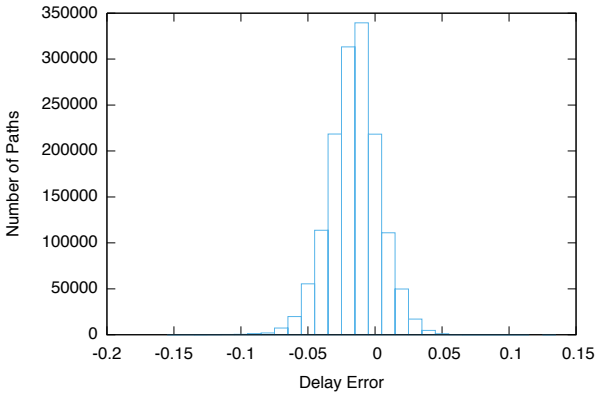
cell to determine the minimum input slew for each voltage level. Maximum input slews were scaled using a 4-input NOR gate. Even though this gate is not present in the reduced cell library, it was employed in this analysis because it presents the slowest transition delay of the original library. This step resulted in 15 unique simulation setup files, each targeting a supply voltages in the range from 950mV to 250mV. These files are based on the simulation setup file tuned for nominal voltage, with the input slew vectors for nominal voltage replaced by the scaled ones.

Table 4.3 – Scaled input slew vectors for reduced cell library obtained through cross-multiplication. Reference values (1000mV) obtained from characterization environment validated at the nominal voltage.

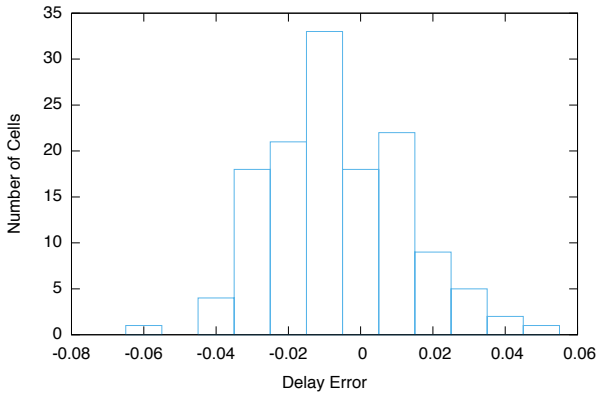
| | | **Input Slew (s)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Supply Voltage (mV)** | **250** | 8.46E-09 | 6.56E-08 | 1.27E-07 | 2.51E-07 | 5.02E-07 | 9.65E-07 | 1.93E-06 | 3.86E-06 |
| | **300** | 2.40E-09 | 1.94E-08 | 3.77E-08 | 7.42E-08 | 1.48E-07 | 2.85E-07 | 5.71E-07 | 1.14E-06 |
| | **350** | 7.06E-10 | 5.88E-09 | 1.14E-08 | 2.25E-08 | 4.49E-08 | 8.64E-08 | 1.73E-07 | 3.46E-07 |
| | **400** | 2.21E-10 | 1.87E-09 | 3.63E-09 | 7.16E-09 | 1.43E-08 | 2.75E-08 | 5.51E-08 | 1.10E-07 |
| | **450** | 8.00E-11 | 6.72E-10 | 1.30E-09 | 2.57E-09 | 5.14E-09 | 9.89E-09 | 1.98E-08 | 3.95E-08 |
| | **500** | 3.56E-11 | 2.90E-10 | 5.63E-10 | 1.11E-09 | 2.22E-09 | 4.26E-09 | 8.53E-09 | 1.71E-08 |
| | **550** | 1.93E-11 | 1.51E-10 | 2.93E-10 | 5.78E-10 | 1.16E-09 | 2.22E-09 | 4.44E-09 | 8.89E-09 |
| | **600** | 1.22E-11 | 9.14E-11 | 1.77E-10 | 3.49E-10 | 6.99E-10 | 1.34E-09 | 2.69E-09 | 5.37E-09 |
| | **650** | 8.70E-12 | 6.18E-11 | 1.20E-10 | 2.36E-10 | 4.73E-10 | 9.09E-10 | 1.82E-09 | 3.64E-09 |
| | **700** | 6.65E-12 | 4.53E-11 | 8.80E-11 | 1.73E-10 | 3.47E-10 | 6.67E-10 | 1.33E-09 | 2.67E-09 |
| | **750** | 5.39E-12 | 3.54E-11 | 6.87E-11 | 1.35E-10 | 2.71E-10 | 5.21E-10 | 1.04E-09 | 2.08E-09 |
| | **800** | 4.57E-12 | 2.88E-11 | 5.60E-11 | 1.10E-10 | 2.21E-10 | 4.24E-10 | 8.48E-10 | 1.70E-09 |
| | **850** | 3.99E-12 | 2.44E-11 | 4.73E-11 | 9.31E-11 | 1.86E-10 | 3.58E-10 | 7.16E-10 | 1.43E-09 |
| | **900** | 3.57E-12 | 2.12E-11 | 4.11E-11 | 8.09E-11 | 1.62E-10 | 3.11E-10 | 6.22E-10 | 1.24E-09 |
| | **950** | 3.25E-12 | 1.88E-11 | 3.65E-11 | 7.19E-11 | 1.44E-10 | 2.76E-10 | 5.53E-10 | 1.11E-09 |
| | **1000** | 3.00E-12 | 1.70E-11 | 3.30E-11 | 6.50E-11 | 1.30E-10 | 2.50E-10 | 5.00E-10 | 1.00E-09 |

The last step of the MVC flow consists in performing library characterizations employing the simulation setup files created in the previous step. This way, the outcome of the flow is a set of Liberty libraries that extend the voltage corners supported by the reduced cell library to the range of 1V to 250mV. In addition to providing characterization data at the nominal voltage, the ST-Microelectronics 28nm FDSOI CORE library includes timing models characterized at 900mV and 800mV. These models were used as additional verification points for the reduced cell library characterization. The approach taken is similar to what was previously done to validate the characterization environment at the nominal voltage. Modified versions of the 900mV and 800mV ST-Microelectronics 28nm FDSOI CORE libraries, containing only the set of gates that are part of the reduced library, were created and used

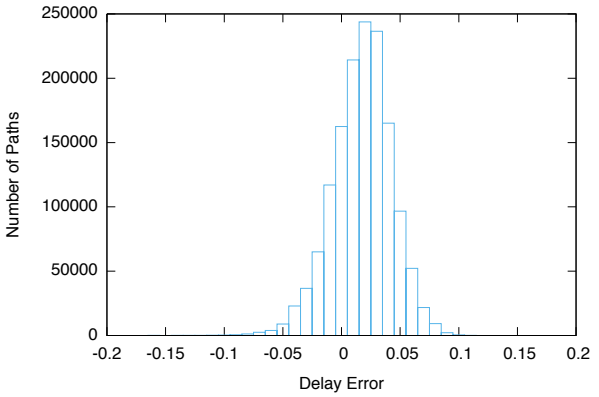
as the reference libary for LC. Figure 4.16 depictcs the results. The 900mV library presented a mean path delay error is -0.010, with a standard deviation of 0.019, and a mean cell delay error is -0.010, with a standard deviation of 0.045. At 800mV, LC points to a mean path error delay of 0.022, with a standard deviation of 0.025, and a mean cell delay error is 0.025, with a standard deviation of 0.053. Since the statistical data displays mean path and cell delay errors within the 5% error limit established for this work, we can attest the accuracy of the timing characterizations performed by the MVC flow.



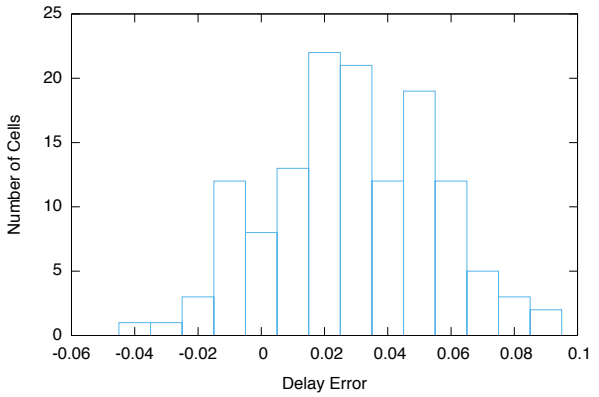

(a) Path delay error at 900mV




(b) Cell delay error at 900mV




(c) Path delay error at 800mV




(d) Cell delay error at 800mV

Figure 4.16 – Error histograms comparing the 800mV and 900mV characterizations of the reduced cell library to the reference. Histograms (a) and (b) show, respectively, a mean path delay error is -0.010, with a standard deviation of 0.019, and a mean cell delay error is -0.010, with a standard deviation of 0.045 for the library characterized at 900mV. Histograms (c) and (d) indicates, respectively, a mean path error delay of 0.022, with a standard deviation of 0.025, and a mean cell delay error is 0.025, with a standard deviation of 0.053 for the library characterized at 800mV.

# 5. TIMING ANALYSIS OF CIRCUITS UNDER VOLTAGE SCALING

Digital circuits are an abstraction of electronic devices where voltage levels at circuit nodes are quantized and interpreted as logic levels [EM12]. As previously discussed in Section 4.1, voltage levels above $V_{sh}$ represent logic high, and voltage levels below $V_{sl}$ denote logic low. During regular circuit operation, voltage levels outside these ranges only appear for a short period of time while signals are being toggled and, thus, do not have a logical meaning. The ST-Microelectronics 28nm FDSOI standard cell library, which is the basis for this work, defines $V_{sl}$ as 20% of the supply voltage of the circuit ($V_{supply}$), and $V_{sh}$ as 80% of $V_{supply}$.

During typical operation, digital circuits are subject to many different input vectors. The propagation of an input vector through digital logic leads to signal transitions at the input pins of standard cells. Each input transition causes transistors to turn on or off, what may cause the gate output node to charge or discharge (i.e. to change logic level), depending on the state of the other inputs. The time required to charge or discharge this node (i.e. $d_{cell}$, see Section 4.1.1) determines how fast a circuit can operate. Assuming a simplified model of a static CMOS gate, the time required to charge or discharge a node depends mainly on two circuit parameters: *i)* the node capacitance; and *ii)* the driving strength of the gate transistors (i.e. how much current they can drive). For instance, consider the CMOS inverter depicted in Figure 5.1. Parameter *i* is represented by the load capacitance $C_L$, which combines the input capacitances of the gates driven by the inverter plus parasitic and interconnect capacitances. The second parameter is represented by the current driven by the PMOS and NMOS transistors – $I_{PMOS}$ and $I_{NMOS}$, respectively.

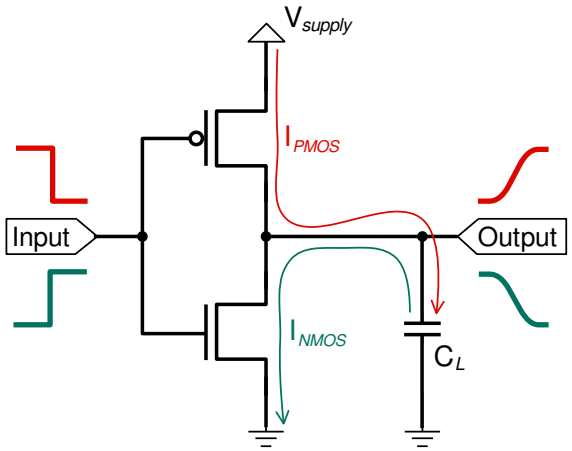



Figure 5.1 – Simplified model of CMOS inverter illustrating the charging and discharging of the gate output node.

Figure 5.1 illustrates two scenarios of operation: a rising output transition, caused by a high-to-low input transition (depicted in red), and a falling output transition, due to a low-to-high input transition (in green). In the first scenario, the time required to charge

the capacitance $C_L$ (i.e. the output node) is determined by the amount of current flowing through the PMOS transistor ($I_{PMOS}$). Likewise, in the second scenario, the time to discharge $C_L$ depends on the current driven by the NMOS transistor ($I_{NMOS}$).

Assuming balanced transistors (i.e. $I_{PMOS} = I_{NMOS} = I_d$), the delay of a CMOS gate can be modeled by Equation 5.1 [HSSB08], where $C_L$ is the load capacitance (parameter *i*), $I_d$ is the transistors' drain current (parameter *ii*), $k_d$ is a fitting parameter, and $V_{supply}$ is the supply voltage of the gate. Since $k_d$ is constant, and assuming $C_L$ to be an invariable physical parameter, the delay of a CMOS gate can be modeled as a function of $I_d$ and $V_{supply}$. When operating at sub-threshold voltages (i.e. $V_{supply} < V_{th}$), $I_d$ is exponentially inversely proportional to $V_{supply}$ – thus, making the delay of a CMOS gate at sub-threshold voltages ($d_{gateSUB}$) exponentially inversely proportional to $V_{supply}$. This is depicted in Equation 5.2, where $V_{th}$ is the transistor threshold voltage, $m$ is the sub-threshold slope factor (refer to [HSSB08]), and $I_o$ is the transistor current at the threshold voltage. In super-threshold operation (i.e. $V_{supply} > V_{th}$), the drain current of a saturated transistor (disregarding channel-length modulation effects) is proportional to the square of $V_{supply}$ – making the gate delay in this scenario ($d_{gateSUPER}$) inversely proportional to the supply voltage. This is shown in Equation 5.3 [WH10], where $k$ represents physical characteristics of the transistor and $V_{th}$ is the transistor threshold voltage. In both sub- and super-threshold regimes, the supply voltage has an inverse relationship to the CMOS gate delay. In other words, lower supply voltages lead to larger gate delays – thus, making digital circuits operate slower, due to the increase in propagation delay.

$$d_{gate} = \frac{k_d . C_L . V_{supply}}{I_d} \tag{5.1}$$

$$d_{gateSUB} = \frac{k_d . C_L . V_{supply}}{I_{dSUB}} = \frac{k_d . C_L . V_{supply}}{I_o . e^{\frac{V_{supply} - V_{th}}{m * v_T}}} \tag{5.2}$$

$$d_{gateSUPER} = \frac{k_d . C_L . V_{supply}}{I_{dSUPER}} = \frac{k_d . C_L . V_{supply}}{k . (V_{supply} - V_{th})^2} \tag{5.3}$$

The above described voltage/delay relationship can be verified in Figure 5.2, which plots the normalized delay of a CMOS inverter as a function of the supply voltage. The delay values were obtained through simulation of the SPICE deck previously depicted in Figure 4.12 and normalized to the delay at nominal voltage (1000mV). Note the exponential increase in delay for supply voltages below 500mV, as suggested by Equation 5.2.

The previous discussion reached a widely known conclusion: gate delays become slower as supply voltage reduces [HSSB08, RCN08, ZHA15, WC05, KRV+08]. However, what is the impact of this phenomenon on the design resilient circuits? To keep overheads low, only some combinational paths of resilient circuits are selected to implement error detection capabilities. Since this selection is usually based on how critical each path

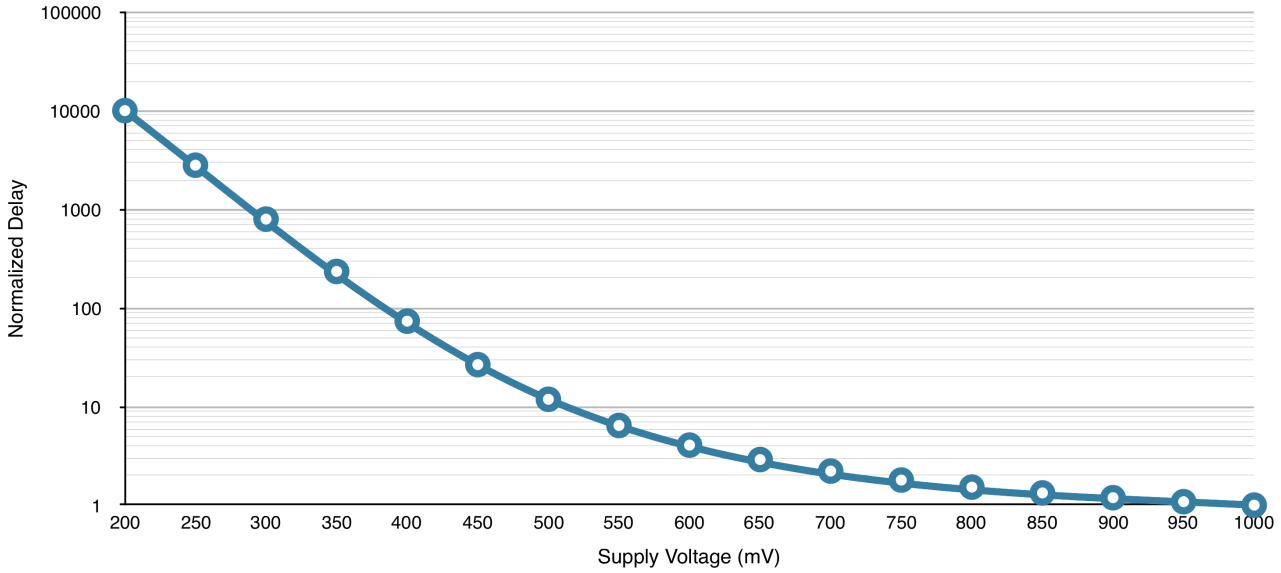



Figure 5.2 – Plot of the normalized delay of a CMOS static inverter as a function of the supply voltage. Note the large increase in delay, specially for voltages below 500mV.

is [EKD+03, HMH+15], changes in path criticality potentially have huge implications on the design of resilient circuits. In this context, this Chapter aims to provide an initial study on how logic paths of digital circuits behave under voltage scaling, focusing on the analysis of path fluctuations (i.e. change of path criticality) and its impact in resilient circuit design.

The remaining of this Chapter is organized as follows. Section 5.1 presents related work in timing analysis of circuits under voltage scaling. The environment used in the analyses presented in this Chapter is described in Section 5.2. Section 5.3 discusses the behavior of logic paths under voltage scaling. The effects of supply voltage reduction on resilient circuits is explored in Section 5.4. Finally, Section 5.5 provides an overview on the voltage scaling effects in the design of resilient circuits.

## 5.1 Related Work

This Section explores previous works that proposed to investigate the effects of voltage scaling on digital circuit timing.

Elgebaly and Sachdev [ES04] evaluate the impact of interconnect delay and process variation on the identification of critical paths in digital circuits. The work shows that the critical path of a digital circuit can change depending on the process corner and the ratio of interconnect/logic delay. The analysis was conducted for a 130µm CMOS technology. To cope with that behavior, an on-chip critical path emulator to track the changing critical path was proposed. The emulator has a voltage scaling behavior similar to the actual critical path under all investigated conditions. The architecture comprises two delay lines: one to emulate

the interconnect delay, and another to imitate the logic delay. An A/D converter determines the actual circuit delay. This information is fed to a set of look-up tables to select the delays lines that appropriately match the critical path. The analysis discussed in this work, however, is nonetheless restricted to super-threshold supply voltages only.

Kahng et al. [KKKS10] show that modules from an OpenSPARC T1 processor fail at different rates as voltage scales. Considering this behavior, the authors propose an optimization technique to redistribute the slack of modules that fail earlier, as a way to increase the range of voltage scaling supported by the design. The technique consists in swapping all cells from a given logic path with faster cells of the same functionality, to increase the timing slack. Assuming that designs are implemented using a resilient architecture, this technique can increase the range of voltages over which the circuit error rate is acceptable.

In [CWL+14], Chen et al. state that, when operating under a sub-threshold regime, small changes in temperature or transistor threshold voltage can result in exponential variations in delay. Taking this into account, the paper proposes a dynamic voltage scaling scheme that accounts for variations in temperature and process to determine the ideal supply voltage for a circuit, considering a target error rate. This technique uses the timing information obtained from a critical path monitor to capture the dependency between temperature, process and critical path slack. The values read from the monitor drive the decisions to increase or decrease the supply voltage. The authors mention that the delay value measured by the critical path monitor should be equal to or larger than the critical path in the chip. Therefore, the assumption is that the critical path of a circuit does not change as supply voltage scales.

Among the previously presented related work, only [CWL+14] analyzes digital circuits operating under a sub-threshold regime. In that paper, a dynamic voltage scaling technique that tracks the delay variations of the critical path is proposed. The technique assumes that the critical path of a circuit does not change as supply voltage scales. However, according to [ES04], this assumption does not always hold true. In fact, the authors of [ES04] showed that the critical path of a digital circuit can change, due to process variability and voltage scaling. Still, the analysis in [ES04] only covers super-threshold voltage levels. Further investigation about the fluctuation of critical paths when operating in near- and sub-threshold voltage levels is, therefore, one of the objectives of this work. In addition, none of the aforementioned papers investigate how voltage scaling affects the design of resilient architectures.

## 5.2 Analysis Environment

Path Analyzer (PA) is an automated environment designed by the Author to compute changes in path delays, considering a number of voltage corners. The proposition of

a custom environment for this was necessary to enable the analysis of logic paths under voltage scaling. As Figure 5.3 shows, PA works by extracting logic path delays from a set of benchmark netlists, using a number of libraries characterized at different voltage corners. It relies on Synopsys PrimeTime to extract path delays and on two in-house tools to parse (Timing Report Parser) and analyze (Timing Analyzer) the timing data. To ensure that each netlist is thoroughly analyzed, PrimeTime generates unique timing reports for each path that connects a register output (or primary input) to a register input (or primary output) without any intervening registers – thus guaranteeing that data from all timing paths are acquired. Both the minimum and the maximum propagation delays are extracted for each path. The path delay information for each combination of netlist and library is stored on a individual file. The Timing Report Parser reads these files and generates a database for each netlist, combining the path delay information of all voltage corners. These databases are processed by the Timing Analyzer to generate the analysis reports used throughout the remaining of this Chapter.

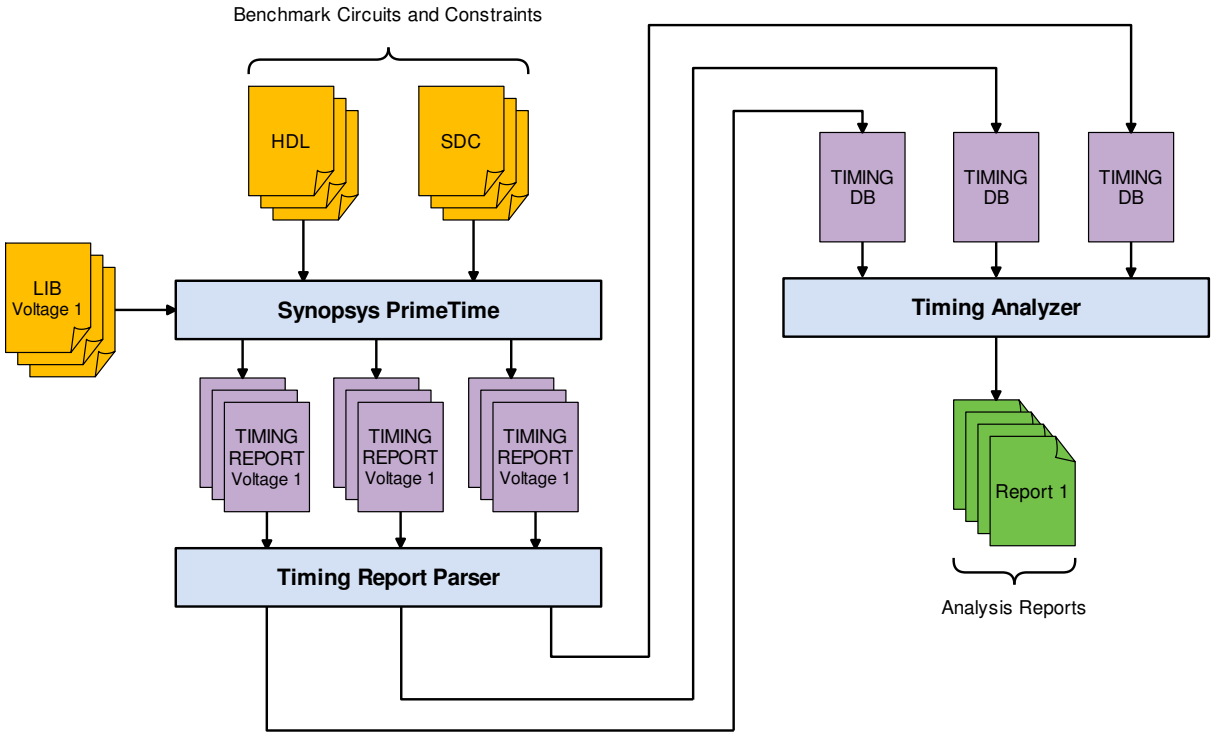



Figure 5.3 – Path Analyzer (PA), an automated environment used to compute the changes of path delays, considering a number a number of voltage corners. Synopsys PrimeTime performs STA to extract all path delays from a set of benchmark circuits. Reports are processed by the Timing Report Parser tool, which outputs a timing database for each circuit. These databases are processed by the Timing Analyzer tool to generate the reports employed on the analyses discussed in this Chapter.

Data employed on the analyses contained in the following Sections of this Chapter were extracted using the set of benchmark circuits previously detailed in Section 3.1.1. This set of circuits contains both combinational and synchronous sequential circuits. Since the studies discussed in this Chapter focus on the analysis of combinational logic path delays,

the timing information associated to sequential cells is disregarded. The set of 58 benchmarks was synthesized at nominal voltage and the resulting netlists were used to evaluate all target voltage corners. This simulates a real world circuit implementation: a single manufactured chip, optimized for one PVT corner subjected to a range of supply voltages. The circuits were analyzed using the libraries generated through the multi-voltage characterization flow (Section 4.3) of the reduced cell library (Section 3.3). In this way, this study covers supply voltages ranging from 1V down to 250mV.

## 5.3 Behavior of Logic Paths Under Voltage Scaling

This Section provides insight on the behaviour of logic paths under voltage scaling. The goal is to evaluate the impact of supply voltage reduction on the fluctuation of logic paths. In other words, this study investigates at which level the criticality of logic paths changes as voltage scales. To do so, the PA environment detailed in Section 5.2 was used to extract each and every logic paths delay from all 58 benchmark circuits when subjected to each one of the 16 supply voltages corners targeted in this analysis. This resulted on the extraction of about 220,000 paths delays for each target voltage corner. The proposed method for the voltage scaling analysis is the comparison of delay variations between voltage corners in steps of 50mV. This way, for the remaining of this work, a *voltage scaling step* is defined as a 50mV reduction on supply voltage. The *delay variation* of a path is determined by the expression in Equation 5.4, where $path_A$ is the path name and $V_A, V_B$ are the initial and final supply voltages for a given voltage scaling step, respectively. In this Equation, $delay(path_A, V_A)$ is the delay of path $path_A$ on the voltage corner $V_A$. Thus, the delay variation measures the increase in logic path delay resulting from the supply voltage reduction.

$$delay\_variation(path_A, V_B) = \frac{delay(path_A, V_B)}{delay(path_A, V_A)} \text{ , where } V_B = V_A - 50mV \tag{5.4}$$

The analysis of path fluctuation is based on the comparison between the delay variation of each path and the global variation (i.e. the average delay variation among all paths), considering each voltage scaling step. The percentage of paths that exhibit delay variations above average convey a notion of the amount of paths that will likely fluctuate. In other words, since the delay of these paths grow at a rate that is higher than the average, they are inclined become slower than other paths and can potentially become the (new) critical path. The magnitude of delay variations expresses the amount of delay increase that a given group of paths is expected to present. This metric conveys a notion of how much impact the delay variation causes. For example, logic paths that present delay variations slightly above average have very little impact on the overall path fluctuation and can be

disregarded – that is, if these paths become slower than other paths, it will be by a negligible amount. Hence, paths that present delay variations of up to and including 1% above the average are disregarded from the analysis. Therefore, combining the percentage of paths with the magnitude of delay variation provides a comprehensive metric to evaluate path fluctuations.

Figure 5.4 shows the magnitude distribution for logic path fluctuations, considering a voltage range from 1000mV to 250mV, with 50mV scaling steps. Horizontal bars represent the percentage of logic paths that exhibit delay variations above a certain magnitude. The magnitudes are color-coded and defined as a percentage of the average path delay at each voltage scaling step (i.e. a percentage of the global variation at each voltage scaling step) – these percentages are listed in the upper right side of the Figure. For instance, consider the voltage scaling from 950mV to 900mV: the blue bar indicates that about 10% of the paths exhibit delay variations at least 1% above of the average variation for the current step; the green bar shows that around 5% of the paths present variations at least 5% above average; finally, the yellow bar indicates that 1% of the paths demonstrate delay variations at least 10% above average. The absence of bars showing variations of 25% and above suggests that the highest level of variation in this voltage scaling step is in the range from 10% to 25% above average.

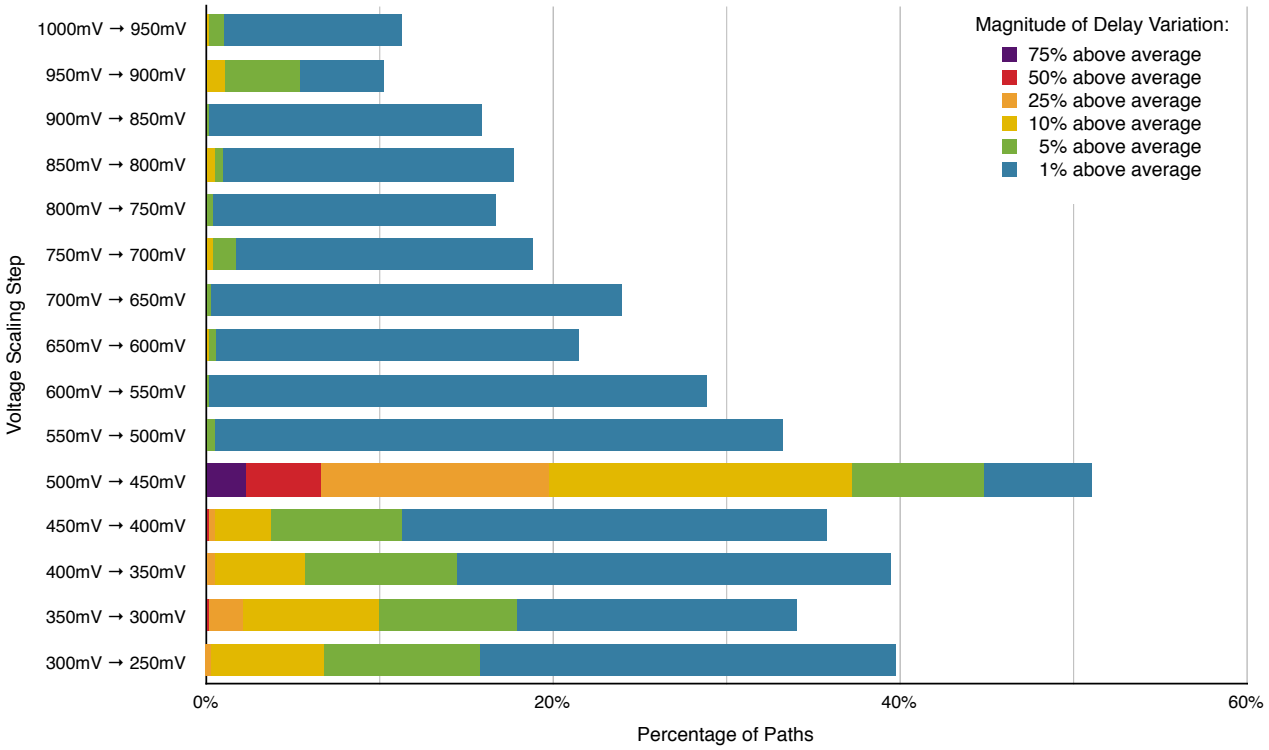



Figure 5.4 – Magnitude distribution of logic path fluctuations for a range of voltage scaling steps. Horizontal bars represent the percentage of logic paths that exhibit delay variations above a certain magnitude. The magnitudes are color-coded and defined as a percentage of the average path delay at each voltage scaling step.

Considering super-threshold voltage levels ($V_{supply} \geq 500mV$), the data presented in Figure 5.4 suggests that supply voltage reduction leads to a large increase on the percentage of paths that present delay variations of at least 1% above average. The highest variation level seen in the super-threshold is in the range from 10% to 25% above average. In most scaling steps, however, this level of variation is limited to a percentage of paths inferior to 0.15% (6 scaling steps). As a matter of fact, only the voltage scaling from 950mV to 900mV presented at least 1% of the paths with this level of delay variations. Paths that exhibit delay variations at least 5% above average are more common, though. In fact, from the 10 scaling steps is this voltage range, eight present this level of variation in at least 0.25% of the paths and six in at least 0.5%. In summary, the analysis suggests that, due to the large percentage of paths that exhibit delay variations above average, path fluctuations may take place at super-threshold voltage levels. In spite of that, since most of the delay variations seen in this voltage levels are not larger than 10% above average, these fluctuations are limited to variations of small magnitude. Nonetheless, the author considers that further investigations are still needed to assess the impact of these minor path fluctuations in the design of resilient circuits.

The voltage scaling from 500mV to 450mV demonstrates that the largest level of path fluctuation takes place when moving from super- to sub-threshold voltages. This is an expected behaviour since, as previously mentioned on the introduction of this Chapter and shown in Equations 5.2, at sub-threshold voltage levels the delay of a CMOS gate is exponentially dependent on the supply voltage. The data in Figure 5.4 shows that over 2% of the paths present delay variations at least 75% above the average. Furthermore, over 40% of the paths exhibit variations at least 5% above the average. More than half of the analyzed paths are affected by delay variations at least 1% above the average. Hence, the analysis suggests that scaling the supply voltage from super- to sub-threshold voltages results in a large degree of logic paths fluctuations, in both magnitude and number of paths.

In each voltage scaling step at sub-threshold level ($V_{supply} \leq 450mV$), over 10% of the logic paths exhibit delay variations at least 5% above the average variation, and more than 30% of the paths exhibit variations at least 1% above the average. Furthermore, up to 0.07% of the paths in the sub-threshold region present levels of variation at least 75% above average. This suggests that the sub-threshold domain presents a much larger degree of logic path fluctuations, when compared to super-threshold operation.

The study presented in this Section analyzed the magnitude distribution of logic path fluctuations for a range of voltage scaling steps. This investigation compared the variation of each logic path delay extracted from a set of circuits to the global average variation along all benchmarks. The average delay variation of a circuit depends on the logic depth as well as on the types of employed logic functions. Even though each individual circuit average delay variation pattern may depart from the global variation mean, experiments (not addressed here) showed that the overall trend remains the same: there is a correlation be-

tween the increase in path fluctuation and the reduction of supply voltage, specially when operating at sub-threshold voltage levels. Results indicate that this occurs regardless of the type of circuit design. The next Section explores the implications of logic path fluctuation on resilient architectures.

## 5.4 Voltage Scaling Effects on Resilient Architectures

One of the most impacting decisions in resilient circuit design is the choice of which paths will feature error detection capabilities. As previously discussed in Section 2.5, this design choice not only impacts the area overhead due to error detection hardware, but also determines the conditions in which the circuit can operate correctly. In other words, resilient circuits can only detect and correct timing violations that occur on paths that are specifically designed with these capabilities (i.e. resilient paths). Therefore, to ensure reliable and error-free operation, resilient architectures can only operate under conditions that guarantee timing correctness of all non-resilient paths. If this condition is not met, the correctness of the whole system is compromised, as errors due to undetected and/or undetectable timing violations could propagate through the circuit.

The resilient paths of a circuit are usually selected based on the target timing resiliency window (TRW) for the design. The TRW, which is defined as a percentage of the worst-case delay of a circuit ($d_{worst}$), determines the window of time where errors can gracefully take place. Logic paths that exhibit delay above $d_{worst} * (1 - TRW)$ are *inside the TRW* and, therefore, need to feature error detection capabilities. For instance, consider the 30% TRW illustrated in Figure 5.5. In this scenario, the logic paths with delay grater than 70% of the worst-case delay are inside the TRW.

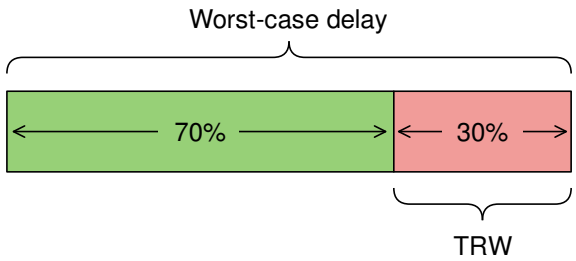



Figure 5.5 – Representation of a timing resilience window (TRW) set to 30% of the worst-case delay of a circuit. Paths that exhibit delay above 70% of the worst-case delay are said to be *inside the TRW*.

In summary, the selection of resilient paths is associated with the timing characteristics of the circuit. Consequently, when designing a resilient circuit to operate under voltage scaling, the impact of logic path fluctuation, shown in the previous Section, must be accounted for, to guarantee correct operation for the system. This Section investigates the

effects of voltage scaling in the design of resilient circuits, focusing on understanding the impact of supply voltage reduction on the selection of resilient paths.

The previous Section suggested a correlation between supply voltage reduction and the increase in path fluctuations. To examine how this correlation affects the number of paths inside the TRW, the average percentage of resilient paths ($\overline{PRP}$) was computed with respect to the following TRWs: 5%, 10%, 20%, 30%, 50% and 75%. The $\overline{PRP}$ expresses the overall ratio between resilient paths and the total number of logic paths found in the set of benchmark circuits employed in this analysis. Apart from representing the number of resilient paths in a circuit, this metric acts as an estimate of the area overhead cost due to error detection hardware – i.e. a larger $\overline{PRP}$ suggests a higher cost. The $\overline{PRP}$ is calculated by Equation 5.5, where $V$ is the voltage corner of the analysis, $TRW$ is the timing resilience window, $n$ is the number of circuits analyzed and $PRP(V, TRW, k)$ is the percentage of resilient paths of circuit $k$, given a voltage corner $V$ and the $TRW$. The percentage of resilient paths ($PRP$) for a given circuit is determined by Equation 5.6, where $number\_paths(k)$ is the number of logic paths found in circuit $k$ and $number\_resilient\_paths(k, V, TRW)$ represents the number of resilient paths in circuit $k$, given a voltage corner $V$ and the TRW. The number of resilient paths is determined based on the delay information extracted by the PA environment.

$$\overline{PRP}(V, TRW) = \frac{\sum_{k=1}^{n} PRP(V, TRW, k)}{n} \tag{5.5}$$

$$PRP(V, TRW, k) = \frac{number\_resilient\_paths(k, V, TRW)}{number\_paths(k)} \tag{5.6}$$

Figure 5.6 shows the $\overline{PRP}$ for each voltage corner and TRW analyzed. As expected, the results point to the existence of a direct relationship between $\overline{PRP}$ and the TRW size – that is, larger TRWs result in a larger number of resilient paths. In addition, significant differences in $\overline{PRP}$ can be seen when comparing the overall percentage of resilient paths at super- and sub-threshold voltages. For instance, considering a 50% TRW, the overall $\overline{PRP}$ is about 65% for supply voltages above 500mV and in the order of 40% for voltage levels below 450mV. This behaviour, which is seen in each of the analyzed TRWs, suggests that resilient circuits designed to only operate at sub-threshold levels could present smaller area overheads due to error detection logic, when compared to circuits designed to work at super-threshold supplies. Further investigation in necessary to thoroughly understand the reason for this behaviour. Nevertheless, considering the exponential increase in propagation delays at sub-threshold supplies [WC05] and the large magnitude of logic path fluctuations seen for supplies below 500mV in the analysis presented in Section 5.3, this behaviour could be explained by the worst-case delay of the circuit increasing at a significantly higher rate than the remaining delays. Since the delay range of the TRW is a function of the worst-case de-

lay of the circuit, resilient paths that present a low rate of delay increase could be eventually excluded from the TRW.

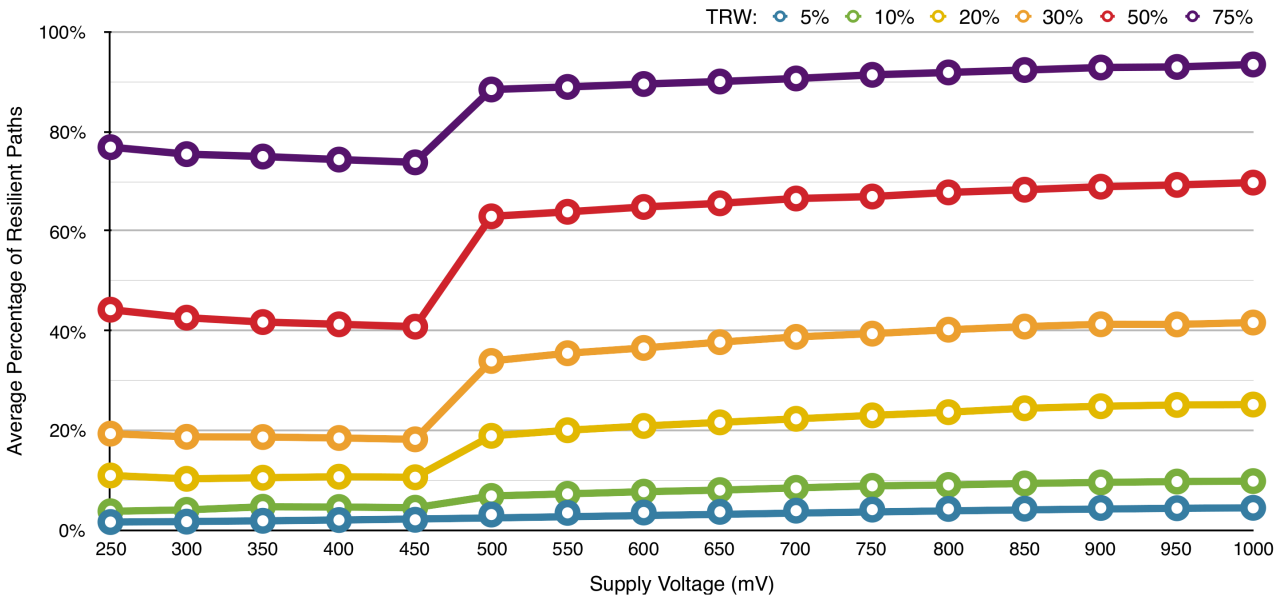



Figure 5.6 – Average percentage of resilient paths ($\overline{PRP}$) for each TRW and voltage corner analyzed.

The $\overline{PRP}$ results presented in Figure 5.6 convey a notion of the expected number of resilient paths in resilient circuits designed to operate under a specific supply voltage corner and target TRW. These results show that the cost of a resilient circuit in terms of error detection hardware is determined not only by the desired TRW but also by the target supply voltage of the circuit. Therefore, to evaluate the effects of voltage scaling on resilient paths, the average percentage of resilient paths metric was extended to support a range of supply voltages. The $\overline{PRP}_{VS}$ represents the average percentage of resilient paths in a resilient circuit designed to operate under a given range of voltage scaling. The range of voltage scaling is defined by the highest ($V_i$, where $i$ stands for initial) and the lowest ($V_f$, where $f$ stands for final) voltage corners supported by the design. As an example, a circuit designed to support voltage scaling from 800mV to 450mV is defined by a $V_i$ of 800mV and a $V_f$ of 450mV. The metric $\overline{PRP}_{VS}$ is calculated by the expression in Equation 5.7, where $V_i, V_f$ are the initial and final voltage corners supported by the circuit, $TRW$ is the timing resilience window, $n$ is the number of circuits analyzed and $PRP_{VS}(V_i, V_f, TRW, k)$ is the percentage of resilient paths in circuit $k$, given a voltage scaling range from $V_i$ to $V_f$ and the $TRW$. The percentage of resilient paths for a given circuit supporting voltage scaling ($PRP_{VS}$) is determined as stated in Equation 5.8, where $number\_paths(k)$ is the number of logic paths found in circuit $k$ and $number\_resilient\_paths(k, V_i, V_f, TRW)$ represents the number of resilient paths in circuit $k$, given a voltage range from $V_i$ to $V_f$ and the TRW. When considering voltage scaling support, the number of resilient paths in a circuit is determined by the amount of paths necessary to guarantee that the target TRW is reachable in any voltage corner belonging to the voltage scaling range. In other words, consider $R(k, TRW, V)$ to be

the set of resilient paths for circuit k, for a given a TRW and voltage corner $V$. The number of resilient paths in circuit $k$ for a given TRW and a voltage range from $V_i$ to $V_f$ is determined by the expression in Equation 5.9, where $|x|$ is the cardinality of the set $x$ and $\bigcup_{v=V_i}^{V_f}$ is the set union operation iterating over the voltage range. The set of resilient paths for a voltage corner is determined based on the delay information extracted by the PA environment.

$$\overline{PRP}_{VS}(V_i, V_f, TRW) = \frac{\sum_{k=1}^{n} PRP_{VS}(V_i, V_f, TRW, k)}{n} \tag{5.7}$$

$$PRP_{VS}(V_i, V_f, TRW, k) = \frac{number\_resilient\_paths(k, V_i, V_f, TRW)}{number\_paths(k)} \tag{5.8}$$

$$number\_resilient\_paths(k, V_i, V_f, TRW) = |\bigcup_{v=V_i}^{V_f} R(k, TRW, v)| \tag{5.9}$$

Table 5.1 shows the $\overline{PRP}_{VS}$ results for all voltage scaling ranges between 1V and 250mV in 50mV steps, considering a TRW of 10%. As previously mentioned, the range of voltage scaling is defined by the highest and lowest supply voltages supported by the design. The highest voltage, referred as the *initial supply voltage*, is determined by the columns of the table. Likewise, the lowest voltage supported by the design, called *final supply voltage*, is specified by the rows of the table. For instance, a resilient circuit designed to support voltage scaling from 750mV (column) to 550mV (row) presents a 9.6% $\overline{PRP}_{VS}$. To ease visualization of the data, each cell is colored according to its $\overline{PRP}_{VS}$ value – red denote large values and green indicate small values.

The first analysis presented in this Section suggested that resilient circuits designed to operate at sub-threshold levels exhibit smaller overheads due to error detection hardware, when compared to circuits designed for the super-threshold. The same conclusion can be drawn from the data presented in Table 5.1. Overall, the error detection cost for circuits designed to operate only at the sub-threshold region (i.e. $V_i > 500mV$, shown in green) is over 50% smaller than for circuits that operate at the super-threshold region. This indicates that resilient architectures can be a viable approach to increase the performance of ultra-low power circuits with reasonable error detection overheads.

The analysis of individual columns in Table 5.1 reveals that increasing the range of voltage scaling supported by a resilient circuit leads to a rise in $\overline{PRP}_{VS}$. This is an expected outcome, given the logic path fluctuations associated to each voltage scaling step, as previously discussed in Section 5.3 and illustrated in Figure 5.4. As a matter of fact, the behaviour regarding logic path fluctuations in the super- and sub-threshold regions observed in the aforementioned Section can also be noticed by examining the magnitude of

Table 5.1 – Average percentage of resilient paths ($\overline{PRP_{VS}}$) for a given range of voltage scaling, considering a 10% TRW. The Table shows the results for all possible ranges of voltage scaling from 1V to 250mV in 50mV steps. Each cell is colored according to the $\overline{PRP_{VS}}$ magnitude – red denote large values and green indicate small values.

| Final Supply Voltage (mV) \ Initial Supply Voltage (mV) | 1000 | 950 | 900 | 850 | 800 | 750 | 700 | 650 | 600 | 550 | 500 | 450 | 400 | 350 | 300 | 250 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 13.3% | 13.1% | 12.8% | 12.6% | 12.3% | 12.0% | 11.5% | 11.1% | 10.6% | 10.2% | 9.8% | 5.9% | 5.6% | 5.4% | 4.8% | 3.9% |
| 300 | 13.3% | 13.1% | 12.8% | 12.5% | 12.2% | 11.9% | 11.5% | 11.1% | 10.5% | 10.1% | 9.7% | 5.7% | 5.5% | 5.2% | 4.2% | |
| 350 | 13.2% | 13.0% | 12.7% | 12.5% | 12.2% | 11.9% | 11.4% | 11.0% | 10.5% | 10.0% | 9.6% | 5.5% | 5.3% | 4.8% | | |
| 400 | 13.0% | 12.8% | 12.5% | 12.3% | 12.0% | 11.7% | 11.2% | 10.8% | 10.3% | 9.8% | 9.4% | 5.1% | 4.7% | | | |
| 450 | 12.9% | 12.7% | 12.4% | 12.1% | 11.8% | 11.6% | 11.1% | 10.7% | 10.1% | 9.7% | 9.3% | 4.6% | | | | |
| 500 | 11.2% | 11.0% | 10.7% | 10.4% | 10.1% | 9.7% | 9.2% | 8.7% | 8.1% | 7.5% | 7.0% | | | | | |
| 550 | 11.1% | 10.9% | 10.6% | 10.3% | 9.9% | 9.6% | 9.1% | 8.6% | 8.0% | 7.4% | | | | | | |
| 600 | 11.0% | 10.8% | 10.5% | 10.1% | 9.8% | 9.5% | 9.0% | 8.4% | 7.8% | | | | | | | |
| 650 | 10.7% | 10.5% | 10.1% | 9.8% | 9.5% | 9.2% | 8.6% | 8.1% | | | | | | | | |
| 700 | 10.6% | 10.4% | 10.1% | 9.8% | 9.4% | 9.1% | 8.6% | | | | | | | | | |
| 750 | 10.6% | 10.3% | 10.0% | 9.7% | 9.3% | 9.0% | | | | | | | | | | |
| 800 | 10.4% | 10.2% | 9.9% | 9.5% | 9.2% | | | | | | | | | | | |
| 850 | 10.4% | 10.1% | 9.8% | 9.5% | | | | | | | | | | | | |
| 900 | 10.3% | 10.0% | 9.7% | | | | | | | | | | | | | |
| 950 | 10.1% | 9.8% | | | | | | | | | | | | | | |
| 1000 | 9.9% | | | | | | | | | | | | | | | |

the changes in $\overline{PRP_{VS}}$ in the same regions of Table 5.1. For instance, consider the super-threshold region – i. e. initial and final supply voltages in the range from 1000mV to 500mV. Each additional voltage scaling step in this region results in an average $\overline{PRP_{VS}}$ change of 0.14 percentage point – no changes larger than 0.4 are seen. As an example, the $\overline{PRP_{VS}}$ for a resilient circuit operating in the voltage scaling range from 900mV to 750mV is 10.0%. Increasing the voltage scaling range of this circuit in one step – i.e. allowing it to operate from 900mV to 700mV – results in a $\overline{PRP_{VS}}$ of 10.1% – that is, a 0.1 percentage point increase when compared to the previous voltage scaling range. This behaviour is consistent with the one observed in the previous Section, which leads to the following conclusion: super-threshold fluctuations are limited to variations of small magnitude; thus, the overhead cost to increase the voltage scaling range of resilient circuits operating in this voltage levels is relatively small. However, in the sub-threshold region (i. e. initial and final supply voltages below 500mV), the average $\overline{PRP_{VS}}$ increase due to one additional voltage scaling step is 0.34 percentage point, with changes no larger than 0.6 percentage point. The behaviour seen in this region also matches the observations of Section 5.3, suggesting sub-threshold operation presents a much larger degree of logic path fluctuations, when compared to the

super-threshold region. This way, the resiliency cost to increase one voltage scaling step in purely sub-threshold circuits is 2.5 times larger than the cost seen in super-threshold implementations. Finally, the largest levels of resiliency cost are seen when extending the voltage scaling range across the threshold barrier – i.e. extending the final supply voltage level from 500mV to 450mV. In this scenario, the average change in $\overline{PRP}_{VS}$ is 1.9 percentage points – a value that is over 13 times larger than the average $\overline{PRP}_{VS}$ increase for an additional voltage scaling step in the super-threshold region. The largest increase in resiliency is 2.3 percentage points, seen when enabling a single-voltage resilient circuit designed to operate at 500mV to reach a final supply voltage of 450mV. This behaviour is consistent with the one observed in the previous Section, suggesting that scaling the supply voltage across the threshold barrier results in the largest magnitude of logic paths fluctuations and resiliency costs. In conclusion, this analysis points to a direct correlation between logic path fluctuation and the cost of resiliency/$\overline{PRP}_{VS}$ in all regions of operation.

The previous analysis investigated the voltage scaling effects on the $\overline{PRP}_{VS}$ considering a resilient circuit designed with a TRW of 10%. Since the trends seen in the study are independent of TRW, the conclusions drawn from the previous analysis can be carried over to other values of TRW. However, it is important to note that, as seen in Figure 5.6, larger TRWs typically include more resilient paths and, thus, are less liable to the effects of logic path fluctuations. In other words, $\overline{PRP}_{VS}$ changes of smaller magnitude are expected in resilient circuits designed with larger TRWs. The $\overline{PRP}_{VS}$ results for the following TRWs are available in Appendix A: 5%, 20%, 30%,50% and 75%.

## 5.5 Effects of Voltage Scaling in the Design of Resilient Circuits

This Section provides a summary of the conclusions drawn from the analyses performed in this Chapter.

The investigation in Section 5.3 pointed to a correlation between the increase in logic path fluctuations and the reduction of supply voltage, specially when crossing the threshold barrier. The study in Section 5.4 suggested a direct relationship between logic path fluctuations and the percentage of resilient paths ($\overline{PRP}_{VS}$) in a resilient circuit. This relationship holds true for all analyzed regions of operation. The $\overline{PRP}_{VS}$ is a metric that conveys a notion of the ratio between the resilient and non-resilient paths in a resilient circuit and can thus be used to estimate the overheads due to error detection hardware addition. The analysis of this metric suggested that the cost of error detection in circuits designed to operate only at the sub-threshold region is over 50% smaller than the cost to operate in the super-threshold region.

The following discussion outlines the impact of these conclusions on the design of resilient circuits targeting super- and sub-threshold operation:

Design of resilient circuits targeting super-threshold operation

According to the results presented in this Chapter, digital circuits operating at super-threshold voltage levels exhibit logic path fluctuations of small magnitude. In the context of resilient circuit design, this translates to relatively small overhead costs to increase the voltage scaling range of a circuit up to the near-threshold region. Crossing the threshold barrier, however, results in a large overhead increase. Overall, the super-threshold operation of resilient circuits presents a large cost of resiliency when compared to circuits designed to operate at the sub-threshold region only. This suggests that, for power critical applications, increasing the range of voltage scaling supported by a resilient design up to a near-threshold voltage can present a good area/power trade-off, as this leads to large energy savings at the cost of minor area increase due to resiliency overheads.

Design of resilient circuits targeting sub-threshold operation

Resilient circuits designed to operate at sub-threshold voltages present small overheads due to error detection logic addition, when compared to circuits designed for super-threshold operation. The overhead costs to increase the voltage scaling range supported by the circuit, however, are about 2.5 times larger than the cost seen in super-threshold implementations. Nevertheless, this increase is not very impactful, as the overall resiliency overhead is still small. This suggests that resilient architectures can be a viable approach to increase the performance of ultra-low power circuits, such as battery-powered or energy-harvesting based IoT devices or wearables, with reasonable error detection overheads.

# 6. CONCLUSIONS

This work investigated the effects of voltage scaling on the design of resilient circuits. This investigation, which was based on the timing analysis of circuits under voltage scaling, was made possible by the characterization of a reduced cell library over a wide range of supply voltages. This way, the original contributions of this work can be summarized in three items: i) a method to select reduced standard cell libraries; ii) the multi-voltage characterization flow; iii) the timing analysis of digital circuits under voltage scaling.

The first contribution of this work is a method to select a subset of cells capable of achieving performance results comparable to the full library, while keeping the overheads low and maintaining the circuit functional at low supply voltages. This method was successfully used in the selection of a set of cells employed throughout the development of this work. In addition, an automated synthesis environment was designed to compare the overheads between reduced library candidates and the full library.

The second contribution is a method designed to extend the voltage corners supported by standard cell libraries. This method, which is entitled Multi-voltage Characterization (MVC) flow, provides a systematic way to characterize cell libraries to a wide range of target voltages, ensuring the proper scaling of voltage-dependent parameters. The MVC flow was used to characterize the reduced cell library from 1V to 250mV, which enabled the study of voltage scaling effects on resilient circuit design. In addition to the flow, a tool designed to validate cell characterizations by the comparison of STA reports was proposed.

Finally, the main contribution of this work is the study of how logic path fluctuations due to voltage scaling impact the design of resilient circuits. This investigation pointed to a correlation between voltage scaling, the increase in logic path fluctuations and the overhead costs due to error detection hardware addition in resilient architectures. Interestingly, the analysis suggests that resilient circuits designed to operate only at sub-threshold levels may present significantly lower error detection overheads than circuits targeting super-threshold operation.

## 6.1 Future Work

This work encompasses many topics for further research. The most immediate one is an investigation to thoroughly understand why resilient circuits designed to operate at sub-threshold levels present smaller area overheads due to error detection logic than circuits designed to work at super-threshold supply ranges. In addition, considering that bundled-data circuits such as Blade rely on the assumption that delay lines are properly matched to the worst-case path delay of the circuit at all times, another interesting research topic is

the investigation of the impact of logic path fluctuations in bundled-data design. Another interesting topic is the evaluation of impact of voltage scaling on the operation of the Blade controller. This analysis is crucial to enable the sub-threshold operation of Blade circuits.

# REFERENCES


[BBK89] Brglez, F.; Bryan, D.; Kozminski, K. “Combinational Profiles of Sequential Benchmark Circuits”. In: IEEE International Symposium on Circuits and Systems (ISCAS), 1989, pp. 1929–1934 vol.3.

[BCC+14] Beer, S.; Cannizzaro, M.; Cortadella, J.; Ginosar, R.; Lavagno, L. “Metastability in better-than-worst-case designs”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) - Fresh Ideas Workshop, 2014, pp. 101–102.

[BF85] Brglez, F.; Fujiwara, H. “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN”. In: IEEE International Symposium on Circuits and Systems (ISCAS), Special Session on ATPG and Fault Simulation, 1985, pp. 695–698.

[BOF10] Beerel, P.; Ozdag, R.; Ferretti, M. “A Designer’s Guide to Asynchronous VLSI”. Cambridge University Press, 2010.

[BTK+09] Bowman, K.; Tschanz, J.; Kim, N. S.; Lee, J.; Wilkerson, C.; Lu, S.; Karnik, T.; De, V. “Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance”, *IEEE Journal of Solid-State Circuits*, vol. 44–1, Jan 2009, pp. 49–63.

[Cad13] Cadence Design Systems, Inc. “Encounter Library Characterizer User Guide, Version 13.1”, 2013.

[CCGC13] Chang, K.-L.; Chang, J.; Gwee, B.-H.; Chong, K.-S. “Synchronous-Logic and Asynchronous-Logic 8051 Microcontroller Cores for Realizing the Internet of Things: A Comparative Study on Dynamic Voltage Scaling and Variation Effects”, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3–1, Mar 2013, pp. 23–34.

[CWL+14] Chen, Y.-G.; Wang, T.; Lai, K.-Y.; Wen, W.-Y.; Shi, Y.; Chang, S.-C. “Critical Path Monitor Enabled Dynamic Voltage Scaling for Graceful Degradation in Sub-threshold Designs”. In: 51st Design Automation Conference (DAC), 2014, pp. 1–6.

[DS89] Deng, A.; Shiau, Y. “Generic Linear RC Network Model for Digital CMOS Circuits”. In: IEEE International Symposium on Circuits and Systems (ISCAS), 1989, pp. 860–863 vol.2.

[DTP+09] Das, S.; Tokunaga, C.; Pant, S.; Ma, W.-H.; Kalaiselvan, S.; Lai, K.; Bull, D. M.; Blaauw, D. T. “Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance”, *IEEE Journal of Solid-State Circuits*, vol. 44–1, Jan 2009, pp. 32–48.

[EKD+03] Ernst, D.; Kim, N. S.; Das, S.; Pant, S.; Rao, R.; Pham, T.; Ziesler, C.; Blaauw, D.; Austin, T.; Flautner, K.; Mudge, T. “Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation”. In: Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36), 2003, pp. 7–18.

[EM12] Even, G.; Medina, M. “Digital Logic Design: A Rigorous Approach”. Cambridge University Press, 2012.

[ES04] Elgebaly, M.; Sachdev, M. “Efficient Adaptive Voltage Scaling System Through On-Chip Critical Path Emulation”. In: International Symposium on Low Power Electronics and Design (ISLPED), 2004, pp. 375–380.

[FFK+13] Fojtik, M.; Fick, D.; Kim, Y.; Pinckney, N.; Harris, D. M.; Blaauw, D.; Sylvester, D. “Bubble Razor: Eliminating Timing Margins in an ARM Cortex-M3 Processor in 45 nm CMOS Using Architecturally Independent Error Detection and Correction”, *IEEE Journal of Solid-State Circuits*, vol. 48–1, Jan 2013, pp. 66–81.

[Hau95] Hauck, S. “Asynchronous design methodologies: an overview”, *Proceedings of the IEEE*, vol. 83–1, Jan 1995, pp. 69–93.

[HHC+15] Hand, D.; Huang, H.; Cheng, B.; Zhang, Y.; Moreira, M. T.; Breuer, M.; Calazans, N. L. V.; Beerel, P. A. “Performance Optimization and Analysis of Blade Designs Under Delay Variability”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2015, pp. 61–68.

[HMH+15] Hand, D.; Moreira, M. T.; Huang, H.; Chen, D.; Butzke, F.; Li, Z.; Gibiluka, M.; Breuer, M.; Calazans, N. L. V.; Beerel, P. A. “Blade – A Timing Violation Resilient Asynchronous Template”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2015, pp. 21–28.

[HSSB08] Hanson, S.; Seok, M.; Sylvester, D.; Blaauw, D. “Nanometer Device Scaling in Subthreshold Logic and SRAM”, *IEEE Transactions on Electron Devices*, vol. 55–1, Jan 2008, pp. 175–185.

[HYH99] Hansen, M.; Yalcin, H.; Hayes, J. “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering”, *IEEE Design & Test of Computers*, vol. 16–3, Jul-Sep 1999, pp. 72–80.

[JCDGH15] Jayakrishnan, M.; Chang, A.; De Gyvez, J.; Hyoung, K. T. “Slack-aware Timing Margin Redistribution Technique Utilizing Error Avoidance Flip-Flops and Time Borrowing”. In: IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), 2015, pp. 159–164.

[KKFK13] Kim, S.; Kwon, I.; Fick, D.; Kim, M. “Razor-lite: A Side-Channel Error-Detection Register for Timing-Margin Recovery in 45nm SOI CMOS”. In: IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013, pp. 264–266.

[KKKS10] Kahng, A.; Kang, S.; Kumar, R.; Sartori, J. “Designing a Processor From the Ground Up to Allow Voltage/Reliability Tradeoffs”. In: IEEE 16th International Symposium on High Performance Computer Architecture (HPCA), 2010, pp. 1–11.

[KKLR11] Kandhalu, A.; Kim, J.; Lakshmanan, K.; Rajkumar, R. “Energy-aware partitioned fixed-priority scheduling for chip multi-processors”. In: IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011, pp. 93–102.

[KRV+08] Kwong, J.; Ramadass, Y.; Verma, N.; Koesler, M.; Huber, K.; Moormann, H.; Chandrakasan, A. “A 65nm Sub-Vt Microcontroller with Integrated SRAM and Switched-Capacitor DC-DC Converter”. In: IEEE International Solid-State Circuits Conference (ISSCC), 2008, pp. 318–319, 616.

[KSSF10] Kalla, R.; Sinharoy, B.; Starke, W.; Floyd, M. “Power7: IBM’s Next-Generation Server Processor”, *IEEE Micro*, vol. 30–2, March 2010, pp. 7–15.

[MHBC15] Moreira, M.; Hand, D.; Beerel, P.; Calazans, N. “TDTB Error Detecting Latches: Timing Violation Sensitivity Analysis and Optimization”. In: 16th International Symposium on Quality Electronic Design (ISQED), 2015, pp. 379–383.

[MHH+14] Moreira, M. T.; Heck, L. H.; Heck, G.; Gibiluka, M.; Calazans, N. L. V.; Moraes, F. G. “The YeAH! NoC Router”, Technical Report 083, Faculty of Informatics, PUCRS, 2014.

[NS11] Nowick, S.; Singh, M. “High-Performance Asynchronous Pipelines: An Overview”, *IEEE Design & Test of Computers*, vol. 28–5, Sept 2011, pp. 8–22.

[Pla14] “Plasma CPU”. Source: http://opencores.org/project,plasma, 2014.

[RCN08] Rabaey, J. M.; Chandrakasan, A.; Nikolic, B. “Digital Integrated Circuits”. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008, 3rd ed..

[RMCF88] Rosenberger, F.; Molnar, C.; Chaney, T.; Fang, T.-P. “Q-Modules: Internally Clocked Delay-Insensitive Modules”, *IEEE Transactions on Computers*, vol. 37–9, Sep 1988, pp. 1005–1018.

[SN07] Singh, M.; Nowick, S. “MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15–6, Jun 2007, pp. 684–698.

[Sut89] Sutherland, I. E. “Micropipelines”, *Communications of the ACM*, vol. 32–6, Jun 1989, pp. 720–738.

[Syn06] Synopsys, Inc. “CCS Timing: Technical White Paper”, 2006, available at https://www.opensourceliberty.org/ccspaper/ccs_timing_wp.pdf.

[Syn15] Synopsys, Inc. “Library Compiler User Guide, Version K-2015.06-SP”, 2015.

[Vit15] Vitale, S. “Low voltage devices and circuits for energy-starved systems”. In: SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 2015 IEEE, 2015, pp. 1–3.

[WC05] Wang, A.; Chandrakasan, A. “A 180-mV Subthreshold FFT Processor Using a Minimum Energy Design Methodology”, *IEEE Journal of Solid-State Circuits*, vol. 40–1, Jan 2005, pp. 310–319.

[WH10] Weste, N.; Harris, D. “CMOS VLSI Design: A Circuits and Systems Perspective”. USA: Addison-Wesley Publishing Company, 2010, 4th ed..

[ZGC+12] Zhang, D.; Guo, D.; Chen, F.; Wu, F.; Wu, T.; Cao, T.; Jin, S. “TL-Plane-Based Multi-Core Energy-Efficient Real-Time Scheduling Algorithm for Sporadic Tasks”, *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8–4, Jan 2012, pp. 47:1–47:20.

[ZHA15] Zhao, W.; Ha, Y.; Alioto, M. “AES Architectures for Minimum-Energy Operation and Silicon Demonstration in 65nm with Lowest Energy per Encryption”. In: IEEE International Symposium on Circuits and Systems (ISCAS), 2015, pp. 2349–2352.

[Zod15] Zodik, G. “Future Technologies Supporting the Convergence of Mobile, Wearables, and IoT”. In: 2nd ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft), 2015, pp. 129–130.

# APPENDIX A – SUPPLEMENTARY RESULTS FOR $\overline{PRP_{VS}}$ ANALYSIS

This Appendix contains the $\overline{PRP_{VS}}$ results for the following TRWs: 5%, 20%, 30%, 50% and 75%. The Tables show the results for all possible ranges of voltage scaling from 1V to 250mV with 50mV steps. Cells are colored according to the $\overline{PRP_{VS}}$ magnitude – red denotes higher values and green indicates lower values. Additional details about these results can be found in Section 5.4.

Table A.1 – Average percentage of resilient paths ($\overline{PRP_{VS}}$) for a given range of voltage scaling, considering a 5% TRW.

| Final Supply Voltage (mV) \ Initial Supply Voltage (mV) | 1000 | 950 | 900 | 850 | 800 | 750 | 700 | 650 | 600 | 550 | 500 | 450 | 400 | 350 | 300 | 250 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 7.0% | 6.8% | 6.6% | 6.5% | 6.4% | 6.2% | 6.0% | 5.8% | 5.6% | 5.4% | 5.1% | 2.8% | 2.6% | 2.4% | 2.1% | 1.7% |
| 300 | 6.9% | 6.8% | 6.6% | 6.5% | 6.4% | 6.2% | 6.0% | 5.7% | 5.5% | 5.4% | 5.0% | 2.6% | 2.4% | 2.2% | 1.8% | |
| 350 | 6.9% | 6.7% | 6.5% | 6.4% | 6.3% | 6.1% | 5.9% | 5.6% | 5.4% | 5.3% | 4.9% | 2.5% | 2.3% | 2.0% | | |
| 400 | 6.8% | 6.6% | 6.5% | 6.3% | 6.2% | 6.0% | 5.8% | 5.6% | 5.3% | 5.2% | 4.8% | 2.3% | 2.1% | | | |
| 450 | 6.7% | 6.5% | 6.4% | 6.3% | 6.1% | 6.0% | 5.7% | 5.5% | 5.2% | 5.1% | 4.7% | 2.2% | | | | |
| 500 | 5.6% | 5.4% | 5.2% | 5.0% | 4.9% | 4.7% | 4.4% | 4.1% | 3.8% | 3.7% | 3.2% | | | | | |
| 550 | 5.5% | 5.3% | 5.2% | 5.0% | 4.8% | 4.6% | 4.3% | 4.0% | 3.7% | 3.6% | | | | | | |
| 600 | 5.5% | 5.3% | 5.1% | 4.9% | 4.8% | 4.5% | 4.2% | 3.9% | 3.6% | | | | | | | |
| 650 | 5.3% | 5.1% | 5.0% | 4.8% | 4.6% | 4.4% | 4.1% | 3.8% | | | | | | | | |
| 700 | 5.3% | 5.1% | 4.9% | 4.7% | 4.6% | 4.4% | 4.0% | | | | | | | | | |
| 750 | 5.1% | 5.0% | 4.8% | 4.6% | 4.4% | 4.2% | | | | | | | | | | |
| 800 | 5.1% | 4.9% | 4.7% | 4.5% | 4.4% | | | | | | | | | | | |
| 850 | 5.0% | 4.8% | 4.6% | 4.4% | | | | | | | | | | | | |
| 900 | 4.9% | 4.7% | 4.5% | | | | | | | | | | | | | |
| 950 | 4.7% | 4.5% | | | | | | | | | | | | | | |
| 1000 | 4.6% | | | | | | | | | | | | | | | |

Table A.2 – Average percentage of resilient paths ($\overline{PRP_{VS}}$) for a given range of voltage scaling, considering a 20% TRW.

| Final Supply Voltage (mV) \ Initial Supply Voltage (mV) | 1000 | 950 | 900 | 850 | 800 | 750 | 700 | 650 | 600 | 550 | 500 | 450 | 400 | 350 | 300 | 250 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 29.4% | 29.1% | 28.5% | 28.0% | 27.4% | 26.7% | 26.0% | 25.1% | 24.1% | 23.3% | 22.2% | 13.1% | 12.9% | 12.4% | 11.8% | 11.1% |
| 300 | 29.4% | 29.0% | 28.4% | 27.9% | 27.3% | 26.6% | 25.9% | 25.0% | 24.0% | 23.1% | 22.0% | 12.2% | 12.0% | 11.5% | | |
| 350 | 29.2% | 28.9% | 28.2% | 27.8% | 27.1% | 26.5% | 25.7% | 24.8% | 23.8% | 22.9% | 21.8% | 11.7% | 11.5% | | | |
| 400 | 29.1% | 28.7% | 28.1% | 27.6% | 27.0% | 26.3% | 25.5% | 24.6% | 23.6% | 22.7% | 21.6% | 11.2% | 10.8% | | | |
| 450 | 28.9% | 28.6% | 28.0% | 27.5% | 26.8% | 26.2% | 25.4% | 24.4% | 23.4% | 22.5% | 21.4% | 10.7% | | | | |
| 500 | 27.5% | 27.1% | 26.5% | 25.9% | 25.1% | 24.3% | 23.5% | 22.5% | 21.4% | 20.3% | 19.0% | | | | | |
| 550 | 27.3% | 27.0% | 26.3% | 25.8% | 25.0% | 24.2% | 23.3% | 22.3% | 21.2% | 20.1% | | | | | | |
| 600 | 27.2% | 26.8% | 26.1% | 25.6% | 24.8% | 24.0% | 23.1% | 22.1% | 21.0% | | | | | | | |
| 650 | 26.8% | 26.4% | 25.8% | 25.2% | 24.4% | 23.6% | 22.7% | 21.7% | | | | | | | | |
| 700 | 26.5% | 26.1% | 25.5% | 24.9% | 24.1% | 23.3% | 22.4% | | | | | | | | | |
| 750 | 26.4% | 26.0% | 25.3% | 24.8% | 23.9% | 23.1% | | | | | | | | | | |
| 800 | 26.3% | 25.9% | 25.2% | 24.6% | 23.7% | | | | | | | | | | | |
| 850 | 26.2% | 25.8% | 25.1% | 24.5% | | | | | | | | | | | | |
| 900 | 26.1% | 25.7% | 24.9% | | | | | | | | | | | | | |
| 950 | 25.7% | 25.2% | | | | | | | | | | | | | | |
| 1000 | 25.3% | | | | | | | | | | | | | | | |

Table A.3 – Average percentage of resilient paths ($\overline{PRP_{VS}}$) for a given range of voltage scaling, considering a 30% TRW.

| Final Supply Voltage (mV) \ Initial Supply Voltage (mV) | 1000 | 950 | 900 | 850 | 800 | 750 | 700 | 650 | 600 | 550 | 500 | 450 | 400 | 350 | 300 | 250 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 45.0% | 44.7% | 44.2% | 43.8% | 43.2% | 42.5% | 41.7% | 40.8% | 39.9% | 39.0% | 37.5% | 21.5% | 21.3% | 21.0% | 20.3% | 19.4% |
| 300 | 45.0% | 44.7% | 44.2% | 43.7% | 43.1% | 42.4% | 41.7% | 40.8% | 39.8% | 38.9% | 37.4% | 20.6% | 20.3% | 20.0% | 18.8% | |
| 350 | 44.9% | 44.6% | 44.1% | 43.6% | 43.0% | 42.3% | 41.5% | 40.6% | 39.6% | 38.7% | 37.2% | 19.9% | 19.5% | 18.8% | | |
| 400 | 44.8% | 44.4% | 43.9% | 43.5% | 42.8% | 42.1% | 41.3% | 40.4% | 39.4% | 38.5% | 37.0% | 19.1% | 18.6% | | | |
| 450 | 44.7% | 44.4% | 43.9% | 43.4% | 42.8% | 42.0% | 41.3% | 40.3% | 39.3% | 38.3% | 36.8% | 18.3% | | | | |
| 500 | 43.3% | 42.9% | 42.4% | 41.8% | 41.1% | 40.3% | 39.4% | 38.2% | 37.0% | 35.8% | 34.0% | | | | | |
| 550 | 43.1% | 42.7% | 42.2% | 41.6% | 40.9% | 40.0% | 39.2% | 38.0% | 36.7% | 35.6% | | | | | | |
| 600 | 43.0% | 42.6% | 42.1% | 41.5% | 40.8% | 39.9% | 39.0% | 37.9% | 36.6% | | | | | | | |
| 650 | 42.9% | 42.5% | 42.0% | 41.4% | 40.7% | 39.8% | 38.9% | 37.8% | | | | | | | | |
| 700 | 42.8% | 42.4% | 41.9% | 41.3% | 40.6% | 39.7% | 38.8% | | | | | | | | | |
| 750 | 42.6% | 42.2% | 41.7% | 41.1% | 40.3% | 39.5% | | | | | | | | | | |
| 800 | 42.5% | 42.1% | 41.6% | 41.0% | 40.3% | | | | | | | | | | | |
| 850 | 42.4% | 42.0% | 41.5% | 40.9% | | | | | | | | | | | | |
| 900 | 42.3% | 41.9% | 41.4% | | | | | | | | | | | | | |
| 950 | 41.9% | 41.3% | | | | | | | | | | | | | | |
| 1000 | 41.7% | | | | | | | | | | | | | | | |

Table A.4 – Average percentage of resilient paths ($\overline{PRP_{VS}}$) for a given range of voltage scaling, considering a 50% TRW.

| Final Supply Voltage (mV) \ Initial Supply Voltage (mV) | 1000 | 950 | 900 | 850 | 800 | 750 | 700 | 650 | 600 | 550 | 500 | 450 | 400 | 350 | 300 | 250 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 71.4% | 71.0% | 70.4% | 69.8% | 69.3% | 68.6% | 68.0% | 67.3% | 66.6% | 65.7% | 64.9% | 46.1% | 45.9% | 45.6% | 44.8% | 44.3% |
| 300 | 71.4% | 71.0% | 70.4% | 69.8% | 69.3% | 68.6% | 68.0% | 67.3% | 66.6% | 65.7% | 64.9% | 44.5% | 44.2% | 43.8% | 42.7% | |
| 350 | 71.4% | 70.9% | 70.3% | 69.8% | 69.2% | 68.6% | 67.9% | 67.2% | 66.5% | 65.6% | 64.8% | 43.1% | 42.7% | 41.8% | | |
| 400 | 71.3% | 70.8% | 70.2% | 69.7% | 69.1% | 68.5% | 67.8% | 67.1% | 66.4% | 65.5% | 64.6% | 42.0% | 41.3% | | | |
| 450 | 71.3% | 70.8% | 70.2% | 69.6% | 69.1% | 68.4% | 67.7% | 67.0% | 66.3% | 65.4% | 64.5% | 40.9% | | | | |
| 500 | 70.8% | 70.3% | 69.6% | 68.9% | 68.3% | 67.6% | 66.8% | 65.9% | 65.1% | 64.0% | 63.0% | | | | | |
| 550 | 70.7% | 70.2% | 69.5% | 68.8% | 68.2% | 67.5% | 66.7% | 65.8% | 65.0% | 63.9% | | | | | | |
| 600 | 70.6% | 70.1% | 69.4% | 68.8% | 68.1% | 67.4% | 66.6% | 65.7% | 64.9% | | | | | | | |
| 650 | 70.6% | 70.0% | 69.4% | 68.7% | 68.1% | 67.3% | 66.6% | 65.6% | | | | | | | | |
| 700 | 70.5% | 70.0% | 69.4% | 68.7% | 68.0% | 67.3% | 66.5% | | | | | | | | | |
| 750 | 70.4% | 69.9% | 69.2% | 68.5% | 67.8% | 66.9% | | | | | | | | | | |
| 800 | 70.3% | 69.8% | 69.1% | 68.4% | 67.8% | | | | | | | | | | | |
| 850 | 70.2% | 69.7% | 69.0% | 68.3% | | | | | | | | | | | | |
| 900 | 70.1% | 69.6% | 68.9% | | | | | | | | | | | | | |
| 950 | 69.9% | 69.3% | | | | | | | | | | | | | | |
| 1000 | 69.7% | | | | | | | | | | | | | | | |

Table A.5 – Average percentage of resilient paths ($\overline{PRP_{VS}}$) for a given range of voltage scaling, considering a 75% TRW.

| Final Supply Voltage (mV) \ Initial Supply Voltage (mV) | 1000 | 950 | 900 | 850 | 800 | 750 | 700 | 650 | 600 | 550 | 500 | 450 | 400 | 350 | 300 | 250 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 250 | 93.6% | 93.2% | 92.9% | 92.4% | 92.0% | 91.5% | 90.8% | 90.3% | 89.9% | 89.4% | 88.9% | 77.7% | 77.6% | 77.5% | 77.2% | 76.8% |
| 300 | 93.6% | 93.2% | 92.9% | 92.4% | 92.0% | 91.5% | 90.7% | 90.3% | 89.9% | 89.4% | 88.9% | 76.3% | 76.2% | 76.0% | 75.4% | |
| 350 | 93.6% | 93.2% | 92.9% | 92.4% | 92.0% | 91.5% | 90.7% | 90.3% | 89.8% | 89.3% | 88.9% | 75.4% | 75.3% | 74.9% | | |
| 400 | 93.6% | 93.2% | 92.9% | 92.4% | 92.0% | 91.5% | 90.7% | 90.3% | 89.8% | 89.3% | 88.9% | 74.6% | 74.3% | | | |
| 450 | 93.6% | 93.2% | 92.9% | 92.4% | 92.0% | 91.5% | 90.7% | 90.3% | 89.8% | 89.3% | 88.9% | 73.8% | | | | |
| 500 | 93.6% | 93.2% | 92.9% | 92.4% | 91.9% | 91.4% | 90.7% | 90.0% | 89.5% | 88.9% | 88.4% | | | | | |
| 550 | 93.5% | 93.1% | 92.9% | 92.4% | 91.9% | 91.4% | 90.6% | 90.0% | 89.5% | 88.9% | | | | | | |
| 600 | 93.5% | 93.1% | 92.9% | 92.4% | 91.9% | 91.4% | 90.6% | 90.0% | 89.5% | | | | | | | |
| 650 | 93.5% | 93.1% | 92.9% | 92.3% | 91.9% | 91.4% | 90.6% | 90.0% | | | | | | | | |
| 700 | 93.5% | 93.1% | 92.8% | 92.3% | 91.9% | 91.4% | 90.6% | | | | | | | | | |
| 750 | 93.5% | 93.1% | 92.8% | 92.3% | 91.8% | 91.3% | | | | | | | | | | |
| 800 | 93.5% | 93.1% | 92.8% | 92.3% | 91.8% | | | | | | | | | | | |
| 850 | 93.5% | 93.1% | 92.8% | 92.3% | | | | | | | | | | | | |
| 900 | 93.4% | 93.0% | 92.8% | | | | | | | | | | | | | |
| 950 | 93.4% | 92.9% | | | | | | | | | | | | | | |
| 1000 | 93.4% | | | | | | | | | | | | | | | |