ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME HECK

THE IMPACT OF VOLTAGE SCALING OVER DELAY ELEMENTS WITH FOCUS ON POST-SILICON TESTS

Porto Alegre

2018

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

THE IMPACT OF VOLTAGE SCALING OVER DELAY ELEMENTS WITH FOCUS ON POST-SILICON TESTS

GUILHERME HECK

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

Advisor: Prof. Dr. Ney L. V. Calazans

Porto Alegre
2018

Ficha Catalográfica

H448i Heck, Guilherme

The impact of voltage scaling over delay elements with focus on post-silicon tests / Guilherme Heck . – 2018.
147.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Ney Laert Vilar Calazans.

1. Delay elements. 2. Voltage scaling. 3. Asynchronous. 4. Bundled-data. 5. Resiliency. I. Calazans, Ney Laert Vilar. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Guilherme Heck

The Impact of Voltage Scaling over Delay Elements with Focus on post-Silicon Tests

This Dissertation/Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 9th, 2018.

COMMITTEE MEMBERS:

To my family.

"Whoever you are, no matter what social position you have, rich or poor, always show great strength and determination, and always do everything with much love and deep faith in God. One day you will reach your goal."
(Ayrton Senna da Silva, 1991)

ACKNOWLEDGMENTS

Gostaria de agradecer muito aos meus pais. Meu pai Reinaldo Heck e minha mãe Ida heck foram as peças fundamentais para a conclusão dessa etapa. Foram tempos difíceis e, graças a eles, voltei a ter forças para seguir essa ardua caminhada. Esse trabalho eu dedico principalmente a vocês que fizeram tantos sacrifícios e estiveram sempre comigo. Sou grato também aos meus irmãos Marcos Heck e Bárbara heck que também se preocuparam com o desenrolar dessa novela e pelo apoio emocional. Agradeço aos meus cunhados Sônia Heck e Sérgio Ucoski que também estiveram ao meu lado. De um modo todo especial também gostaria de agradecer aos meus sobrinhos Artur, Henrique, Caroline, Eduardo, Diego e Isadora. Estiveram sempre em meus pensamentos e, assim como todos, me alegraram e deram força para seguir essa história. Peço desculpa pela falta de tempo para com vocês, mas serei eternamente grato a cada um pela compreenção. Não fazem ideia de quão importantes foram para a conclusão dessa etapa. Amo muito todos vocês.

Ao Ney, meu orientador, agradeço pelas horas de conversa e pelos auxílios durante os trabalhos realizados aqui. Tudo que consta aqui só foi possível com sua ajuda. Fizeste uma grande diferença na minha visão de acadêmico e tudo que aprendi com você serviu de referência para meu futuro profissional. Agradeço também aos demais professores do Grupo de Apoio ao Projeto de Hardware (GAPH) e do Grupo de Sistemas Embarcados (GSE). Alexandre Amory e seus auxílios em testes e uma pitadinha de robótica. Carlos Moratelli e as conversas do cotidiano e da política econômica. César Marcon pelo grande apoio no desenrolar dessa caminhada. Fabiano Hessel pelas discuções acaloradas de futebol. Fernando Moraes pelas conversas técnicas e discuções de trabalho. Grandes papos sobre infraestrutura e sobre tecnologia. Obrigado pelos papos fundamentais para acalmar os ânimos e seguir em frente. Sérgio Johann Filho e os papos aleatórios desde alimentação com sucos de laranja e frangos assados de televisão de cachorro a questões técnicas sobre implementações de SPI em hardware. Esse contato quase que diário me mostrou como um professor deve ser. Me mostrou quanto orgulho vocês tem em ajudar os outros e trazer um pouco de luz na escuridão atual.

Aos meus colegas de laboratório nessa caminhada, todos sabem o que desejo. Mas fica aqui um agradecimento especial aos meus colegas diretamente relacionados ao grupo de assíncronos: Leandro Heck, Ricardo Guazzelli, Matheus Gibiluka e Matheus Moreira. O Leandro me apoiou e ensinou a necessidade de deixar tudo organizado. A memória nem sempre ajuda num período tão longo. Sharelatex é vida. Ricardo sempre pronto para ajudar em algum projeto, dando várias ideias e levantando discuções relevantes. Obrigado pelas ajudas e pelas conversas. Matheus Gibiluka também incistiu em trabalhar com algo semelhante ao meu trabalho (voltage scaling). Agradeço pelas ajudas desde o mestrado quando bolsista até durante o doutorado. Por fim, mas não menos importante, Matheus Moreira me ajudou muito nas questões técnicas justamente por se interessar por tudo envolvendo microeletrônica. Muito obrigado por ter me recebido em sua casa em Los Angeles e pelos altos churrascos. Agradecimento especial também ao Felipe Kuentzer e ao Leonardo Rezende que, por várias vezes ouviram e discutiram problemas de testabilidade dos circuitos comigo. Os demais integrantes do GAPH e GSE tiveram importância principalmente em manter minha sanidade mental, dentre eles André Del Mestre, Bruno Oliveira, Eduardo Wachter, Felipe Magalhães, Felipe Bortolon, Gelmar da Costa, Guilherme Castilhos, Guilherme Madalozzo, Guilherme Medeiros, Lucas Copetti, Luciano Caimi, Luciano Ost, Marcelo Links, Marcelo Mandelli, Marcelo Ruaro, Walter Lau Neto. Desculpe a chatice e intrometimento nos papos. A intenção é sempre ajudar e melhorar. Muitas cervejas tomadas, churrascos comidos, mas principalmente muita amizade. Obrigado pelo convívio.

I am so grateful to Peter. Thanks for all the technical discussions we had. Thanks for always being so supportive. Thanks also go to my friends at USC who received me with open arms. Special thanks to Fei and Dylan, who became good friends. I also thank my colleagues and friends in the Async lab, Atharva, Yang, Ramy and Will, for the technical discussions and fun times. Thank you for all the good times in Los Angeles!

Finally, thank's to IEEE and its societies, specially the Computer Society and the Circuits and Systems Society, for the awards and student grants that helped me attending conferences. Agradeço à Faculdade de Informática (FACIN), ao Programa de Pós-Graduação em Ciências da Computação (PPGCC) e à Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) pela excelente estrutura e grandes funcionários. Agradeço à Hewlett-Packard (HP) que me forneceu auxílio durante o período do doutorado, além dos auxílios provenientes do governo brasileiro junto ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (Processo 142263/2012-5) e também o auxílio durante o período de Sanduíche na University of Southern California (USC) (processo 202519/2014-7), e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) (programa PROSUP taxas) e, por fim, à DELL que proporcionaram algumas das taxas de matrículas à PUCRS além da IEEE (grants em eventos).

O IMPACTO DE VARIAÇÕES DA TENSÃO DE ALIMENTAÇÃO SOBRE ELEMENTOS DE ATRASO COM FOCO EM TESTES PÓS-FABRICAÇÃO

RESUMO

A demanda sem precedentes por poderosos dispositivos de processamento gerou quebras consecutivas de paradigma de projeto de circuito na área de Circuitos Integrados (CIs). O uso de tecnologia submicrométrica profunda aumenta a densidade de integração a níveis nunca vistos antes. No entanto, com CIs mais densos, a inclinação do relógio e outros efeitos requerem compensações em design síncrono, o que pode aumentar a área e o consumo de energia a valores inaceitáveis. Como alternativa, o paradigma assíncrono está re-emergindo, focado na eficiência de energia. Entre os modelos clássicos de projeto assíncrono, o Empacotamento-de-Dados (ED) se destaca pela sua capacidade de fornecer alto desempenho, reduzir a potência e obter resultados de área semelhante à dos modelos síncronos. Diferentemente dos modelos mais robustos de quase-atraso insensível, uma outra classe comum de modelos para implementar circuitos assíncronos, circuitos ED requerem o uso extensivo de Elementos de Atraso (EAs) para garantir a correta funcionalidade. No entanto, todos os circuitos são afetados por variações de Processo, Tensão e Temperatura (PTT), incluindo a Lógica Combinacional (LC) em ED impondo margem em elementos de atraso. Além disso, projetos atuais usam escalonamento de tensão para melhorar a eficiência de energia, o que afeta o atraso diferentemente em LCs e EAs adicionando mais margem em EAs. Um novo modelo baseado em ED chamado Blade usa o conceito de resiliência como uma esperança para evitar a margem de atraso causada por PTT e escalonamento de tensão. Contudo, o uso de dois elementos de atraso irá representar mais margens e mais tempo de teste no circuito final. Assim, este trabalho mostra uma análise do comportamento de elementos de atraso sob escalonamento de tensão e o impacto em testes pós-silício. Ele introduz um novo termo para determinar o impacto da escala de tensão sobre os elementos de atraso e também a comparação entre os EAs mais utilizados em projetos ED usando esta nova métrica. Uma análise de testes em modelos ED e Blade é apresentada e o impacto da escala de tensão nestes projetos é analisado. Finalmente, um novo elemento de atraso é proposto focando na redução de margem e redução no tempo de teste para o modelo Blade.

Palavras-Chave: Elementos de Atraso, Escalonamento de Tensão, Assíncronos, Empacotamento de Dados, Resiliência, Variação de Processo, Tensão e Temperatura, Teste Pós-Silício.

THE IMPACT OF VOLTAGE SCALING OVER DELAY ELEMENTS WITH FOCUS ON POST-SILICON TESTS

ABSTRACT

The unprecedented demand for powerful processing devices has generated consecutive circuit design paradigm breaks in the Integrated Circuits (ICs) arena. The use of deep submicron technology increases the integration density to levels never seen before. However, with denser ICs, clock skew and other effects require compensations in synchronous design, which can increase area overhead and power consumption to unacceptable values. As an alternative, the asynchronous paradigm is re-emerging, focused on power efficiency. Among classical asynchronous design templates, the Bundled-Data (BD) one stands off for its capability to provide high performance, reduce power and achieve area results similar to that of synchronous designs. Unlike the more robust Quasi-Delay Insensitive (QDI) templates, another common class of templates to implement asynchronous circuits, BD circuits require the extensive use of Delay Elements (DEs) to guarantee correct functionality. However, all circuits are affected by Process, Voltage and Temperature (PVT) variations, including the Combinational Logic (CL) on BD imposing margin on delay elements. In addition, current designs use voltage scaling to improve power efficiency, which impacts the delay differently in CLs and DEs adding more margin in DEs. A new template based on BD called Blade uses resiliency concept as a hope to avoid the delay margin caused by PVT and voltage scaling. Although, the use of two delay elements will represents more margins and extra test time on final circuit. So, this work shows an analysis of delay elements behavior under voltage scaling and the impact on post-silicon tests. It introduces a new term to determine the voltage scaling impact on delay elements and also the comparison between the most used DEs on BD designs using this novel metric. An analysis of tests in BD and Blade templates are presented and the impact of voltage scaling in these designs is analyzed. Finally, a novel delay element is proposed focusing in margin reduction and reduction in test time for Blade template.

Keywords: Delay Elements, Voltage Scaling, Asynchronous, Bundled-Data, Resiliency, Blade, Process, Voltage and Temperature (PVT) Variations, Post-Silicon Tests.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

ADE – Analog Design Environment

AICSP – Analog Integrated Circuits and Signal Processing

AS – At-Speed

ATPG – Automatic Test Pattern Generation

BD – Bundled-Data

BIST – Built-In Self-Test

BTBT – Band-To-Band Tunneling

CAD – Computer Aided Design

CL – Combinational Logic

CMCS – Current mirror-Controlled Current-Starved

CMOS – Complementary Metal Oxide Semiconductor

CPU – Central Processing Unit

CSI – Current-Starved Inverter

CUT – Circuit Under Test

DCCS – Direct-Controlled Current-Starved

DCO – Digitally Controlled Oscillator

DE – Delay Element

DEF – Design Exchange Format

DFS – Dynamic Frequency Scaling

DFT – Design For Testability

DI – Delay Insensitive

DIBL – Drain-Induced Barrier Lowering

DLL – Delay-Locked Loop

DSM – Deep SubMicron

DVFS – Dynamic Voltage and Frequency Scaling

DVS – Dynamic Voltage Scaling

ECAD – Electronic Computer Aided Design

EDL – Error Detector Logic

FDSOI – Fully Depleted Silicon-On-Insulator

FET – Field Effect Transistor

FF – Flip-Flop

FFT – Fast Fourier Transform

FO4 – Fanout Of Four

FS – Frequency Scaling

FTAS – Faster-Than-At-Speed

GALS – Globally Asynchronous Locally Synchronous

GDS – Graphic Database System

GIDL – Gate-Indiced Drain Leakage

HDL – Hardware Description Language

IC – Integrated Circuit

IOT – Internet of Things

IP – Intellectual Property

ITRS – International Technology Roadmap for Semiconductors

LASCAS – Latin America Symposium in Circuits and Systems

LEF – Library Exchange File

LFSR – Linear Feedback Shift Register

LOC – Launch-Off-Capture

LOCOS – LOCal-Oxidation of Silicon

LOS – Launch-Off-Shift

LUT – Look Up Table

LVS – Layout Versus Schematic

MOS – Metal Oxide Semiconductor

MOSFET – Metal Oxide Semiconductor Field Effect Transistor

MUX – Multiplexer

NCL – Null Convention Logic

NDA – Non-Disclosure Agreement

NRZ – Non-Return to Zero

OASIS – Open Artwork System Interchange Standard

ORA – Output Response Analyzer

PDF – Path-Delay Fault

PDK – Process Design Kit

PEX – Practices EXtraction

PLL – Phase-Locked Loop

PNS – Pass-nSection

PS – Pass-Section

PVT – Process, Voltage and Temperature

PWL – PieceWise Linear

QDI – Quasi-Delay Insensitive

RF – Radio Frequency

RTL – Register-Transfer Level

RZ – Return to Zero

SDD – Small-Delay Defect

SDF – Standard Delay Format

SEE – Single Event Effect

STI – Shallow-Trench Isolation

STM – STMicroelectronics

TDF – Transition-Delay Fault

TPG – Test Pattern Generation

TSMC – Taiwan Semiconductor Manufacturing Company

TDC – Time-to-Digital Converter

VLSI – Very Large Scale Integration

VS – Voltage Scaling

VSDR – Voltage Scaled Delay Ratio

CONTENTS

1. INTRODUCTION

The semi-conduction property of some chemical elements and compounds and the almost limitless exploitation of their potential enabled most of the technological feats in the recent history of Humankind. Silicon electronic devices were invented taking advantage of this property. The reliability of these devices compared to the available alternatives led to their increasing adoption and spurred the production of information processing systems that changed the speed and way with which many, if not most human activities are conducted. Discrete electronic devices evolved into Integrated Circuits (ICs) in silicon. The miniaturization of device dimensions and the mass manufacturing of these create the possibility of building highly complex information processing systems in very small packages at a very low cost. Small and cheap systems can be deployed easily everywhere. Based on Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs), most of these systems are widely used, being present in many home appliances as well as in satellites that orbit planets.

Very-Large-Scale Integration (VLSI) technology allows to put a complete Central Processing Unit (CPU) on a single chip since 1970 [Wol05]. In that time, synchronous circuit design techniques, which provide continuous time abstraction, help designers to easily design and fabricate complex circuits. The number of registers and clock-dependent gates did not pose a problem to distribute global signals such as clock and reset to all transistors involved. Today, the use of Deep SubMicron (DSM) technologies increases transistor integration density deeply, leading to devices that can exceed 30 billion transistors in a single chip [Gaz15].

The gains achieved by using DSM do not come free from problems. Global or semi-global wires distribution, their delay determination and related effects constitute a highly complex problem in multi-billion-transistor designs. Clock, reset and power lines are examples of signals that require careful design. Effects that need tight control include, but are not restricted to:

i The skew in clocks, caused by wire parasitic resistances and large capacitive loads;

ii Voltage drops in power supply lines, affected by gate switching activity and by the relative length and resistivity of supply wires;

iii Crosstalk in wide buses carrying data among chip modules.

Current methods and tools implemented in Electronic Computer Aided Design (ECAD or simply CAD) try to compensate such effects by specific techniques, like inserting buffers or deskewers (specific cells for handling clock delays) when automatically generating clock distribution trees. These solutions increase the IC area overhead and power consumption. According e.g. to Amde et al. [AFE+05], the power consumption for distributing a clock signal in a synchronous chip can reach up to 50% of the total chip power consumption.

Table 1.1 – Low power Design Technology Improvements and Impact on Dynamic and Static Power [ITR].

DT Improvement	Year	Dynamic Power Improvement (x)	Static Power Improvement (x)	Description of Improvements
Low Power Physical Libraries	Before 2011	1.50	1.50	Optimizing transistor size, layout style and cell topology for the standard-cell library
Back Biasing		1.00	1.35	Biasingwells of devices independently of the sources to shift the threshold voltage
Adaptive Body Biasing (ABB)		1.20	2.00	Delivering a positive or negative voltage below a transistor to reduce leakage
Power Gating		0.90	10.00	Turning off the power supplies to idle blocks for leakage reduction
Dynamic Voltage and Frequency Scaling (DVFS)		1.50	1.00	Dynamic management of supply voltage and operating frequency for power reduction
Multilevel Cache Architecture		1.00	1.20	Reduce amount of off-chip memory accesses for performance improvement and power reduction
Hardware Multithreading		1.00	1.30	Using multithreads to improve hardware utilization with leakage reduction
Hardware Virtualization		1.00	1.20	Using one physical server to support multiple guest operating systems simultaneously
Superscalar Architecture		1.00	2.00	Parallel instruction issue and execution for performance improvement and power reduction
Symmetric Multiprocessing (SMP)		1.50	1.00	Lowering the frequency by using multiple processors and parallel programing
Software Virtual Prototype	2011	1.23	1.20	Virtualization tools to allow the programmer to develop software prior to silicon
Frequency Islands	2013	1.26	1.00	Designing blocks that operate at different frequencies
Near-Threshold Computing	*2015*	*1.23*	*0.80*	*Lowering Vdd to 400 - 500 mV*
Hardware/Software Co-Partitioning	2017	1.18	1.00	Hardware/software partitioning at the behavioral level based on power
Heterogeneous Parallel Processing (HPP)	2019	1.18	1.00	Using multiple types of processors in a parallel computing architecture
Many Core Software Development Tools	2021	1.20	1.00	Using multiple types of processors in a parallel computing architecture
Power-Aware Software	2023	1.21	1.00	Developing software using power consumption as a parameter
Asynchronous Design	*2025*	*1.21*	*1.00*	*Non-clock driven design*

In view of this situation and of a growing demand for low power devices, the International Technology Roadmap for Semiconductors (ITRS) [ITR] predicts the need for IC design paradigm changes. As presented in Table 1.1, the latest ITRS document available to date on IC design indicated the possibility of improvements on dynamic power by up to 23% using near-threshold voltage operation until 2015 and 21% gains through the adoption of asynchronous paradigm design in the following ten years targeting more energy efficient devices. Asynchronous design regained attention in several recent research works [BR07, CCGC10, ZSD10, CVPS11, Rab, LNS13, MAGC14, SHB+14, HHS+15, HMH+15, CLM+16, MCFB16, THG+16, HSL17]. This shows a trend to exploit this hardware design paradigm to develop low power ICs.

While synchronous circuits depend on the clock synchronization signal, asynchronous circuits employ handshake protocols to mediate data exchange. One classical asynchronous design template, the Quasi-Delay Insensitive (QDI), uses special information coding schemes to include part of the handshake signals within the data representation. This robust encoding combines control and data information, which provides an advantage when using a wide range of Voltage Scaling (VS) operation. Works such as [CL10, CCGC10, HCGC15] show QDI circuits properly operating in subthreshold voltages, achieving very low power dissipation. However, QDI coding schemes usually increase area overhead considerably, which implies low efficiency under nominal voltage. Also, QDI design often requires the use of special types of logic gates, but current commercial technologies Process Design Kits (PDKs) and cell libraries do not include such gates, which is a major impediment to improve CAD tool support to QDI design.

An alternative to design asynchronous circuit can more easily take advantage of current commercial CAD tools are to use Bundled-Data (BD) templates. Unlike QDI asynchronous design templates, BD templates use classical information binary coding and the handshake protocol relies on Delay Elements (DEs). BD is also highlighted for its potential to provide high performance, low power and area similar to that of synchronous designs [THK+05].

Due to the intrinsic characteristics of BD templates, their design flow is similar to conventional synchronous circuits. Thus, much of the developed CAD tools for synchronous design can be applied for BD design flows. The handshake protocol substitutes the traditional clock signaling by handshake mechanisms. A common assumption among these in BD design consists in using DEs, as discussed e.g. in [TB02, KHS07, SLKR13, GMMC15, RGP+15, HMH+15, THG+16, RDH+17]. DEs are responsible for ensuring that the computed data are stable in some module input when activation of the handshake control signals takes place.

DEs are widely used in digital and analog circuits. Most applications, especially the Radio Frequency (RF) domain, require an accurate delay value even under Process, Voltage or Temperature (PVT) variations. Thus, it is common that these elements are designed and evaluated for tolerance to PVT variations as discussed e.g. in [AESK+12, KKJ12, JKMK13, HHM+16]. However, BD circuit designers recognize that these changes inevitably affect the Combinational Logic (CL), which involves insertion of additional margins in delay elements [TB02, KHS07, SLKR13, HHC+15, RDH+17].

This Thesis deals with the investigation, design and evaluation of delay elements deemed for use in asynchronous bundled-data circuits and, to a minor extent, in synchronous circuits.

1.1 Motivation

The challenge to modern IC designers are the tight power budgets that extend from mobile applications concerned with battery lifetime, to high-end server processors concerned with heat dissipation [HB13]. The power dissipation in ICs can be estimated by the sum of static power (P_{static}) and dynamic power ($P_{dynamic}$) like Equation 1.1 shows.

$$P_{total} = P_{static} + P_{dynamic} \tag{1.1}$$

Static power is produced independent of any circuit activity. It is related to the current that leaks (I_{leak}) from the circuit. Dynamic power, on the other hand, comes from the circuit switching activity when wires change logic values. The switching activity in synchronous circuits is directly dependent on the clock frequency (f). In this case, Equation 1.1 can be rewritten as [WH11]:

$$P_{total} = I_{leak} \cdot V_{DD} + I_{avg} \cdot V_{DD} = I_{leak} \cdot V_{DD} + \alpha \cdot f \cdot C \cdot V_{DD}^2 \tag{1.2}$$

where V_{DD} is the supply voltage, I_{avg} is the average current consumed during circuit switching, αf is the effective switching frequency and C is the overall circuit capacitance.

Analyzing Equation 1.2, it is possible see two ways to handle power: change the frequency activity and changing the supply voltage. Dynamic Frequency Scaling (DFS) [WH11] [RLCM12] [HHM+15] is the easiest technique to reduce power in synchronous circuits, by changing clock period according to power constraints. However, the benefits of the technique in power grow linear and do not interfere in P_{static} values. Dynamic Voltage Scaling (DVS) is in turn more efficient to reduce power [RCN03, WH11]. In addition to linear static power dependence, dynamic power depends quadratically on the supply voltage. Nevertheless, gate delay also depends on the supply voltage, resulting in an interdependence between supply voltage and clock period [RCN03]. Because of this effect, many designers plead for the use of Dynamic Voltage and Frequency Scaling (DVFS) techniques instead of DVS [GJZ+17, MRSM17].

Despite the wide range of studies available involving the analysis and design of voltage scaling techniques in synchronous circuits, few works deal with the impact of VS in BD circuits. The lack of studies in BD covering VS is a relevant and so far neglected research field, specially with regard to impacts of VS in delay elements. This constitutes a basic motivation for the research proposed herein.

1.2 Description of the Problem to Address

Bundled-data templates use delay elements to provide the self-timed characteristic of asynchronous circuits [SF13]. However, the structure and physical characteristics of DEs are distinct from those of their associated combinational logic.

Usually, it is easy to match the combinational logic delay with the DE delay for specific process, voltage and temperature conditions. However, as DEs employ specific gates or cells, that often are not those used to build combinational logic, when process, voltage or temperature vary, the delay produced in these two circuit parts may evolve in different ways. The usual solution in this case is add delay margins in DEs to guarantee a minimum delay that considers the worst case scenario, which may result losses in area, power and performance. In this case, performance in BD templates can be bad, even worse than that of a synchronous solution.

To reduce the delay margin effects, one of the most efficient solutions is to implement programmable DEs. Programmable DEs allow adjustments in the produced delay to increase performance. Basically, programmable DEs can be adjusted at design time or at runtime. However, voltage variations can affect circuitry faster than it can be sensed and compensated [WH11], producing unexpected delays and resulting in timing violations. Alternatives to handle timing violation using resilient circuit design techniques have recently received attention and variant design techniques now exist for BD circuits.

By now, it should be clear the importance of modeling the delay behavior in delay elements and other gates and reference these changes to voltage scaling. Understanding how the delay behavior of logic is affected, it can be possible to predict the impact of VS in both DE and data processing logic, thus reducing the need to add delay margins.

It is also important to indicate how to design DEs for BD with focus on correct operation and high performance reducing margins even under VS. Obviously, discrepancies between how delay parameters in DEs and in the remainder of the circuit change cannot interfere in circuit functionality.

Despite resilient alternatives to BD design templates can circumvent or recover from timing violation, this architecture can add complexities in the design and production of systems that were not previously considered. This shows the importance of analyzing and proposing techniques to reduce the complexity of designing and producing this kind of circuits.

1.3 Goals

Given the exposed motivation and the description of the problem to address, the strategic goal of this Thesis is to propose a set of contributions that enhance the delay element design for bundled-data asynchronous design templates.

To accomplish this strategic goal, the following set of specific objectives were defined:

1. To develop a model enabling to evaluate voltage scaling effects in delay elements and other parts of a circuit;

2. To evaluate the voltage scaling impact in DEs;

3. To analyze resilient alternative for BD templates;

4. To reduce the need to add delay margins in DEs and combinational logic.

1.4 Originality of this Thesis

This work has its roots in the DE delay margin reduction for bundled-data templates under voltage scaling. The paper [HHS+15] approached for the first time the voltage scaled delay ratio modeling of cells. It also evaluates three classical programmable delay elements usually implemented in bundled-data templates. This paper was presented at the 2015 VLSI Design conference and its results are extended in this Thesis.

Another original contribution is the proposition of how to design a programmable DE to reduce voltage scaled delay ratio variations. A new delay element is proposed focused in this point. Obtained improvements are around 90% better results when compared with other solutions, for selected relevant parameters.

Voltage scaling affects not only delay but may change functionality. It is necessary to guarantee manufactured devices work properly. A brief analysis about delay test complexity is brought here mainly concentrated in small delay defects and on the resilient BD alternative. The necessity of reducing the number of test runs is highlighted.

Focusing on test runs reduction, a novel delay element for resilient BD templates is proposed. The main characteristic of this DE is to reduce process variation impact during DE reconfiguration. The Thesis shows that the reduction in delay margin can reduce the number of test runs or at least can improve test coverage. Delay margins are reduced by at least 67% and this can reach up to 88% reduction in recent technologies. The new DE can be implemented as part of a clock generator for resilient synchronous templates as well.

All Thesis experiments are replicated for three widely distinct commercial technologies to properly validate the research predictions. Two of these are bulk CMOS technologies and the third is based on fully depleted silicon on insulator devices.

1.5 Document Structure

The remaining of this document comprises five Chapters. Chapter 2 presents basic definitions and concepts used in this volume. Next, Chapter 3 proposes an analysis of voltage scaling impacts in delay. The Voltage Scaled Delay Ratio (*VSDR*) model is also discussed in this Chapter. Tree different delay elements are evaluated and a new one is proposed to reduce *VSDR* margin. Chapter 4 brings a procedure for delay fault tests in BD templates, focusing on delay elements characteristics. This Chapter includes the impact of voltage scaling in time and number of test runs. It also shows an alternative to detect small delay defects in BD templates using the resilient BD template. Chapter 5 discusses the peculiarities of resilient BD design template, with target on two-DEs implementation. Alternatively, it brings the proposal of a new DE, designed to reduce delay margins from process variations under dynamic configuration. Finally, Chapter 6 presents a set of final considerations and discusses future work. Note that each chapter in this Thesis presents a revision of the state-of-the-art relative to its content when this makes sense.

2. AN OVERVIEW OF BASIC CONCEPTS

This Chapter provides an introductory overview of several concepts that serve as basis for the work developed in this Thesis. It starts with a discussion of the classical synchronous circuit design process, in Section 2.1. This Section also approaches some contemporary synchronous circuit design techniques based on error resiliency, and addresses some of their characteristic issues. Next, Section 2.2 addresses the problem of designing asynchronous circuits, scrutinizing different ways to design them. The Section also covers an alternative bundled-data asynchronous template, with error resiliency capabilities. Section 2.3 explores some specifically relevant information on the main electronic component addressed in this work, the Metal Oxide Semiconductor (MOS) Field Effect Transistor (FET). The diversity of delay elements for bundled-data design are the subject of Section 2.4. Finally, Section 2.5 brings some fundamental concepts related to the test of Integrated Circuits (ICs), including topics on delay fault tests with focus on the voltage scaling impact on post-silicon tests of synchronous digital ICs. Alongside the discussion, this Chapter establishes a precise terminology employed throughout the rest of the document.

2.1 Synchronous Circuit Design

According to Calazans [Cal98], a digital design allows abstracting input or output signals from physical quantities such as voltage and current into discrete signals with a numerical representation. This characteristic allows increasing the circuit design abstraction level, facilitating the process of complex information handling. Another possible abstraction level includes treating time as a discrete variable that can be associated to the just mentioned digital signal abstraction. If time is taken as a set of discrete equally spaced moments in time domain, the digital signal is called *synchronous*. The usual way to discretize time relies on the use of a periodic control signal called *clock*. The clock simultaneously controls every action in a digital system. When a digital signal does not rely on a discrete notion of time, it is denominated an *asynchronous* signal.

This Section introduces and explains the basic concepts about the classical synchronous approach, based on information representation using synchronous signals. It also explores the synchronous resilient template proposed in the past to improve performance and/or power efficiency of digital circuits.

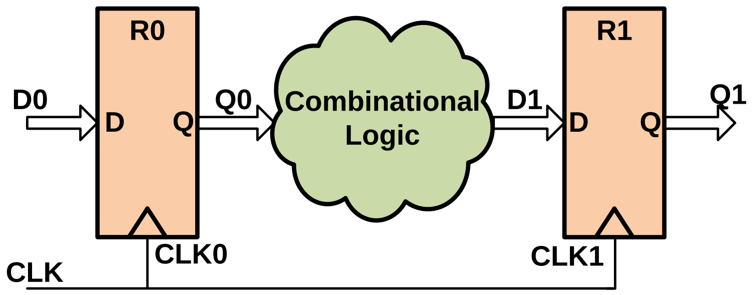


Figure 2.1 – A typical stage of a synchronous circuit.

2.1.1 The Classical Synchronous Circuit Design Approach

A digital circuit that uses exactly one periodic control signal to control all its registers, is a synchronous circuit [RCN03]. The clock simultaneously updates all registers, allowing an orderly data propagation process along all IC modules. Through this simple concept, the synchronous paradigm allows the hardware developer to abstract timing issues almost completely, facilitating the design of ICs enormously. For this reason, most digital designs employ this paradigm, even if locally, in the case of current complex systems.

A conventional synchronous digital circuit usually consists of a set of stages composed by registers that encircle a piece of Combinational Logic (CL), as depicted by the example stage of Figure 2.1. The use of registers controlled by a same clock allows that data present in the output of a register Q0 remain stable between clock ticks, which control their propagation through the CL between two successive clock edges. When the second of two successive edges occurs, data present at input *D1* of register *R1* is stored and propagated to output *Q1*. Equation 2.1 based on [RCN03] allows determining the minimum clock period ($t_{clk(n)}$) for a given circuit stage n, by summing the time the first register *R0* takes to propagate its input to the output ($t_{p_{R(n)}}$), the computation time required by the CL ($t_{p_{CL(n)}}$) and the setup time of the next register ($t_{setup_{R(n+1)}}$ of *R1*). Since the clock signal is global, its minimum period is the minimum time necessary to satisfy all instances of Equation 2.1, for all stages in the circuit.

$$t_{clk(n)} \geq t_{p_{R(n)}} + t_{p_{CL(n)}} + t_{setup_{R(n+1)}} \tag{2.1}$$

Conceptually, a synchronous circuit connects the clock signal to all circuit registers [RCN03]. However, variations caused by the wires used to route the clock, as well as gate and parasitic capacitances can affect the clock propagation delay. *Clock skew* is the generic denomination of the effect that dictates that the clock signal might not reach all registers exactly at the same time.

According to Rabaey et al. [RCN03], the clock skew implies loss of performance and functionality in digital systems. When considering a clock skew effect of value d (in seconds) between two points where the clock connects to registers *CLK0* and *CLK1* in Figure 2.1, the circuit could operate at a higher frequency than previously defined, according to Equation 2.2. However, d also involves adding extra time to maintain the hold time of registers, as Equation 2.3 shows. It is important to highlight the skew uncertainty at each stage [DS94]. Due to long distances, effects such as crosstalk, power distribution and other manufacturing discrepancies can affect the clock skew. These variations in skew result in time margins, which in turn reduce the frequency, causing performance degradation.

$$t_{clk(n)} \geq t_{p_{R(n)}} + t_{p_{CL(n)}} + t_{setup_{R(n+1)}} - d \tag{2.2}$$

$$t_{hold_{R(n)}} < t_{p_{R(n)min}} + t_{p_{CL(n)min}} - d \tag{2.3}$$

Skew effects can be quantitatively significant. Accordingly, Computer Aided Design (CAD) tools try to compensate this effect by inserting buffers or dedicated elements to “deskew” the clock, i.e. to reduce clock skew. However, in recent technologies, the price paid to keep clock skew within tolerable values can be significant as pointed out, for example, in references like [LSGB11], [SRG+01] and [SGB11].

2.1.2 The Resilient Synchronous Circuit Design Approach

Energy consumption is one of the most relevant issues in most current digital systems, from high-performance servers to portable devices. To cope with the energy-performance trade-off it is necessary to employ innovative circuit architectures. One family of such architectural innovations appeared along the proposal of the Razor approach to design synchronous digital circuits [EKD+03, DTP09, KKFK13, FFKP13]. Razor is a family of resilient synchronous techniques proposed by Ernst et al. consisting in adopting speculative operation techniques for circuit design. Originally designed for voltage scaling applications, Razor circuits can operate at a frequency higher than that predicted by the use of worst-case delay margins. Of course, this implies the circuit will eventually incur in timing violations. When such violations do occur, special circuits detect it and trigger the on-the-fly execution of error correction procedures.

As Figure 2.2 shows, using a delayed version of the clock, a shadow latch collects data computed after the main flip-flop. The design of the shadow latch and of the delayed clock guarantee that the computed worst-case delays of the circuit are always respected. If data collected in these two latches differ, the circuit indicates an error and the data stored in the shadow latch replace the contents of the flip-flops in the circuit data flow.

However, the authors of [EKD+03] show that precautions are necessary when computing the delay applied to the clock. If the delay is too small, data collected in the shadow

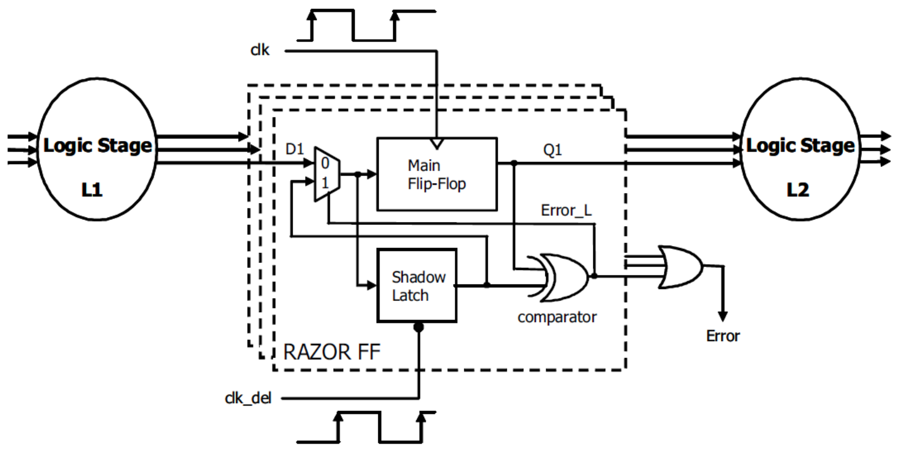


Figure 2.2 – The original Razor design template, as proposed in [EKD+03].

latch can sample the same (wrong) value of the main flip-flop, not signaling the error while the CL is still computing. If the delay is longer than the ideal, the latch may loose computed data and get a next data, which also causes incorrect operation.

Razor can detect incorrectly stored data, but the clock dependency makes the circuit sensitive to skew. Metastability on the main flip-flop, possibly caused by an early clock edge (with regard e.g. to too long a computation taking place in the Logic Stage *L1*) can also propagate to the error signaling gate, maybe taking the entire circuit to an unknown state [HMH+15]. Thus, it is natural to consider asynchronous design paradigms as an alternative to Razor. Studies such as the ITRS [ITR] indicate a continuous trend of migration from the use of synchronous to asynchronous signaling. The next Section briefly explores the scene of asynchronous circuit design as it stands today.

2.2 The Asynchronous Circuit Design Approach

Asynchronous designs, unlike synchronous ones, do not abstract time, obliging designers not only to reason about the logic, but also to consider a large amount of time constraint effects during design [SF13]. Asynchronous circuits employ, instead of clock tree, handshake processes between registers to perform synchronization, communication and sequencing of operations [BOF10, SF13]. Thus, asynchronous circuits can potentially lead to lower power consumption, higher operation speed, less emission of electromagnetic noise, more robustness towards PVT variations, better composability and modularity and avoidance of global clock distribution and the associated clock skew problems.

In a more abstract view, the control of asynchronous circuits relies on token handling. Consider for example Figure 2.3. For some data to be stored and be available at the output *Q1* of register *R1*, controller *C1* requires two tokens. One comes from the predecessor circuit that informs data is stable in *D1* to be stored. The other comes from the next

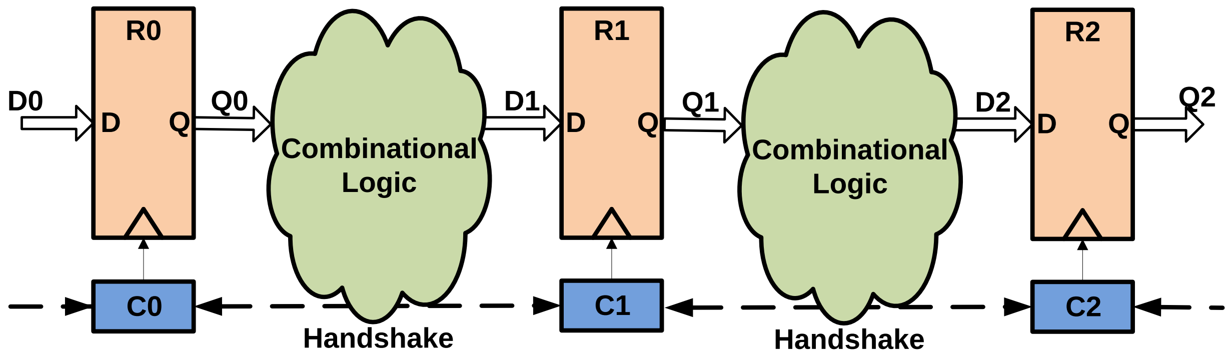


Figure 2.3 – An example of an asynchronous pipeline circuit segment.

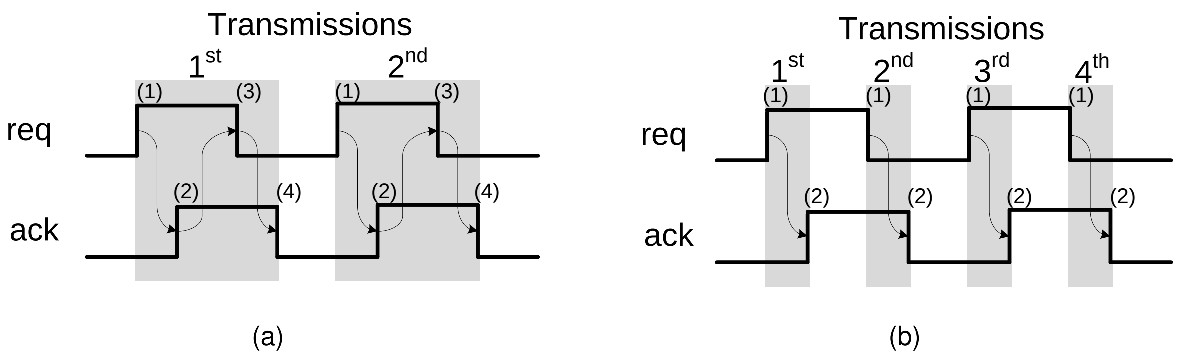


Figure 2.4 – Handshake protocols: (a) 4-phase and (b) 2-phase.

control circuit, *C2*, which indicates data in its register was consumed (*R2* being thus free to receive new data).

Considering the manipulation of logic tokens as a criterion, asynchronous circuits can be grouped into two major classes: Bundled-Data (BD) and Delay Insensitive (DI) circuits. In the first class, control tokens flow through dedicated (control) channels only, which run parallel to the CL blocks and the data paths. In the second class of circuits, data available tokens usually mingle with data in data paths (through the use of special DI codes), while the data consumed token usually employs a dedicated control channel.

2.2.1 The Bundled-Data Asynchronous Circuit Design Approach

The term Bundled-Data (BD) refers to a situation where data signals use ordinary Boolean encoding to carry information, and where separate request and acknowledge lines control the flow of data as a bundle [SF13]. According to the use of level or transition signaling, there are two types of communication protocols useful to pass tokens between controllers: 4-phase and 2-phase protocols. Figure 2.4 illustrates these two protocol types.

In the 4-phase protocol, shown in Figure 2.4(a), both the request (*req*) and acknowledge (*ack*) wires use Boolean levels to carry information and 4 summarizes the number of

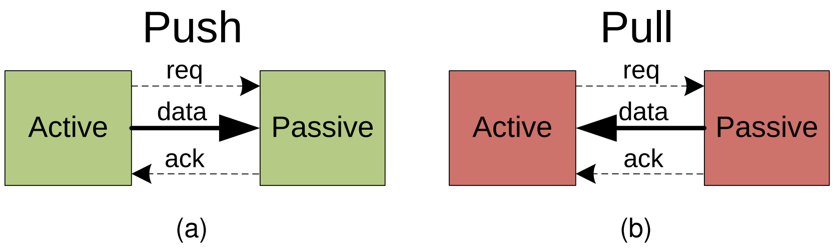


Figure 2.5 – The organization of (a) Push and (b) Pull data channels in BD asynchronous designs.

steps taken for one information exchange action. Initially assume, without loss of generality, that both wires are at ’0’. To pass a token, the protocol requires four steps:

1. The sender issues data and sets *req* to ’1’ (for example, in Figure 2.3 *C0* sends a token to *C1*);
2. The receiver absorbs the data and sets *ack* to ’1’;
3. The sender responds by taking *req* to ’0’;
4. The receiver acknowledges this, by making *ack* low (in Figure 2.3, *C1* sends a token back to *C0*).

The main disadvantage of using this kind of protocol are the mandatory return-to-zero phases, which cost time and energy. 2-phase protocols, shown in Figure 2.4(b) allow the use of only one transition of each signal to transmit data. In this protocol, there is no difference between ’0’ $\rightarrow$ ’1’ or ’1’ $\rightarrow$ ’0’ transitions, because both represent a “signal event”. Thus, the protocol has only two steps:

1. The sender issues data and changes *req* (*C0* sends a token to *C1*);
2. The receiver absorbs the data and changes *ack* (*C1* sends a token back to *C0*).

Another classification for BD circuits refers to how controllers command the send/receive data channel. Each of these acts as an active or passive entity. An active entity initiates a communication, sending or receiving data from another controller, the passive entity. Thus, when data come from the side of the active entity, it constitutes a *push channel*, i.e. data follow the same direction as *req* (see Figure 2.5(a)). If the producer is on the passive entity side, this characterizes a *pull channel*, where data flow with the passive entity acknowledge signal, according to what appears in Figure 2.5(b). The next characteristic of BD circuits to explore is the requirement of delay elements.

In bundled-data circuits, the token that indicates the existence of new data to register must be synchronized to the data arrival itself. This usually means that the token that indicates the existence of new data to register must arrive simultaneously or a little after the

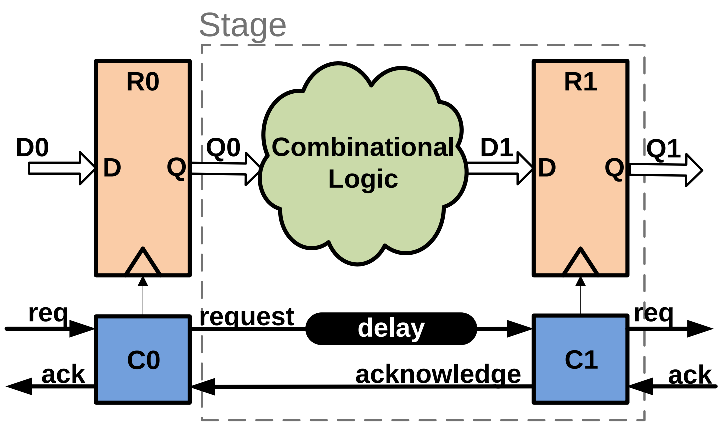


Figure 2.6 – Bundled-data stage segment using a push channel, with a delay element in the request line

data themselves are stable at the input of the next register. Since the logic that generates the control tokens is usually simpler (and thus faster) than the logic that processes data, asynchronous circuits based on BD templates require an element to delay the request arrival at its destination for at least a time as long as the delay of the CL, here called a delay line or delay element (DE). The need to determine if controllers work with push or pull channels allows defining which control signal must have a delay matched to the CL in question. In case communication occurs by push channels, the delay must be in the request signal path, while in pull channels it must be in the acknowledge path. In this work, without loss of generality, all examples and discussions will assume the use of push channels. Figure 2.3 shows an example of an asynchronous pipeline segment where communication takes place using a push channel. This results in the circuit segment that Figure 2.6 depicts.

Since the delay element must take at least as long as the CL delay, delay elements must cover all situations arising in a real circuit, i.e. their design needs to consider the worst-case delay of the CL. Thus, even if the circuit can operate faster, delay elements can affect BD circuit performance significantly.

2.2.2 The Delay Insensitive and Quasi Delay Insensitive Asynchronous Circuit Design Approaches

Delay Insensitive (DI) circuits are asynchronous circuits where delay values on neither wires nor gates influence correct operation. According to Martin [Mar90], DI circuits are in fact a very limited class of circuits. Their construction can only rely on the use of inverters and C-Elements as basic components. C-Elements are sequential components that serve as the basis of many, if not most, asynchronous circuit templates [BOF10]. Table 2.1 displays the behavior of a basic 2-input C-element. In this component, output C can only change when A and B have identical values. Distinct functionalities exist, but C-elements al-

Table 2.1 – Behavior of a basic 2-input C-element with inputs A and B and output C.

A	B	C_n
0	0	0
0	1	C_{n-1}
1	0	C_{n-1}
1	1	1

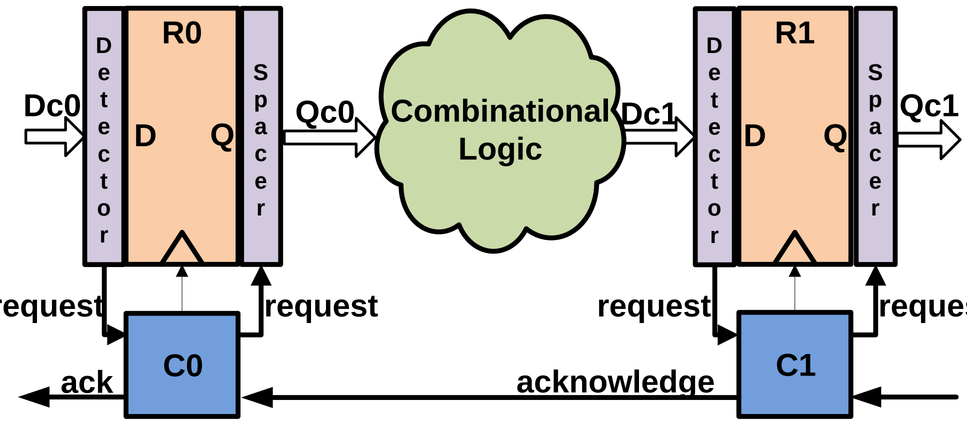


Figure 2.7 – A partial view of a quasi delay insensitive pipeline.

ways present one or more input combinations where a memory effect is present mandating that the output does not change from a previous value, no matter which was this value.

To overcome the limitations of DI circuits the class of Quasi-Delay Insensitive (QDI) circuits relax the requirements on behavior, allowing that certain wires in the circuit be subject to timing constraints. These wires are some of those that fork (i.e. fan out from a source to more than one destination). The timing requirement imposed on these selected (forking) wires is that the time difference to reach every end of the fork be limited to a controlled amount. These wires are accordingly called *isochronic forks*. It has been shown [BOF10] that this relaxation on the DI requirements produces a new unconstrained class of circuit design methods and that the imposed isochronic fork timing requirement can be fulfilled with manageable design resources.

DI and QDI circuits rely on specific data codes (DI codes) that embed the control token for the communicating controller stage. Consider Figure 2.7, where CL blocks recognize and process data using DI codes. The request signal is locally extracted from incoming data using completion detector circuits (note detector in the Figure) [SF13], which identify the validity of the data to register. Similar to what happens in BD templates, QDI templates can employ either 4-phase or 2-phase communication protocols.

In 4-phase protocols, data conversion is often somewhat simpler. Again, without loss of generality, this thesis focus solely on the use of 4-phase protocols that rely on the use of a specific DI code. In this code, a pair of wires (*d.t* and *d.f*) encode a single bit, as Table 2.2 shows. To send a logic value ‘0’, the false wire (*d.f*) should be ‘1’ and true wire (*d.t*) must be ‘0’, and to send a logical value ‘1’, the true wire is ‘1’ and the false wire is ‘0’.

Table 2.2 – The structure of a single bit dual-rail code.

Information/Wire values	d.t	d.f
Spacer	0	0
Valid '0'	1	0
Valid '1'	0	1
Not used or invalid	1	1

There is also a spacer, an invalid codeword that indicates when there is no data in the CL, identified when the two wires are ‘0’. If an n-bit word of data has to be communicated, $2 * n$ are used. This code is accordingly called *dual-rail*.

To operate with a 4-phase protocol, first the CL block need to be empty, i.e. contain the spacer (*d.f*s and *d.t*s are all ‘0’) in all of its inputs and outputs and the next stage acknowledge must be at ‘0’. The sequence of steps is similar to that in BD circuits and follows the sequence:

1. The sender puts data at the input of CL. When the Detector in next controller identifies all individual wire pairs are distinct ($d.f \neq d.t$), it asserts *request* to ‘1’ (*C0* sends a token to *C1*);

2. The receiver then absorbs the data and sets *acknowledge* to ‘1’;

3. The sender puts the all-spacer value in CL. When the next controller identifies all wires at ‘0’, it sets *request* to ‘0’ (*C0* sends a token to *C1*);

4. The receiver acknowledges this by setting *acknowledge* to ‘0’ (*C1* sends a token back to *C0*).

Several other protocols are available, such as 2-phase Non-Return to Zero (NRZ), single-rail, etc. [BOF10]. There are even proposals of 2-phase Return to Zero (RZ) protocols such as the one Elrabaa suggests in [Elr11]. This Thesis restricts attention mostly to circuits that employ the 2-phase NRZ protocol.

The use of particular codes described here or other DI codes mandates that QDI circuits cause large area overheads. Studies pointed by [SGY+09] and [BOF10] show the total area of a QDI circuit can reach up to four times that needed to implement an equivalent synchronous circuit. Because QDI circuits often require specific cells not provided in foundry libraries, this template usually implies difficulties to its implementation. Full custom cells and the lack of commercial design tools support are obstacles to unleash the QDI advantages.

On the other hand, one of benefits of designing circuits in QDI is the low sensitivity to PVT variations. Studies like [CL10, CCGC10, HCGC15] indicate subthreshold operations without errors. However, some attention needs to be applied on C-elements design to guarantee glitch free operation, the weakness point of QDIs [GHMC14, MGHC14].

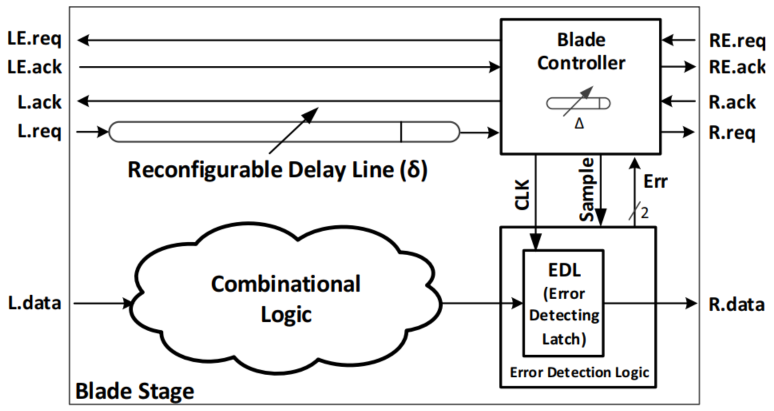


Figure 2.8 – The Blade template [HMH+15].

2.2.3 The Resilient Asynchronous Circuit Design Approach

Just as Razor was designed to reduce margins in synchronous circuits operation, there are asynchronous templates aimed at speculating on data computation completion. One of them, Blade [HHC+15, HMH+15], is a speculative model developed to promote resiliency in asynchronous BD circuits. In Blade, margins imposed on Delay Elements (DE) are reduced by an evaluation mechanism that allows that computed data propagates earlier than worst-case situations.

Figure 2.8 presents a typical Blade template stage. Assume a three-stage Blade pipeline completely empty. When data is available in a predecessor stage, this is sent to the *L.data* bus of the middle stage, and simultaneously *L.req* switches, signaling new data are available. While the middle Combinational Logic (CL) computes the stage data, the *L.req* signal is delayed by the δ DE. At the end of δ, the middle stage Blade Controller receives the request and creates two tokens. The first communicates with the Error Detector Logic (EDL) through the *CLK* signal making the former transparent. The EDL then starts monitoring the CL data bus. From this moment on, data are made available to *R.data* (to the successor stage) and *R.req* is signaled. The second token asks the predecessor stage through the *LE.req* signal if there was an error identified at that stage. The predecessor stage signals *LE.ack* when the data has been correctly sent. Meanwhile, the EDL is transparent during the entire duration of Δ, after which the total CL propagation time ends, including all delay margins. Blade gains performance with these two steps occurring in parallel.

After Δ is through, the mentioned two tokens are expected to have arrived. If both are available, the Blade Controller deactivates *CLK* and asks if there are errors in the EDL (through the Sample signal). At the same time, the controller also indicates the end of data capture to the predecessor stage through *L.ack*. The controller waits for error signaling on the *Err* dual-rail signal. If an error occurred, the controller restarts Δ to ensure the correct data arrives at the next stage. The *RE.req* signal may arrive at any time during this step, but the *RE.ack* response signal is only sent when there are no more errors.

According to the authors of [HHC+15, HMH+15], the ideal error rate for the best Blade performance is around 30%. For this reason, the use of programmable Delay Elements is unavoidable. Some variation in δ is acceptable without incurring performance costs. However, margins need to be minimized to improve performance, since the response to the predecessor stage stating that new data can be sent *L.ack* only happens after $\delta + \Delta$.

Parallel work like Sharp [WK17] and a new version of Blade are under development. However, these versions apply more delay elements (four in Sharp and three in the new Blade template) which adds more margins for each DE, possibly impacting negatively the performance of these templates.

2.3 Metal Oxide Semiconductor Field Effect Transistors (MOSFET)

Complementary Metal Oxide Semiconductor (CMOS) is a technology that typically relies on the use two transistors types (PMOS and NMOS) with complementary behavior. In a typical CMOS circuit, a pull-down network uses NMOS transistors, and a pull-up network uses PMOS transistors. The interconnection of these two networks creates an operational digital circuit.

Figure 2.9 displays a cross section of a traditional Metal Oxide Semiconductor Field Effect Transistor (MOSFET). It is possible to note that this transistor has four terminals: *gate*, *drain*, *source*, and *bulk* (or *body*). In the case of n-channel (or NMOS) transistors, the source and drain terminals present strong doping concentration (+), with chemicals that have electrons in excess (*n*). The channel below the gate is part of the bulk region, which is weakly doped ($-$) with elements with holes in excess, or lack of electrons (*p*). Without application of a voltage between the gate and bulk terminals, no free electrons exist between source and drain. Consequently, no current flows between these. However, when a positive voltage exists between gate and bulk terminals, electrons are attracted to the bulk region just below the gate, creating a channel where there is free electrons between the source and drain terminals and current flows between these, if a voltage between source and drain also exists. For p-channel (or PMOS) MOSFETs these relations reverse with regard to the dopant types, requiring the application of a negative voltage between gate and bulk to form a channel of holes. The gate capacitance (C_g) depends on transistor operation zone and it is discussed in details in Section 2.4.

MOS transistors can work as a current source between the drain and source terminals under certain conditions on the voltages between its terminals. Usually, transistor analysis ignores the bulk terminal, because this often connects to the source transistor terminal.

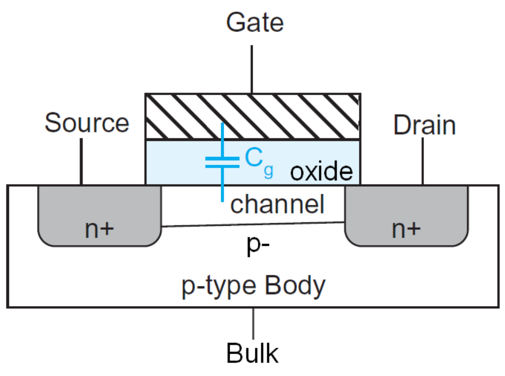


Figure 2.9 – A traditional n-channel MOSFET structure, highlighting the transistor terminals, based on [WH11].

The electric current flowing between drain and source terminals (I_{ds}) can be determined by a set of parameters:

- Carrier mobility (μ);
- Oxide capacitance (C_{ox});
- Channel width (W);
- Channel length (L);
- Voltage between gate and source (V_{gs});
- Voltage between the terminals source and drain (V_{ds});
- Threshold voltage (V_t) characteristic of the component.

Equations 2.4 available e.g. from [WH11] tell about these relations.

$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t & \textit{Cutoff} \\ \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot \left(V_{gs} - V_t - \frac{V_{ds}}{2}\right) \cdot V_{ds} & V_{ds} < \left(V_{gs} - V_t\right) & \textit{Linear} \\ \frac{1}{2} \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot \left(V_{gs} - V_t\right)^2 & V_{ds} \geq \left(V_{gs} - V_t\right) & \textit{Saturation} \end{cases} \tag{2.4}$$

Note that the last of these equations shows how the transistor behaves as a current source: if V_{gs} is kept fixed, the current does not depend on V_{ds}. As these equations show, there are three regions of operation for a MOSFET: *cutoff* (when no free charges exist in the channel), *linear* or *triode* (when a few free charges exist in the channel), and *saturation* (when a channel is fully formed with a large number of carriers). However, these equations apply mostly to older technologies, where process variations and short channel effects were hardly significant.

2.3.1 MOSFET Advanced Technology Effects

According to authors like Rabaey et al. [RCN03] and Weste and Harrys [WH11], effects like *channel-length modulation* and *velocity saturation* interfere with the drain to source current (I_{ds}) in *linear* and *saturation* regions. In addition, *subthreshold* and *weak-inversion conduction* cause exponential reductions in I_{ds} when V_{gs} is lower than V_t (in the *cutoff* region). This last effect is due to current that flows through the p-n junction between the source/drain diffusions and the bulk. The leakage current for subthreshold operation brings an important contribution to static power dissipation in current circuits. Its contribution increases significantly with V_{ds} caused by drain-induced barrier lowering (DIBL) specially in short-channel transistors [WH11]. In this specific chart, the drain to source current I_{ds} is normalized to transistors with width equal 1 μm.

Another current leakage source in new technologies can be seen on the gate terminal. This effect is caused by electron cloud sharing charges between gate and channel with insulator thinner than 15$\sim$20 Å.

Finally, recent technology transistors present one more leakage source, caused by p-n junctions. In older technologies, the voltage across this junction guarantees these diodes do not become forward biased. Today, reverse-biased diodes still conduct a small amount of current. High halo doping used to increase V_t to alleviate subthreshold leakage causes a reverse-bias between high doped drain/source and bulk (called Band-To-Band Tunneling or BTBT). Another junction leakage, called Gate-Induced Drain Leakage (GIDL) is the drain to bulk current caused by the voltage between gate and drain but is more sensible when the drain has higher voltage than the gate.

2.3.2 MOSFET Variability

The IC manufacturing process is unfortunately not an ideal one; in other words, this process is sensitive to variations on fabrication and/or operation. Weste and Harris [WH11] define four variability sources: systematic, random, drift, and jitter. The first two are considered static variations, while the last two are dynamic variations.

Systematic variations have a quantitative relationship with a source. For example, polysilicon gates may systematically be etched narrower in regions of high polysilicon density, differently from regions with lower density of polysilicon lines. This variability can be modeled and compensated at design time.

Random variations include those that are truly random, those whose sources are not fully understood, and those that are too costly to model. Random variations do not change with time, so they can be nulled out by a single calibration step after manufacturing.

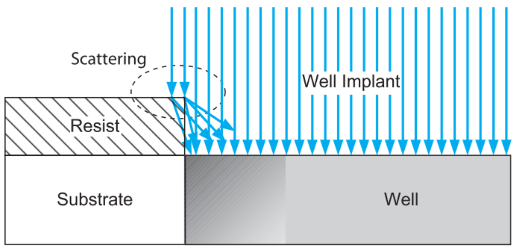


Figure 2.10 – Well-edge proximity effect increasing the doping level near the edge of the well [WH11].

Drift, notably aging and temperature variations, change slowly with time as compared to the operating frequency of the system. Compensation circuits can again null them, but such circuits must recalibrate faster than the drift occurs.

Jitter, often from voltage variations or crosstalk, is the most difficult cause of mismatch. It occurs at frequencies comparable to or faster than the system clock and therefore may not be eliminated through feedback.

Channel length variation is often expressed as a percentage of the nominal channel length, because delay variations are proportional to this percentage. According to Weste and Harris [WH11], it is possible to estimate a target σ/μ = 4% for this variation. That corresponds to $0.04{\cdot}X\ \mu m$ standard deviation of channel length in an $X \mu m$ process. As an example, this corresponds to variations of around 12% in channel length in a 35nm process. The number and location of dopant atoms implanted in the channel or halo region determine the threshold voltage V_t. This ion implantation is a stochastic process, leading to random dopant fluctuations that cause V_t to vary [WH11]. However, the variations have become large in nanometer processes, because the number of dopant atoms is small. To reduce this effect, it is possible to use high-V_t transistors with higher effective channel doping or high-κ metal gate transistors allow using lower halo doping. According to Itoh in West and Harris [Ito09, WH11], the average V_t variation is around 1.0$\sim$2.5 $mV * \mu m$ for a 45 nm process and he predicts a lower bound of 0.4$mV * \mu m$ in future processes.

Oxide thickness shows an average variation of around 0.1 Å in a 10 Å oxide layer. Because of this, the variations cause minor effects when compared with channel length and threshold voltage.

Finally, layout effects can cause problems in threshold voltage for transistors near the edge of a well, caused by scattering in well implant process step as Figure 2.10 shows. Under 65 nm process, transistors may have delays up to 10% higher than those farther from the edge [WH11]. Another layout effect, the mechanic strain, can also change carrier mobility. Therefore, this could increase the ON current of transistors or cause more leakage.

When all current effects are considered, I_{ds} could change dramatically. Equation 2.5 shows the proportional change on the current formula for the cutoff and saturation regions. Here, α is the *velocity saturation index* and is determined by curve fitting measured I-V data (replace the square value) and usually is between one and two, n is the subthreshold exponential effect, ϑ_T is the thermal voltage from thermodynamics ($\vartheta_T = k \cdot T/q$ where k is the Boltzmann's constant, T is the absolute temperature, and q is the charge of an electron).

$$\begin{array}{rc} I_{ds} \propto \frac{W}{L} \cdot e^{-\frac{V_t}{n \cdot \vartheta_T}} & \textit{cutoff} \\ I_{ds} \propto \frac{W}{L} \cdot (V_{gs} - V_t)^{\alpha} & \textit{saturation} \end{array} \tag{2.5}$$

Including the length (L) and V_t variations, more changes occur on nominal I_{ds} values as Equation 2.6 shows. According to Tehranipoor et al. [TPC11], process variations can produce V_t variations of up to 30%. Tsividis in [TM11] deduces V_t is a function of V_{ds}, temperature (T), transistor dimensions (W and L) among others, including the specific foundry process. More details about threshold variation will be discussed in Chapter 3 of this Thesis.

$$\begin{array}{rc} I_{ds} = I_{ds,nominal} \cdot \left(1 - \frac{\Delta L}{L} - \frac{\Delta V_t}{n \cdot \vartheta_T}\right) & \textit{cutoff} \\ I_{ds} = I_{ds,nominal} \cdot \left(1 - \frac{\Delta L}{L} - \frac{\alpha \cdot \Delta V_t}{V_{gs} - V_t}\right) & \textit{saturation} \end{array} \tag{2.6}$$

Using 1500 runs of Monte Carlo simulation and assuming 0.04 for length standard deviation (σ/μ) and 25 mV for threshold standard deviation, Weste and Harris [WH11] calculate the variation over two cases: $V_{gs} = GND$ and $V_{gs} = V_{DD}$. Results show variation changes current on the first case by six times while changing current on the second case by only 40%. This example illustrates how much variations can affect transistor behavior.

2.3.3 IR-Drop effects

One of the effects caused by the high circuit integration capacity is IR-Drop. The reduced dimensions of power rails and the extensive switching activity of transistors produce significant power network noise. In some cases, the variations may reach more than 30% [TPC11], producing significant impact on cell delays at certain IC regions.

According to [WH11], the power distribution of a chip consists of metal wires on the chip responsible for supplying power to the circuits inside it. It also includes bypass capacitors to amortize instantaneous current effects targeting the chip requirements. As main requirements, the design of a power supply network has to:

- Maintain a stable voltage with little noise;
- Provide average and peak power demands;
- Provide current return paths for signals;
- Avoid electromigration and self-heating;

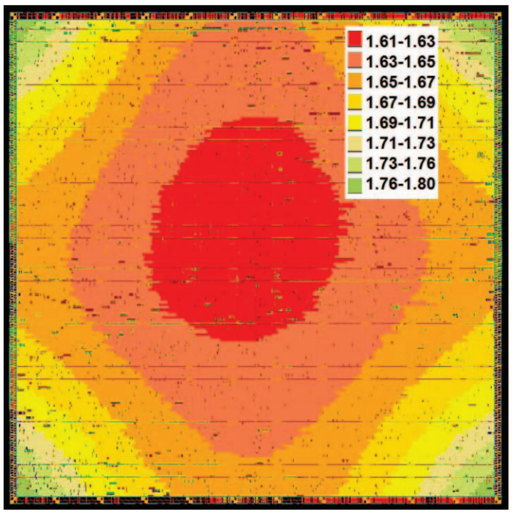


Figure 2.11 – IR Drop plot of a test pattern applied to wb_conmax benchmark [TPC11].

- Consume little chip area and wiring;
- Be easy to layout.

Real ICs impose constraints on power supply distribution. One of the most important is related to noise margin. This noise comes mainly from dynamic current during gate switching and leakage current from gates in stable states [WH11]. Despite the fact that metals have good conductive characteristics, long wires with small width, as seen in new technologies, increase the resistance. The current (I) consumed by gates multiplied by the intrinsic resistance (R) in power wires results in a voltage drop known as IR-Drop.

Ideally, the IC designer intends to design power plans with a noise margin that must not exceed $\pm$10% of the supply voltage [WH11]. However, according to research presented in [TPC11], IR-Drop in emerging technologies could reach up to 36%. This impact comes from wire width reduction, higher integration capability and leakage current increases. Figure 2.11 illustrates the IR-Drop effect in a die designed for 1.8V supply voltage in a 180nm technology [TPC11].

Asynchronous circuits are known to reduce the IR-Drop effect because the switching activity does not occur at pre-determined points in time. In these, switching occurs driven by data availability and stages are free to propagate information at their one pace. So, coincident transitions are rare when compared with synchronous circuits. However, if a stage is bigger enough, it could produce considerable IR-Drop effects. Cortadella et al. [CLM+16] corroborate this by showing a maximum IR-Drop of around 17% in a relatively small die ($1.75mm \times 0.7mm$) design in 65nm technology implementing elastic circuits.

Table 2.3 – Approximation of intrinsic MOS gate capacitance [WH11].

Parameter	Cutoff	Linear	Saturation
C_{gb}	$\leq C_0$	0	0
C_{gs}	0	$C_0/2$	$2/3 \cdot C_0$
C_{gd}	0	$C_0/2$	0
$C_g = C_{gs} + C_{gd} + C_{gb}$	C_0	C_0	$2/3 \cdot C_0$

2.4 Delay Elements

The time constant is the basic theory for the delay elements development. It is usually defined by time to charge/discharge passive devices such as capacitors, inductors considering resistances. As the IC technology is grounded on capacitive characteristics and current handling, the time constant in MOSFET is based on resistor/capacitor (RC).

Besides dedicated capacitors devices, capacitances can be achieved using MOSFETs. The classic gate capacitance (C_0) of such transistors is defined by Equation 2.7. C_{ox} is the technology oxide capacitance, W is the transistor width and L is the channel length. However, gate capacitance does not have a constant value during operation, because of the capacitances arising from the overlapping of the gate to source (C_{gs}) and drain (C_{gd}) diffusions in new technologies. Also, there is the capacitance between gate and bulk (channel), called C_{gb}. Thus, depending on the operating region of the transistor, the capacitance of each terminal is changed as Table 2.3 shows. Despite the influence of the operating regions, it can be seen that the total gate capacitance C_g vary between $2/3 \cdot C_0$ and C_0.

$$C_0 = C_{ox} \cdot W \cdot L \tag{2.7}$$

Resistances can also be directly implemented using MOSFETs, as the transistor can act as a resistance while handling current. Unlike resistors that occupy considerable areas, besides causing similar effect, transistors also allow adjustment in the produced current, in a way that allows using MOSFETs to build adjustable delay elements.

From the above discussion, it should be clear that it is possible to implement a delay element using only transistors, reducing area and power consumption. This Section presents two of the most common architectures of delay elements implemented in digital ICs, based on either digital cells or on current-starved inverters.

2.4.1 Digital Cell Delay Elements

The use of digital cells to generate delay elements is very common in digital designs. Because of the easy inclusion and full compatibility with physical synthesis tools,

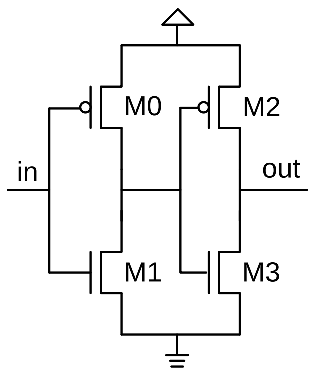


Figure 2.12 – Gate delay element.

these delay elements achieve popularity in asynchronous designs as well as in clock generation and distribution networks.

To adjust delay values, topologies handle the sum of time constants. Each gate acts in two ways: as a capacitor in inputs and as a current source (or resistor) in outputs. The gate input behaves as a capacitor with the transistor gate capacitance previously discussed. The output acts as a resistor using the transistor current and the supply voltage. Two inverter gates connected in series as shown in Figure 2.12 define the time constant as the time to charge/discharge input capacitances using the current produced by some previous gate output. As Equation 2.8 shows, the supply voltage (V_{dd}) divided by the current ($I_{ds_{tot}}$) produced in transistors *M0* or *M1* defines the *R* constant and the $M2+M3$ gate capacitances ($C_{g_{tot}}$) together establish the *C* constant. As two inverters are connected in series, the time is defined by the sum of two *RC*s.

$$t_p = \frac{V_{dd}}{I_{ds_{tot}}} \cdot C_{g_{tot}} \tag{2.8}$$

Clearly, other gates can be used to implement a delay element. Usually, programmable DEs employ multiplexers to allow recombination paths. This kind of architecture also allows manipulating the delay of each edge, changing the selected input in specific gates like NORs or NANDs. In these, it is possible to increase the fanout capacitance according to the selected input.

2.4.2 Current-Starved Inverter Delay Elements

The Current-Starved Inverter (CSI) uses a couple or more of control transistors in series, which allows handling the current flowing through an inverter, changing *R* of the time constant product *RC*. Control is provided by a bias voltage (V_{bias}) from a current mirror manipulated by transistor sizing and association.

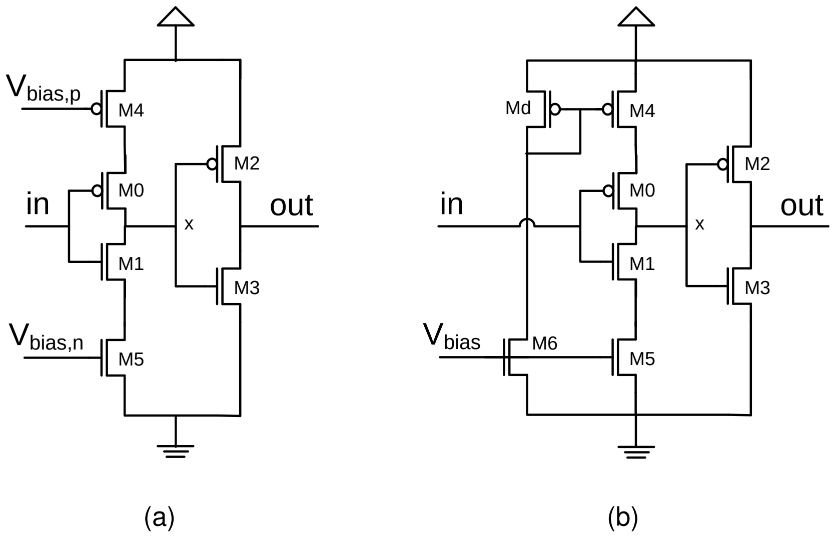


Figure 2.13 – Current-starved inverter topologies: (a) using two independent V_{bias} and (b) using a single V_{bias}.

Figure 2.13(a) displays a CSI where the voltage $V_{bias,p}$ and $V_{bias,n}$ respectively control the current of M4 and M5, which are used to charge the *M2* and *M3* gate capacitors. A variant of this design can be seen in Figure 2.13(b), where only one bias voltage handles both transistors *M4* and *M5* based on the current mirror formed by transistors *M6* and *Md*. The *Md* diode connection allows to transform the current passing in *Md* in voltage.

As current could be changed by transistor sizing and multiplied or divided by transistor associations (parallel or series), there exists a vast amount of possible configurations. Most of these focus on manipulating the current provided by a reference current source using transistor size (width or length) and parallel associations [HHM+15, Bak10]. Others do not employ a current source, using only transistor sizing to change the current [MNS03, MNS05, Bak10].

The graph in Figure 2.14 represents the voltage at node *x* of the CSI in Figure 2.13(b). As it is possible to see, this wave tends to assume a triangular shape as the delay increases. Therefore, it cannot be used directly in digital circuits. Accordingly, the delay elements based on CSIs use conventional inverters (formed by *M2* and *M3*) to restore the digital values properly.

However, the bias performed by voltage tends not to generate linear delay steps to a linear V_{bias} increase. This is due to the quadratic characteristic shown in Equation 2.4 of the drain to source current (I_{ds}).

Other analog circuits to provide Delay Elements are indeed available. However, analog DEs are usually designed for specific delays and need to be readjusted to maintain it constant including under process, voltage and temperature variations. Also, the sensitivity of these devices to noise imposes placing them far from digital circuits. In new technologies,

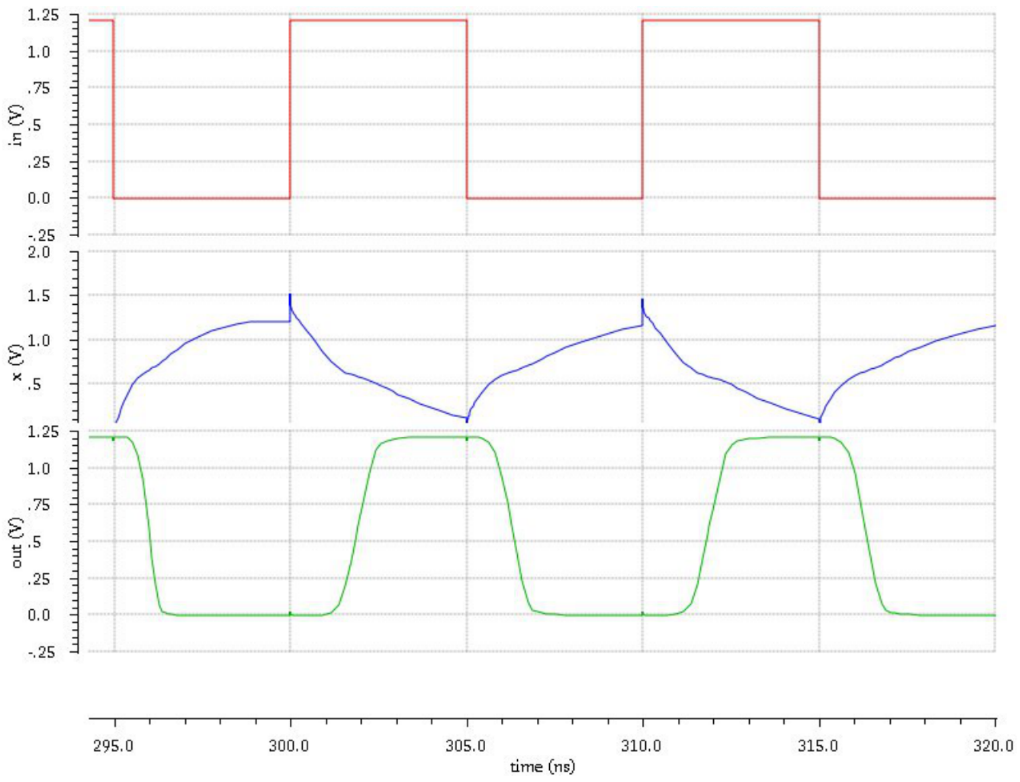


Figure 2.14 – Graphs depicting the transient simulation of Figure 2.13(b) current-starved inverter.

wire delays have significant relevance [WH11]. Thus, a distance large enough could produce wire delays larger than DE itself.

2.5 Test on Integrated Circuits

According to Thernipoor et al. [TPC11], test is a process used to identify ICs containing imperfections or manufacturing defects that may cause failures by applying test patterns to circuits and analyzing their response. The main idea in testing circuits is to guarantee it works correctly before delivering it to costumers. However, the test cost needs to be considered, because in VLSI designs testing accounts for around 30% of the total cost and many companies claim that 50% to 60% of the costs of a circuit go into manufacturing tests. Thus, reducing test cases is essential, but this may imply coverage degradation. In addition, newly emerged defect sources such as small-delay defects could significantly impact the test. To properly understand the functioning of test procedures, some definitions are required.

A *defect* is described as an unintended difference between the fabricated circuit and its design [BA04]. Some kinds of defects include: gate-oxide shorts, missing contact windows, oxide break-down, metal opens, contact degradation, etc. These can be produced during the manufacturing process or be due to design or layout issues. When the defective device produces an incorrect output this is called an *error* or a *failure*. The *fault* is a representation of a defect at an abstracted functional level [TPC11].

Some of the classical fault models are: *stuck-at*, *bridging* and *delay*. The *stuck-at* fault model is the most commonly used fault model to detect nodes between gates that are assumed as permanently connected to ground or to the supply voltage. The *bridging* fault model is used to identify shorts between a group of signals usually inside gates, i.e., at the transistor level. They are mainly caused by paths or devices close to each other. The *delay* fault model covers performance issues, which assume that the faulty circuit element makes signal propagation slower than expected. There are two main delay fault models: *transition-delay* and *path-delay*.

The *Transition-Delay Fault* (TDF) model tests the delay propagation in every node in the circuit. As there are two different levels for each signal, two different faults are possible: *slow-to-rise* or *slow-to-fall*. As each node needs to be tested, this model takes advantage of stuck-at pattern generation. However, because of the existence of two different fault types, the number of faults is bounded by twice the number of nets in circuit [TPC11]. The *Path-Delay Fault* (PDF) model, on the other hand, assumes that there is a cumulative delay defect along a combinational path. In part, this is similar to TDF with two different faults possible, (*slow-to-rise* or *slow-to-fall*). However, instead of only covering the node transition, this model covers all paths that make the node change. Thus, if two gates affect a node, only one is necessary to test it in TDF while both of them are tested in PDF. The number of faults is bounded not only by twice the number of nodes, but by twice the number of all paths in the circuit [TPC11].

For delay fault test, both TDF and PDF require a pair of vectors (V_1, V_2) for each pattern. To guarantee a *slow-to-rise* or a *slow-to-fall* coverage, a reset/set statement needs to be applied firstly (V_1). Usually, the V_1 vector is loaded in CL for longer time (at a lower clock frequency) to ensure a reset/set state in the evaluation path [TPC11]. Only after that the second vector (V_2) is loaded to CL using an at-speed (nominal frequency) or a faster-than (higher frequency) clock. If the node or path under evaluation is compromised, the result will show the reset/set value instead of the expected transition (rise/fall).

All these models are used in Automatic Test Pattern Generation (ATPG) to generate patterns according to the need and will be applied in the Circuit Under Test (CUT). So, Design For Testability (DFT) provides the necessary infrastructure to make circuit test easier and more efficient. The most common methodologies used in industry are *Ad hoc*, *Built-In Self-Test (BIST)* and *Scan-Based Design*.

Ad hoc tests were the first method for DFT techniques [WWW06]. This method only targets the circuit portions difficult to test, adding circuitry to improve controllability and observability. In this case, internal nodes can be accessed directly, generally under the control of multiplexers.

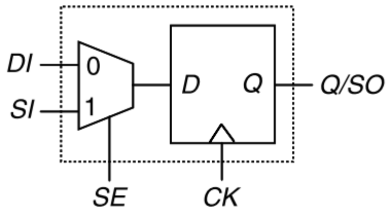


Figure 2.15 – The structure of a Muxed-D flip-flop [WST10].

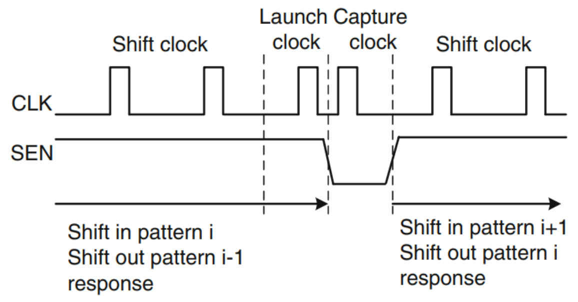
BIST is a technique that includes the test logic inside the own chip to detect faults. BIST has become a promising solution to VLSI designs in recent technologies [URV11]. The BIST controller is in charge of test pattern generation (TPG) and may contains a ROM where expected results are stored. When the BIST controller requires an in-circuit test, TPG applies the patterns in the CUT. The result is compared with the signature at the ROM and, if results differ, the circuit is faulty. However, BIST provides poor coverage of the longest circuit paths [WST10]. The interdependence between previous and current vectors needed to delay evaluation cannot provide a good coverage without a large run of vectors.

On the other hand, *Scan-Based Design* provides a dedicated serial path to load test patterns and collect the test results. This DFT design replaces classical flip-flops, latches or other register structures by scannable ones, interconnects them serially and adds test interfaces scan-in (*SI*), scan-out (*SO*) and scan enable (*SE*). There are three different modes to insert and start the test in scan-based design: *launch-off-shift* (LOS), *launch-off-capture* (LOC) and *enhanced scan* [TPC11].

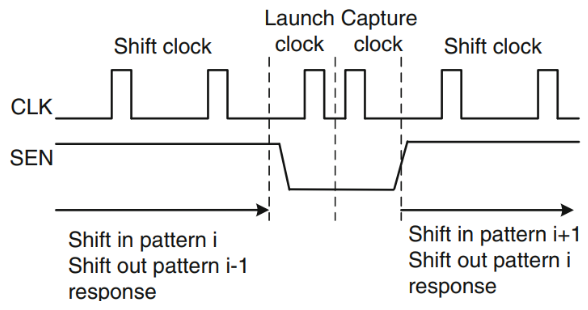
There are divergences in the literature about the nomenclature of these terms. In some books, like [WST10], launch-off-shift, launch-on-shift and skewed load are used for the same purpose. In the same book, authors define launch-off-capture, launch-on-capture, functional justification, double-capture and broad-side test as identical tests methods.

Launch-off-shift and *launch-off-capture* share the same scannable flip-flop (FF) architecture. They have *Muxed-D FFs*, as Figure 2.15 shows, which basically is a type-D FF with one multiplexer selecting either data in (*DI*) or *SI* by *SE* signal. If *SE*='1', the *Muxed-DFF* output is connected directly to next *Muxed-D FF*, making a serial configuration and working as a shift register (shift mode). This allows serially transmit all data from *SI* through all registers using *clock* and *SE* signals. If *SE*='0', the CL is reconnected to *Muxed-D FF* input as in a classical circuit. The difference between LOS and LOC is how the pattern is launched.

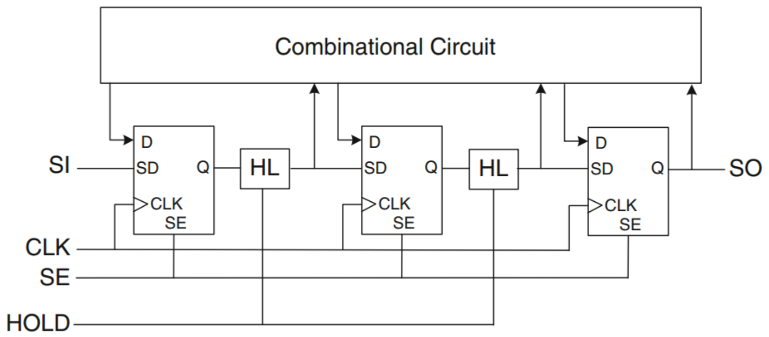
In *launch-off-shift*, as Figure 2.16(a) shows, the transition at the gate output is launched in the last shift cycle during shift operation (*SE*='1'). Then, *SE* goes low to enable response capture at the capture clock edge. So, LOS requires the *SE* signal to be timing critical (low skew) to guarantee all registers to identify capture cycle [TPC11].



(a)



(b)



(c)

Figure 2.16 – Scan-based tests [TPC11]: (a) launch-off-shift, (b) launch-off-capture and (c) enhanced scan.

Launch-off-capture, presented in Figure 2.16(b) can tolerate *SE* skew, because the launch cycle is separated from the shift operation. In this case, when scan in ends (shift mode), V_1 is applied and the CUT is set to an initialized state (reset/set) and V_2 depends on the functional response of V_1. This peculiarity restricts pair V_1, V_2 caused by less controllability of V_2, reducing coverage when compared to LOS.

The *enhanced scan* technique allows the independent combination of V_1 and V_2 patterns. This is made possible by inserting a hold latch (*HL*) between each scan flip-flop as Figure 2.16(c) shows. Both, V_1 and V_2 are loaded simultaneously alternating V_1 and V_2 during the shift operation. With this method, it is possible to generate tests considering the combinational logic alone. However, the addition of the hold latch reduces performance and increases area [TPC11].

The Small-Delay Defect (SDD) is one type of the timing defects that introduces a small amount of extra delay in the design [TPC11]. Considering SDDs is a recent demand, caused by paths with small size relative to the timing margin in recent technologies. The delay introduced by SDDs is small, but in high frequencies and shrinking technologies, this small contribution can become expressive, specially in small time slack designs. According to [TPC11], a large portion of failures in delay-defective parts are due to SDDs in recent technologies. To cover for SDDs in real situations, it is necessary to increase defect coverage and test quality or to decrease the number of test escapes.

One method to reduce test escapes is running the test in *Faster-Than-At-Speed* (FTAS) tests. FTAS tests consist in running the CUT in a frequency above the nominal defined for the design [XZVH09]. In this case, the same number of test patterns can detect more faulty circuits than a classical *At-Speed* (AS) approach, reducing the slack time. However, false-positives can lead to discarding good circuits, which would properly work under the nominal frequency. To increase circuit yield, some works like [XZVH09] calculate the precise up-frequency to use, in order to produce good SDD coverage and reduce false positives.

All the tests covered here focus on pattern generation and test environment in synchronous circuits. Only a few specific works can be found today covering asynchronous circuits test. Studies like [SM06] (Scan-chain), [RGP+15] (MrGO), [SO15] (delay test in BD CL), [MCFB16] (BIST in asynchronous channel), and [HSL17] (QDI Scan-chain), show alternatives to provide test features in asynchronous circuits, but this is yet an open field for asynchronous research.

Although many ICs are fully digital, the market trend to reduce costs with systems-on-chip that integrate multiple modules such as CPU, memory and peripherals, can also integrate analog circuits. However, unlike digital circuits, no analog fault model has been widely accepted as in digital testing [WST10]. Through boundary scan it is possible to insert and collect both digital and analog signals inside ICs. Different from digital ones where delay and logic levels are important, the major testing parameters in analog circuits include amplitude, slew rate, overshoot, settling time, bandwidth, phase noise and timing jitter.

Wang et al. [WST10] divide the test of analog circuits into two basic approaches: *functional testing* and *structural testing*. *Functional testing* is preferred, because of the continuous signal produced by such circuits, the wide range and variety of analog circuits, and their nonlinear characteristics [WST10]. However, as this kind of circuit usually has feedback to compensate process, voltage and temperature variations, highly accurate tests are necessary, requiring the use of expensive equipment. The *structural test* needs access to each analog module individually, to test its internal nodes. However, analog circuitry usually comprises a series of modules which cannot be treated separately, making this test impractical in many cases.

The expected outputs from analog tests are usually based on previously simulated results. Monte Carlo analysis is typically used to generate different component values for fault-free operation with normal (Gaussian) or uniform distributions [WST10]. Standard deviation (σ) is the most important parameter to define a normal distribution. Usually, it is assumed that the components will vary up to $\pm 3\sigma$. The circuit fails if its parameters are measured out of this variation range.

Test architectures include several standards such as IEEE 1149.4, which can provide mixed-signal BIST structures. The oscillation BIST sets the analog circuit under test to form an oscillating circuit and, through this oscillation, determines a faulty or fault-free classification for the circuit.

As an alternative for analog and mixed-signals tests, the BIST technique uses a test controller, a Test Pattern Generator (TPG) and an Output Response Analyzer (ORA). The test controller commands the TPG to produce input signals for digital to analog converters. It also controls the interconnection between components and the TPG doing analog loops between modules. The ORA receives digital signals provided by analog to digital converters, compares the result with the expected ones and indicates the test result.

The BIST approach for structural mixed-signal test uses TPGs provided by Linear Feedback Shift Registers (LFSR) and ORA to compare the result with a test signature. Different from the previously discussed strategy, this ORA accumulates the sum of results to be compared when the test ends. The LFSR can provide many input configurations, including ramp, triangular, sinusoidal and square waves [WST10]. Also, the ORA can accumulate the sum of absolute values of the difference between the input test wave and the output response of the CUT.

Alternatives for frequency domain tests are also available. The Fast Fourier Transform (FFT) mixed-signal BIST and the Direct Digital Synthesis BIST can provide TPGs for many frequency dependent applications like radio frequency (RF). These BIST tests will not be discussed here because they are considered out of the scope of this Thesis. More details about analog circuit testing can be found e.g. in [WST10].

3. DELAY ELEMENTS OPERATING UNDER VOLTAGE SCALING

In current days, power efficiency is one of the main focus of Integrated Circuit (IC) design. Mobile devices, the Internet of Things (IoT) end nodes and smart watches are some examples of devices that can only be devised using ultra-low power circuits. In this way, several techniques such as Frequency Scaling (FS) [HHM+16] and Voltage Scaling (VS) [AYI17] have been applied to provide improvements in power consumption for synchronous circuits.

Asynchronous circuits, on the other hand, are known by their PVT robustness when using Quasi-Delay Insensitive (QDI) design templates and by the power efficiency when using Bundled-Data (BD) templates [SF13, BOF10, GMMC15], when compared to synchronous design approaches. The possibility of employing wide ranges of voltage scaling in QDI designs result in massive power reductions if operation falls to subthreshold supply voltages [CL10, HCGC15] if necessary. However, the lack of commercial Computer-Aided Design (CAD) tools to support QDI design is a major difficulty. Also, the large amounts of extra hardware required when encoding information for QDI templates [CVG07, BOF10] is not power efficient.

BD design, on the other hand, can take advantage of conventional CAD tools more easily. The design flow of Combinational Logic (CL) in BD templates is basically the same used in synchronous design. Controllers and Delay Elements (DEs) can be implemented using standard cells [GMMC15]. However, margins in DEs to compensate PVT variations considerably increases delays, which reduces circuit performance. Process variations could be compensated in post-silicon test steps. Temperature influences the delay too, but can be compensated by monitors and actuators, because of its large time constant curve. However, the voltage could change according to switching activity and this occurs faster than any controller is able to identify and take action [WH11]. Therefore, making the DE scale delays similar to the CL does reduce delay margins and the need for specific controllers.

This Chapter presents a voltage scaling analysis with focus on delay elements behavior. Section 3.1 shows the transistor sizing influence in delay variations. Also, the parameter Voltage Scaled Delay Ratio (VSDR) is proposed and defined as a way to precisely specify the VS impact on delay. Section 3.2 approaches different adjustable DEs commonly used in bundled-data 2-phase designs. They are implemented and simulated in three different technologies and simulated to show the impact of VS over them in Section 3.3. The analysis also shows which topology is less sensitive to VS.

3.1 Voltage Scaling Effects

The operation of circuits built with MOSFETs is sensitive to the supply voltage. A MOSFET is basically a voltage-controlled current source [Bak10]. Thus, any supply voltage variation affects the current provided by the transistor, influencing the time to charge and discharge the capacitances of devices it drives.

The time constant RC defines the propagation delay (t_{pd}) of digital gates. Basically, this constant is computed using the internal resistance (R) of the transistor and the load capacitance (C_L) connected to its output. The value of R can be obtained as the ratio of the voltage applied across the drain and source terminals (V_{ds}) to the current passing between these same terminals, I_{ds}. Equation 3.1 expresses the relationships about t_{pd} discussed here.

$$t_{pd} = R \cdot C = C_L \cdot \frac{V_{ds}}{I_{ds}} \tag{3.1}$$

The current passing through a transistor depends in turn on several parameters, which vary with the physical characteristics of the device, as expressed in Equation 3.2. The alpha current model is adopted in this analysis, because it produces results closer to real measurements for modern technologies, with significant short channel effects [WH11]. In this model, the term α serves to adjust the transistor current behavior. Here, the current produced by a transistor in the saturation region is seen as directly proportional to the capacitance and implant characteristics (represented in the Equation by k). It also depends on the transistor dimensions (directly proportional to the width W and inversely proportional to the length L), on the voltage applied across the gate and source terminals (V_{gs}), as well as on the threshold voltage (V_t).

$$I_{ds} = \frac{1}{2} \cdot k \frac{W}{L} \cdot (V_{gs} - V_t)^{\alpha} \tag{3.2}$$

Combining Equations 3.1 and 3.2 it is possible to obtain a new expression that shows how to compute the propagation delay as a function of the supply voltage (V_{DD}). To achieve this new expression for t_{pd}, consider a CMOS inverter gate and assume the voltage across drain and source (V_{ds}) and the one between gate and source (V_{gs}) as identical to the circuit supply voltage. In this way, the propagation delay under a given supply voltage is given by:

$$t_{pd} = C_L \cdot \frac{2 \cdot L \cdot V_{DD}}{k \cdot W \cdot (V_{DD} - V_t)^{\alpha}}. \tag{3.3}$$

From Equation 3.3, it is possible to relate the delays produced at different supply voltage levels (t_{pd_eval}) with the delay obtained at the nominal supply voltage (t_{pd_nom}). This gives rise to the notion of Voltage Scaled Delay Ratio (*VSDR*) that numerically represents the delay difference between two given supply voltage levels, according to Equation:

$$VSDR = \frac{t_{pd\_eval}}{t_{pd\_nom}}. \tag{3.4}$$

Replacing the terms of Equation 3.3 in Equation 3.4, it is possible to obtain an intermediate equation for the *VSDR* term, i.e.:

$$VSDR = \frac{C_L \cdot \dfrac{2 \cdot L \cdot V_{eval}}{k_{eval} \cdot W \cdot (V_{eval} - V_{t\_eval})^{\alpha_{eval}}}}{C_L \dfrac{2 \cdot L \cdot V_{nom}}{k_{nom} \cdot W \cdot (V_{nom} - V_{t\_nom})^{\alpha_{nom}}}} \tag{3.5}$$

In this new Equation, C_L is obviously identical for both voltages, the same being true for L and W. The constants k_{eval}, k_{nom}, V_{eval} and V_{nom} on the other hand, display a linear dependence with the voltage variation and can be substituted by the single term K. Parameter V_t, however, shows second order effects, depending on W, L, V_{ds} and V_{bs} (bulk/body to source voltage) as demonstrated in the literature. See for example references [TM11, YLM+10]. According to [WH11], since α is an approximation factor, it decreases with the supply voltage. The simplifications result in Equation 3.6 that summarizes a way of computing the *VSDR*. It is important to notice that, according to this Equation, the threshold voltage is of fundamental relevance to define the behavior of the *VSDR* metric.

$$VSDR = K \cdot \frac{(V_{nom} - V_{t\_nom})^{\alpha_{nom}}}{(V_{eval} - V_{t\_eval})^{\alpha_{eval}}} \tag{3.6}$$

Traditionally, V_t increases as L increases, until it asymptotically stabilizes as Figure 3.1(a) displays. This behavior is known as short-channel effect and is caused by source and drain implantations producing drain-induced barrier lowering or DIBL. In this way, it is expected that *VSDR* increases with L. However, in recent technologies, the reverse short-channel effect causes the exactly opposite effect. New technologies employ anti-punch-through implants to produce adequate device isolation [Col04]. According to [TM11], this produces a charge-sharing effect in the transistor channel, which is still apparent at very small values of L. As a result, the effective threshold voltage decreases as L increases as displayed in Figure 3.1(b). The dashed line in this figure displays a series of transistor behaviors in cases where the short-channel effect is more relevant for very small L but where just a slight increase in L reverses this trend and keeps reducing the threshold voltage for larger values of L. Accordingly, in new technologies it is expected that *VSDR* decrease as L increases.

The narrow-channel effect is responsible by the impact of W over V_t. Old technologies (above $0.35\mu m$) and high voltage transistors are commonly implemented using LOCal-Oxidation of Silicon (LOCOS) isolation processes, where W influences inversely the

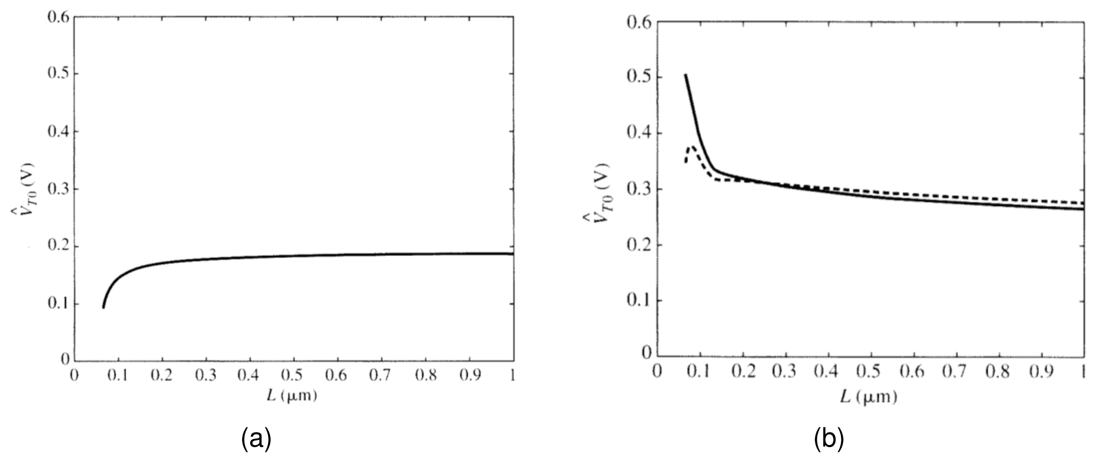


Figure 3.1 – Variation of the effective threshold voltage ($\hat{V}_{T0}$) (author notation for V_t), according to the transistor length (L) [TM11]: (a) without short-channel effects acting; (b) with short-channel effects considered.

V_t. The gradual transition from thick (field) oxide to thin (gate) oxide changes the V_t when W is near to its minimum size [TM11]. In this case, V_t reaches its maximum value, reducing as W increases until the contribution of the latter is nearly nulled. Thus, in LOCOS *VSDR* decreases as W increases. Figure 3.2(a) shows the effective threshold voltage varying with changes in W for the LOCOS isolation case. However, according to [TM11], in the majority of sub 0.35μm processes, another isolation is implemented: Shallow-Trench Isolation (STI) [TM11]. Instead of a gradual transition from field to gate oxide, in STI the change is abrupt altering the W contribution to V_t. In this case, something as a *"reverse"* narrow-channel effect causes increments in V_t when W increases, the opposite effect observed in the LOCOS isolation case. However, as in LOCOS, the W contribution to V_t still decreases when the width increases. Figure 3.2(b) shows the effective threshold voltage increasing as W increases. As a result, it is assumed that in STI process *VSDR* grows when W gets bigger, until it asymptotically stabilizes.

3.1.1 The VSDR Concept Validation

This Section targets the validation of the *VSDR* concept as defined above. To evaluate the dependency with the threshold voltage as predicted by Equation 3.6, a set of simulations were conducted to capture the *VSDR* behavior. These simulations have as objective to obtain results that are close to the values measured in real circuits. To cover a wide spectrum of results, simulations employ three distinct commercial CMOS technologies to which the author has had access. Of these, two are conventional technologies based on bulk CMOS (a 180nm tech from Taiwan Semiconductor Manufacturing Company (TSMC) and a 65nm tech from STMicroelectronics (STM)), while the last one is a modern, Fully

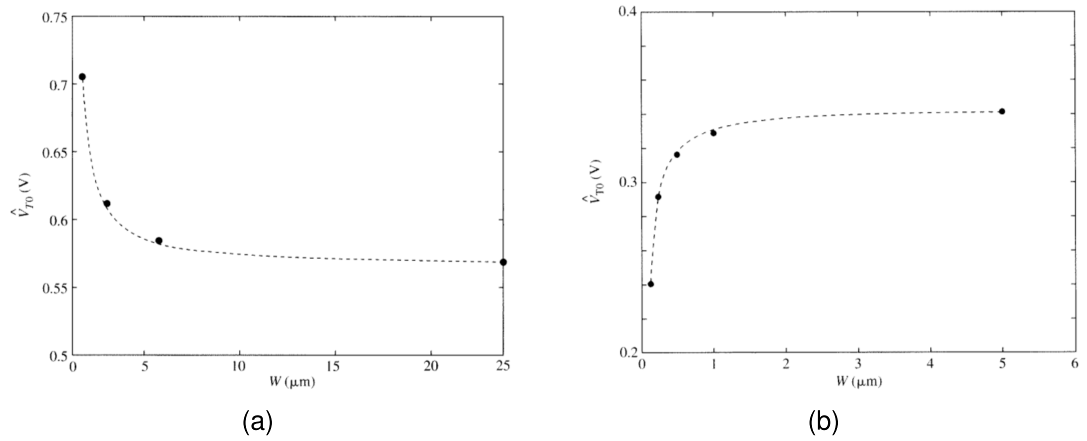


Figure 3.2 – Illustrating the variation of the effective threshold voltage ($\hat{V}_{T0}$ is the notation for V_t employed in reference [TM11]) with transistor width (W): (a) using the LOCOS isolation process; (b) using the STI isolation process.

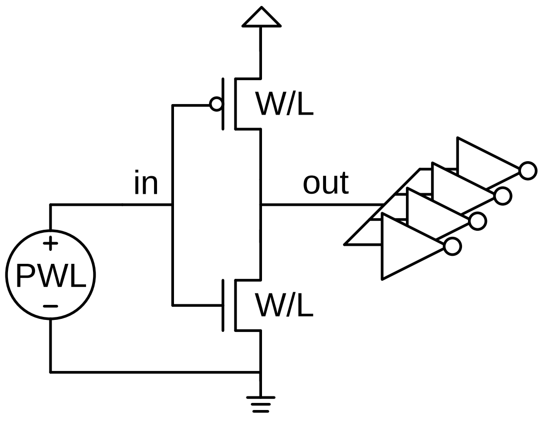
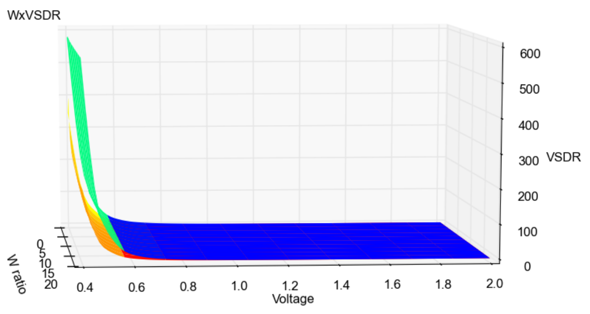


Figure 3.3 – Circuit schematic used to evaluate *VSDR* according to variations in transistors width (W) and length (L) under different supply voltages.

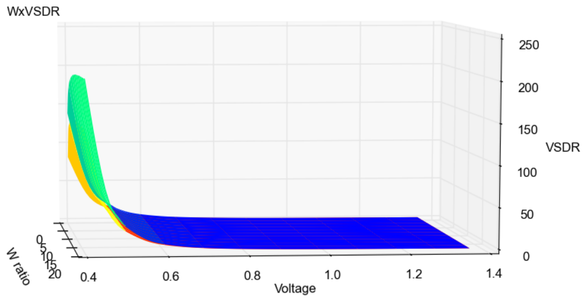
Depleted Silicon-On-Insulator (FDSOI) (a 28nm tech from STMicroelectronics (STM)). The employed simulator is the commercial tool SPECTRE from Cadence, and the technology parameters are those available in the respective Process Design Kits (PDKs) furnished by the foundries.

Since the threshold voltage of a transistor is influenced by transistor sizing, the width and length parameters vary to determine their impact on the *VSDR*. The gate employed to evaluate the *VSDR* behavior is the simplest possible, i.e. an inverter. The simulation environment consists in a variable supply voltage, an input source with a PieceWise Linear (PWL) function and a passive load based on a Fanout Of Four (FO4) of the inverter under evaluation. Ramp and FO4 parameters are defined in a compatible way for each technology. This circuit schematic is shown in Figure 3.3.

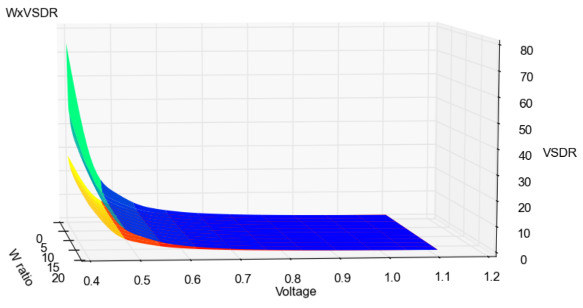
The graphs in Figure 3.4 refer to the *VSDR* behavior at distinct voltage levels when varying the width (W) of the inverter transistors. It is important to highlight that the *VSDR* proposed formulation only remains valid while transistors are in saturation. Thus, it is im-



(a)



(b)



(c)

Figure 3.4 – The *VSDR* variation due to changes in W and in the supply voltage: (a) for the TSMC 180nm bulk CMOS technology; (b) for the STmicroelectronics 65nm bulk CMOS technology; and (c) for the STmicroelectronics 28nm FDSOI technology.

portant to define a voltage below which results do not make sense anymore. This voltage is within the near-threshold region but is above the threshold voltage itself. Only after defining this voltage it is possible to evaluate the impact of varying W.

Since the *VSDR* intrinsically employs the transistor-furnished current to define the delay, it is possible to use its behavior to determine the limit voltage mentioned in the previous paragraph. This voltage corresponds to the point where the *VSDR* curve suffers an inflection towards rapidly increasing values. In Figure 3.4(a) it is possible to identify that for the 180nm technology the inflection takes place below 0.6V. For the 65nm technology, this occurs at around 0.55V (see Figure 3.4(b)), while for the 28nm technology the inflection point is located around 0.5V (see Figure 3.4(c)). The word *ratio* (or equivalently *scale factor*) is a reference to the result of dividing some cell dimension (W or L) by the minimum allowed size for the respective dimension in each technology.

To correctly evaluate the *VSDR* formulation, the presented results are for the lowest possible valid voltage, within the near-threshold region. In this region, it is expected that results highlight the impact of transistor dimensions on the *VSDR*. To execute the simulations, the least dimensions (W e L) of transistors forming an inverter are employed in each technology. The scale factor multiplies the respective dimensions. The load, however, is kept constant, to avoid interference in the results. All simulations present results for the typical corner of each technology, and use the respective nominal supply voltages. Again the employed simulator is the Cadence SPECTRE tool. Figure 3.5 summarizes the results.

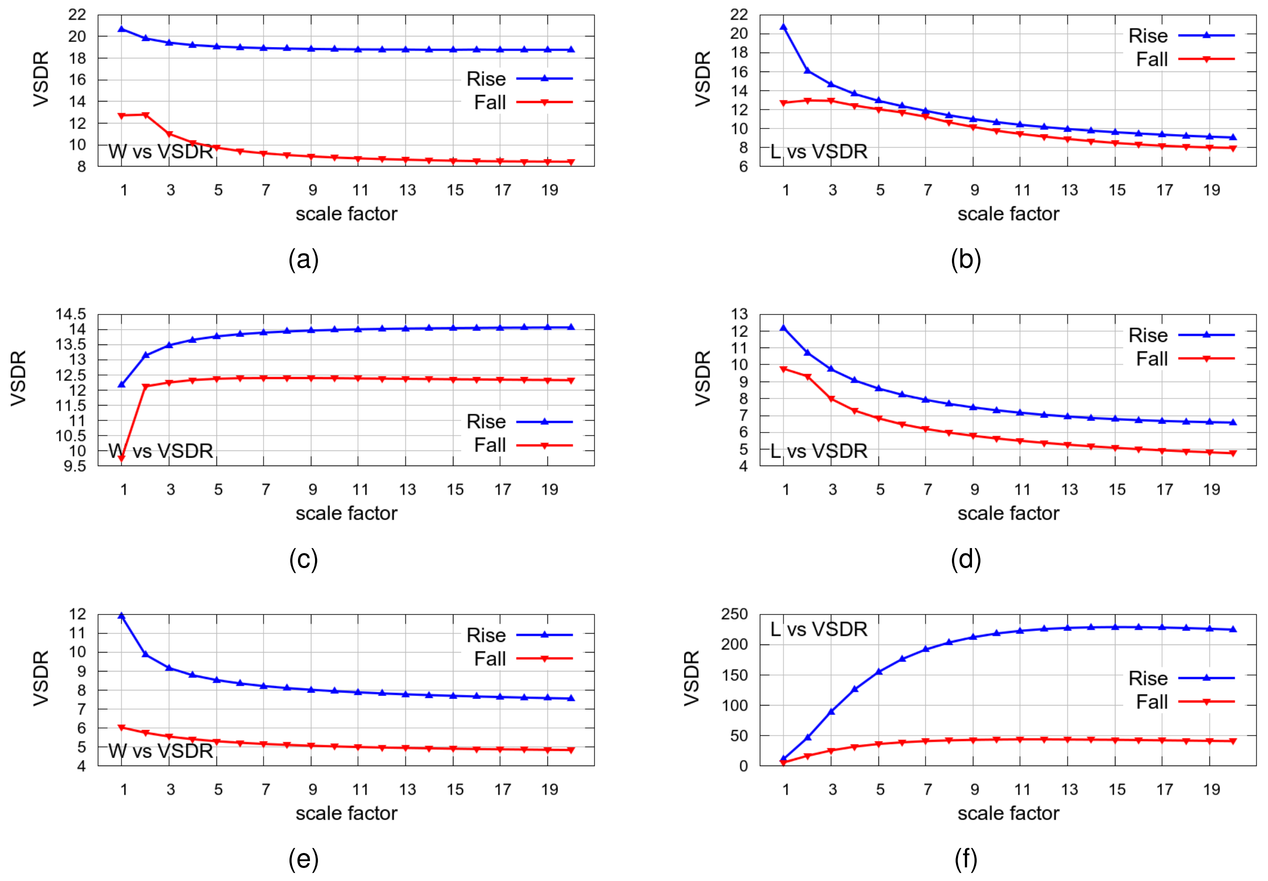


Figure 3.5 – The *VSDR* variation as a function of: (a) Width and (b) Length in 180nm, (c) Width and (d) Length in 65nm, and (e) Width and (f) Length in 28nm.

The graphs of Figure 3.5 reveal that transistor dimensions significantly affect the *VSDR* behavior. Changes in *W* do not seem to influence *VSDR* as much as changes in *L*. Nonetheless, *VSDR* variations as a function of *W* are seven times more significant at minimum dimensions. For the 65nm technology (Figure 3.5(c)), *VSDR* does not change more than 6% when starting with *W* twice the minimum. As for *L* variations, *VSDR* changes are more pronounced for factors below 9 times the minimum dimension. Yet, changes are significant in larger dimensions (see the 180nm technology results in Figure 3.5(b)).

Given the structure of Equation 3.6 for the *VSDR*, and assuming that the *VSDR* values follow threshold voltage variations, it is expected that similar shapes arise for the simulated *VSDR* graphs and those graphs describing the variations of V_t for transistors. Figure 3.6 presents the corresponding variations for the threshold voltage, extracted for each scale factor employed in the *VSDR* simulations. It is relevant to point out that scalar values were omitted here, to avoid violating the Non-Disclosure Agreement (NDA) signed with each of the foundries that provide the commercial PDKs used in the experiments. To capture the threshold voltage values for each transistor according to its dimensions a special feature is available. The feature consists in the "print DC Operating Points" resource, available in the Analog Design Environment (ADE) of Cadence and in the SPECTRE simulator. It is

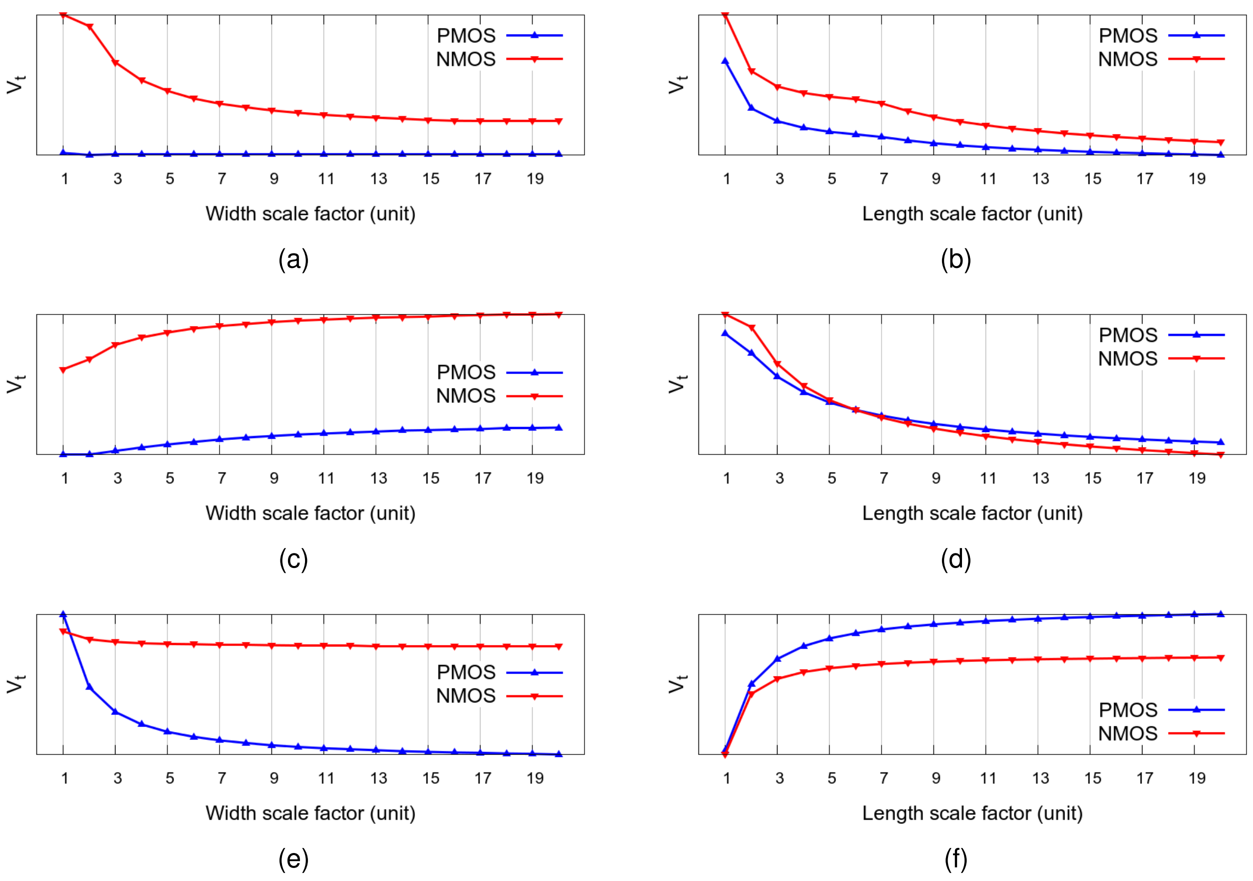


Figure 3.6 – V_t variation according to: (a) Width and (b) Length in 180nm; (c) Width and (d) Length in 65nm; and (e) Width and (f) Length in 28nm.

important to note that the scale used in the graphs hides variations in threshold voltage of 180nm PMOS transistor in width variation.

Correlating graphs of *VSDR* and V_t variations, it can be observed the relevance of V_t in the *VSDR* shape. Still, it is also possible to observe the impact of α in some cases (specially in the NMOS transistor *L* in the 180nm technology), where the nominal supply voltage is more than three times the minimum voltage in saturation. Scalar factors influence variations, but the graph shape correlation seems clear.

The only case where a considerable deviation appears is in the *L* variation for NMOS transistors in the 180nm technology. In this technology varying *L* produces a trend for the fall *VSDR* to grow until the scale factor of 3, contradicting the V_t variation under the same circumstances. This is due to two factors. *L* directly impacts α in the transistor current model. This causes α to grow until the scale factor of 2 as *L* increases. Concomitantly, the 180nm technology is the one that presents highest voltage gap between the nominal supply voltage and the transistor threshold voltage (1.2V). This increases the impact of α. This same effect occurs in the *VSDR* rise curve in this technology, when a voltage slightly above the minimum is adopted. Figure 3.7 presents the behavior of *VSDR* when varying *L*

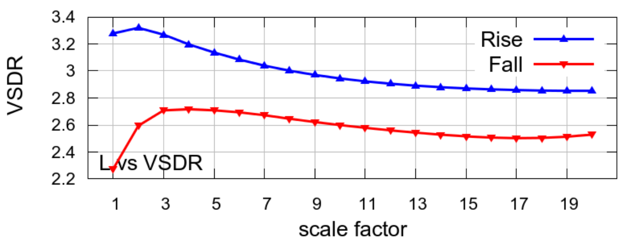


Figure 3.7 – *VSDR* variation in the 180nm technology varying length (*L*) for a 0.9V supply.

under 0.9V instead of the previous 0.6V values. The experiment clarifies that both cases are indeed affected by the value of α.

Thus, it is noticeable the correctness of proposed *VSDR* formulation, where V_t and α are of paramount relevance. Nonetheless, one very salient feature in the *VSDR* graphs of Figure 3.5, for all technologies, is the discrepancy between the rise and fall *VSDR* values. This difference can attain numbers that amount to up to 19.8% in the 65nm technology until up to 81,7% in the 28nm technology. For synchronous circuits, this difference has to be aggregated in the form of margins. For 2-phase bundled-data asynchronous circuits, the difference impacts not only the Combinational Logic, but also the delay element architecture, responsible for guaranteeing correct circuit operation.

3.1.2 Rise and Fall *VSDR* Difference Reduction

Differences between rise and fall times are commonplace in CL blocks. However, delay elements for 2-phase BD templates require that these times to be identical or very close to each other, otherwise delay margins must be applied. Increasing margins directly imply less performance for this kind of circuits. One technique to reduce the impact of such differences between the rise and fall *VSDR* values is to mandate that the signal propagates through complementary transistor networks with similar characteristics.

For example, the use of two identical inverters in sequence is able to balance the rise and fall *VSDR* values. Refer to Figure 3.8 that illustrates this effect. The effect is in fact due to the compensation caused by the signal passing in every situation through two complementary transistors: an NMOS and a PMOS. In the Figure, if the input rises ('0' to '1', following the dashed, blue line) the signal follows through the NMOS transistor of the first inverter, and through the PMOS inverter of the second one. The opposite situation ('1' to '0', following the continuous, red line), mandates that the signal crosses exactly the same number and type of transistors, just in another order (PMOS followed by NMOS). If the two paths are adequately sized, identical or very close delays can easily be obtained for both rise and fall times.

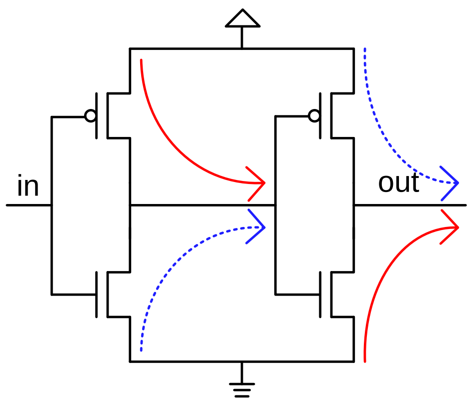


Figure 3.8 – Circuit topology to reduce *VSDR* rise/fall difference.

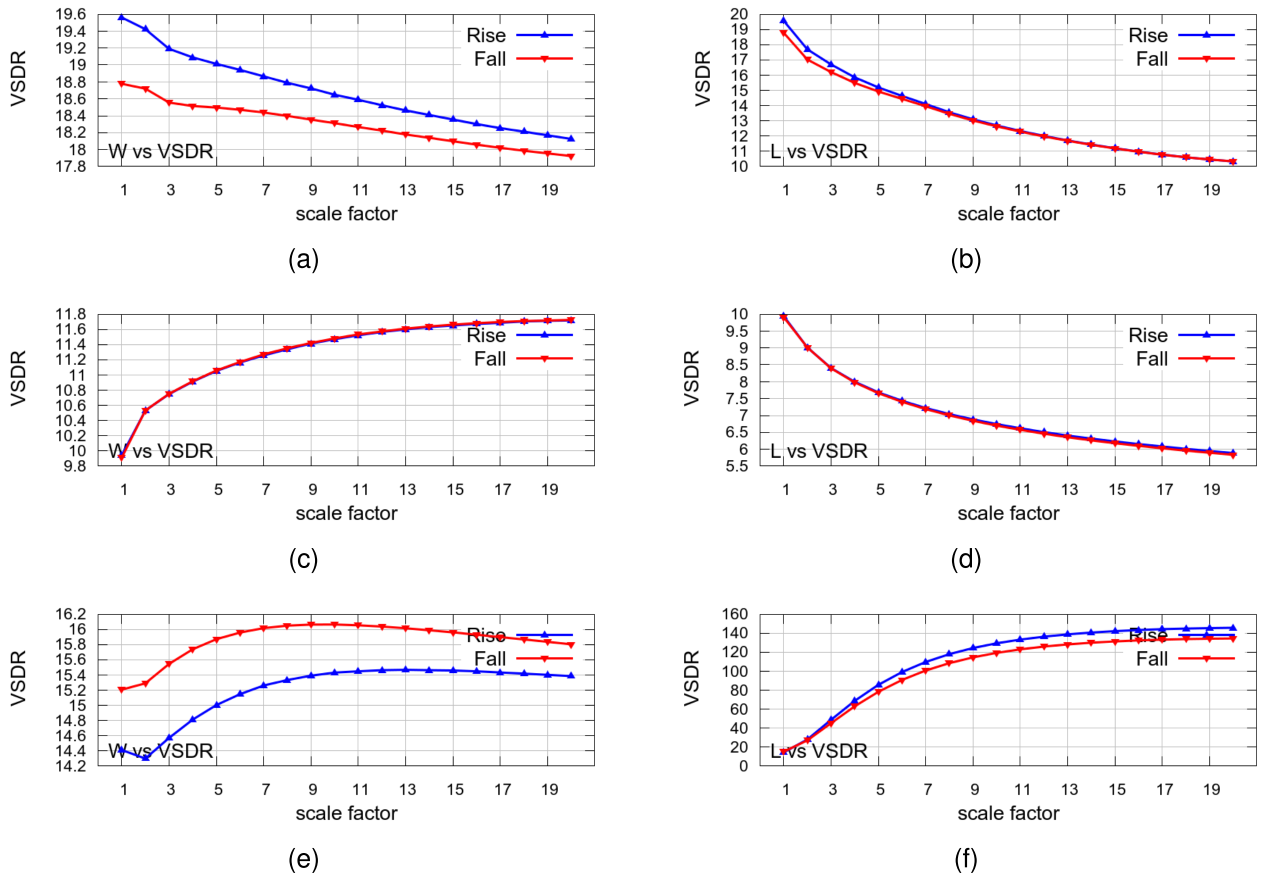


Figure 3.9 – The *VSDR* variations after applying the rise/fall difference reduction technique: (a) Width and (b) Length for the 180nm technology, (c) Width and (d) Length for the 65nm technology, and (e) Width and (f) Length for the 28nm technology.

Applying this simple technique one obtains the *VSDR* results presented in Figure 3.9. When these area compared to the *VSDR* behavior depicted in Figure 3.5, it is clear that the rise and fall differences do reduce in all cases, and for all technologies. Numerically, differences vary between 0.4% in the 65nm technology and 7.7% for the 28nm technology. This is in fact a reduction of more than 90% when compared to the previous range of differences.

3.2 Delay Elements for 2-Phase Bundled-Data

Delay Elements (DEs) are often employed for the design of clock generation and distribution. Examples of the former include Delay-Locked Loops (DLLs), Phase-Locked Loops (PLLs) and Digitally Controlled Oscillators (DCOs) [SLK+00, KY09, MCA12, HHM+16]. For the latter, different DEs exist to fix skew problems post-silicon [CDS+08]. DEs are also needed in Bundled-Data (BD) asynchronous design templates [BOF10, MGT00, VTN12]. Such templates employ explicit request and acknowledge control signals for sending data between blocks [GBN13] and internally rely on DEs whose overall delay must be matched to the Combinational Logic (CL) of each stage.

Delay elements must be conservatively designed to be longer than the CL but this delay difference should be minimized, as it represents an overhead that reduces performance. Unfortunately, increasing process variations in deep submicron technologies forces the addition of margins to delay lines designed in traditional ways. For this reason, *programmable DEs* that can be tuned post-silicon to match the actual CL delays in silicon provide a far more attractive solution. Moreover, for BD templates based on 2-phase control schemes, DEs need to have balanced rise and fall propagation delays. This occurs because both delays must be equally matched to the CL, since any difference between them represents additional overhead.

A challenge to modern IC designers is the tight power budgets that extend from mobile applications concerned with battery lifetime, to high-end server processors concerned with heat dissipation [HB13]. A trend to cope with such a challenge is the dynamic scaling of the supply voltage as discussed in [HB13] and [CPR10]. While this approach can be difficult to develop for synchronous designs, asynchronous circuits are known to efficiently support voltage scaling (VS) techniques as discussed in [CPR10]. Although earlier works discussed the application of asynchronous techniques to VS applications [HB13, CPR10], they are focused on application-specific examples and, as far as it was possible to verify, none of them addresses the effects of VS in DEs. Yet, this is a very important concern because the DEs must remain matched to the CL as voltage scales without unnecessarily increasing margins. In other words, VS affects the CL delay and its associated DEs must be able to conservatively, yet closely, track such variations. The objective must be to ensure correct operation while avoiding performance losses.

DEs are often used in analog circuit design. There, these are usually based on reference current sources or bias voltages. Some such DE architectures put a focus on minimizing temperature or voltage dependency or low power dissipation. These characteristics lead to circuits that frequently are sensitive to digital circuit noise. For this reason the analog circuits are physically placed away and isolated from the digital part of the circuit, to avoid interference. While DEs for analog application are usually programmable, their target is mostly in producing a specific, predefined delay value, which is kept unchanged even under process, voltage and temperature variations. This is not aligned with the objective of DEs studied and proposed in this work, which looks for DEs able to track the delay of some associated digital circuit CL and adapt its delay to this value.

Studies presented in [CLM+16] show that the voltage variation among a cluster of cells close to each other does not exceed 29mV in a 65nm CMOS technology. Indeed, the closer a DE is from its associated CL, the more variations in the CL will be replicated in the DE. Associated to the characteristics expressed above for analog DEs, this leads to the choice of using here DEs that are exclusively based on digital operation. Accordingly, this work proceeded to an investigation of three digital programmable DE architectures commonly used in 2-phase BD designs and their behavior when submitted to VS effects: *i*) one based on multiplexers and inverters and buffers (MUX-DE in Figure 3.10(a)) [TBW+09]; *ii*) one based on directly-controlled current-starved inverters (DCCS-DE in Figure 3.10(b)) [MNS03]; and *iii*) one based on current mirror-controlled current-starved inverters (CMCS-DE in Figure 3.10(c)) [MNS05]. The DE *i*), referred to as the MUX-DE, was chosen due to the fact that it is one of the most popular DEs, given its relatively simple implementation and the fact it can be built using standard cells. Figure 3.10(a) shows the schematic of a 4-bit MUX-DE. It consists of four inverter sequences with an even number of gates or four buffer sequences, where each sequence has twice the *Delay* of its successor sequence, and four MUXes that select the path used to delay the input signal *in*. A selected codeword configures the MUXes, choosing the overall number of cascaded gates in the delay path, hence defining the total delay.

DEs *ii*) and *iii*) are intuitively more power- and area-efficient than the MUX-DE. However, their complexity makes balancing their rise/fall delays challenging, especially under VS. DE *ii*) is a Directly-Controlled Current-Starved Delay Element (DCCS-DE) illustrated in Figure 3.10(b) [MNS03]. It uses a parallel set of transistors as current sources to digitally control the propagation delay. Lastly, DE *iii*) is a Current Mirror Current-Starved Delay Element (CMCS-DE). Its schematic depicted in Figure 3.10(c), and comprises a Current-Starved Inverter (CSI) controlled by a configurable current mirror, similar to the one presented in [MNS05].

DEs *ii*) and *iii*) are programmable via current sources that control the amount of charge that flows through the CSIs. The current is determined by a binary codeword that selects the transistors which source the current. These transistors are carefully sized to provide a defined range of delays. The difference between the two CSI designs is that one uses a direct pull-up and pull-down current source, while the other uses a current mirror to control the current. A previous work [KY09] proposed optimizations for CSI DEs. However, these optimizations do not focus on 2-phase operation, making the new architecture proposed by these authors inadequate to the objective of this work.

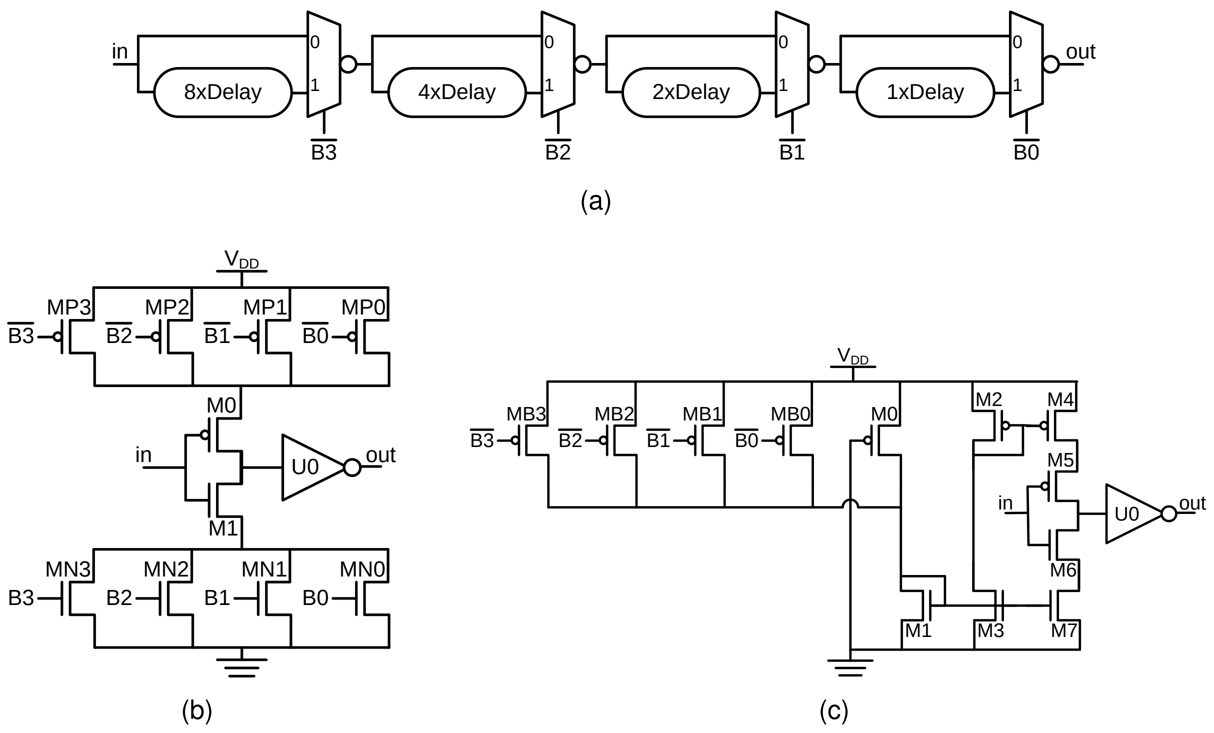


Figure 3.10 – The basic schematic of the three DEs investigated: (a) the MUX-DE [TBW+09]; (b) the DCCS-DE [MNS03]; and (c) the CMCS-DE [MNS05].

The next Section assesses how VS affects delay margins on the three DE architectures defined above. Also, Section 3.3 presents a method to optimize the design of these DEs to reduce performance losses as voltage scales. The method includes the proposal of an approach to balance rise and fall propagation delays. One original contribution of this work is to provide guidelines for the design of DEs when operation under VS is an important issue, moving forward the state of the art in low power asynchronous design.

3.3 The Impact of Voltage Scaling in Delay Elements

VSDR numerically defines the VS impact on the delay between two specific distinct voltage levels. To simplify the analysis, the *VSDR* is calculated between some near-threshold and the nominal voltage. However, the total delay of a programmable DE depends not only on a current source or a single load capacitance. It in fact depends on all its current sources and on all its internal capacitances, as well as on its programming value.

A DE can be modeled in a simple way by an RC constant. This constant can in fact be approximated by multiple RC constants, one for each submodule of a programmable DE. The mathematical ratio defining the total *VSDR* ($VSDR_{tot}$) of a programmable DE is:

$$VSDR_{tot} = \frac{\sum_{i=1}^{n} t_{pd\_near_i}}{\sum_{i=1}^{n} t_{pd\_nom_i}}, \tag{3.7}$$

where i is the submodule index and n is the total amount of submodules in use of the programmable DE.

Considering that the near-threshold propagation delay of a DE submodule can be obtained from Equation 3.4:

$$t_{pd\_near} = VSDR \cdot t_{pd\_nom} \tag{3.8}$$

and substituting the latter in Equation 3.7, a new equation results:

$$VSDR_{tot} = \frac{\sum_{i=1}^{n} VSDR_i \cdot t_{pd\_nom_i}}{\sum_{i=1}^{n} t_{pd\_nom_i}} = \frac{\sum_{i=1}^{n} VSDR_i \cdot t_{pd\_nom_i}}{t_{pd\_nom_{tot}}} \tag{3.9}$$

That represents the total *VSDR* as a weighted average of the *VSDRs* of each submodule.

This preliminary analysis points out that each reprogramming of a DE can result in significant changes of its total *VSDR*, which may in turn lead to a DE with a delay distinct from the expected in a near-threshold voltage. This delay variation is undesirable, because it may require additional margins in the DE design.

To evaluate the *VSDR* impact on the reprogramming of the DE types analyzed earlier in this Chapter, all were implemented in multiple versions on the three target technologies. The target was also to quantify the impact of the *VSDR* in these DEs in different technologies. The three DEs were accordingly designed and synthesized in Taiwan Semiconductor Manufacturing Company (TSMC) 180nm bulk CMOS, STMicroelectronics (STM) 65nm bulk CMOS and in STMicroelectronics (STM) 28nm FD-SOI.

To allow obtaining value that can be easily correlated, the base delay for all designs was defined as the delay provided by a 4-bit (4-stage) MUX-DE producing 16 different delay values. From this specification, versions of the DCCS-DE and of the CMCS-DE were elaborated to produce an identical or similar set of delay values, at least for the maximum delay obtained. Note that due to its structure, the DCCS-DE do not present a "0", since this implies interrupting the connection of the circuit to its supply voltage.

First, Section 3.3.1 presents the *VSDR* behavior for DEs designed to produce identical delays for rise and fall times in their *most nominal corners*. Later, Section 3.3.2 proposes an optimization of the designs to reduce the *VSDR* difference, and discusses a number of experiments that corroborate the benefits of the proposed technique. This Section also analyzes another DE specifically designed to reduce delay margins caused by *VSDR* variations.

3.3.1 Voltage Scaling Effects on Non-Optimized Delay Elements

As already discussed during previous analyses of *VSDR* graphs, each employed technology displays unique characteristics and effects under voltage scaling. Thus, the impact caused in distinct technologies (or classes of technologies) must be separately studied

Table 3.1 – Maximum difference between rise and fall delays in DEs implemented in 180nm.

	@ Nominal	@ Near Threshold
MUX-DE	1.7%	3.2%
DCCS-DE	1.4%	44.7%
CMCS-DE	5.1%	19.3%

and analyzed, targeting the proposal of architectural solutions with applicability to the widest possible range of technology cases. The implemented DEs adopt the respective architectures displayed in Figure 3.10. Their component devices (transistors) are dimensioned to minimize the difference between the rise and fall times of DEs under nominal operation. The maximum delay associated with each technology is determined by the MUX-DE delay, which is implemented using buffers and/or inverters of the respective standard cell libraries. Just as defined for the *VSDR* extraction experiments previously described, the simulation environment consists of a variable supply source, a PieceWise Linear (PWL) input source, a control bus for varying the DE programming and a passive load based on a Fanout Of Four (FO4) of the inverter or buffer employed in the MUX-DE. Ramp and FO4 parameters have compatible values defined for each technology. Note that both, DCCS-DE and CMCS-DE architectures have inverters in their respective outputs. Thus, the rise and fall effects observed in the evaluation of an inverter in each technology correspond to the fall and rise times, respectively.

The graphs displayed in Figure 3.11 reflect the behavior of the MUX-DE, DCCS-DE and CMCS-DE implemented in 180nm as a function of the employed programming codeword to produce the entire range of (discrete) delay values. The left column of graphs shows results for the nominal corner of operation, while the right column of graphs shows results under the defined near-threshold voltage. It is clear the considerable mismatch between the rise and fall times in the behavior of the DCCS-DE and the CMCS-DE in the near-threshold voltage. It is even possible to note the loss of the monotonic behavior for the CMCS-DE between codewords 7 and 8, where an increment in delay takes place where a decrement would be expected.

Table 3.1 summarizes the maximum delay difference for each DE designed in 180nm. Although the design for nominal conditions incurs in a rise to fall difference of up to 5.1% (for the CMCS-DE), the impact of voltage scaling can cause an increase of almost 32$\times$ in the rise to fall difference (for the DCCS-DE) in this technology. In the MUX-DE, where the difference is the smallest, the difference almost doubles.

When the analysis takes the point of view of *VSDR* values, the impact of reconfiguring the DE under VS is even more impressive. Figure 3.12 presents the *VSDR* variation for each DE in 180nm. The normalized *VSDR* axis refers to the fraction above the lowest *VSDR* value obtained in each simulation. This normalization helps comparing results among different DE architectures and technologies. Here, variations reach up to 128% (for the DCCS-DE). The smallest *VSDR* variation is for the CMCS-DE, and reaches up to 44.5%.

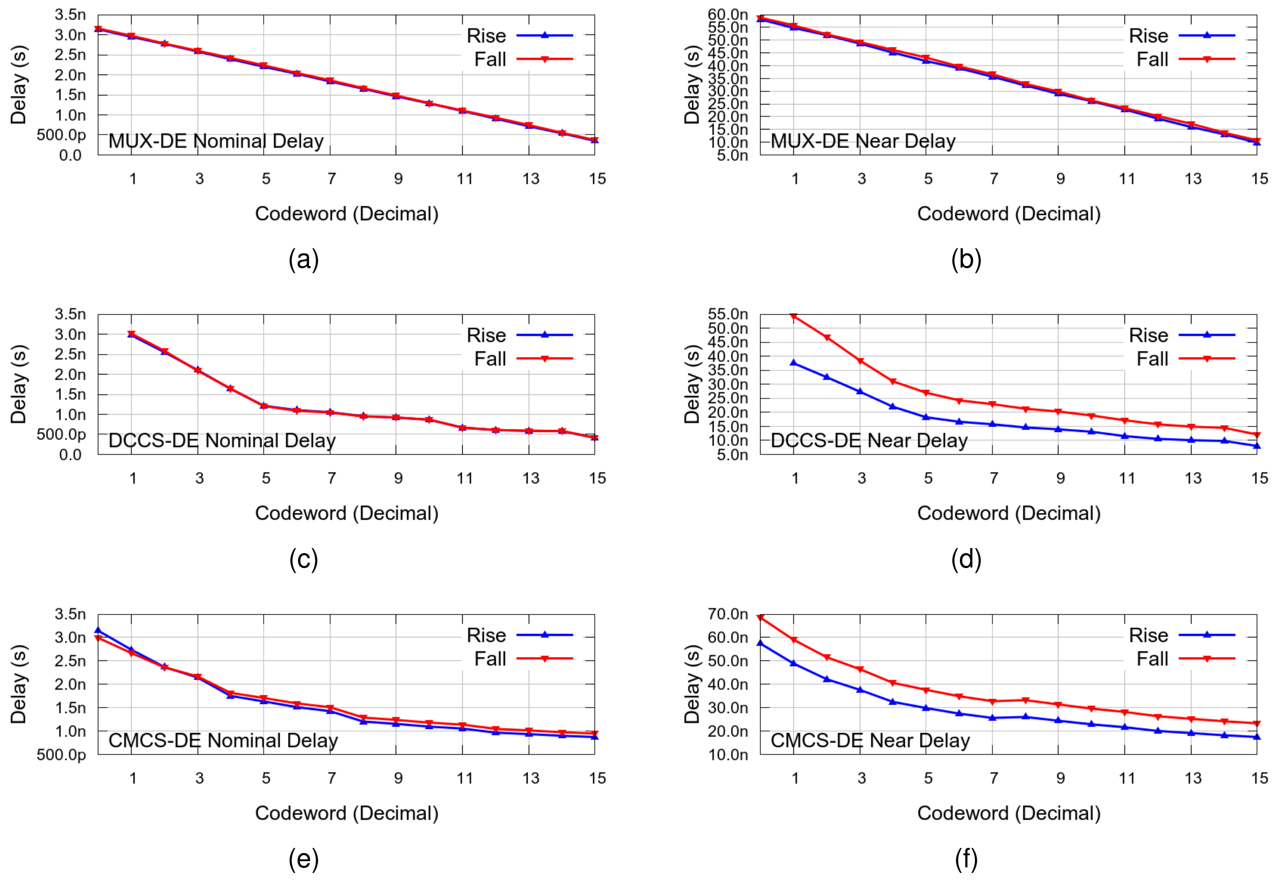


Figure 3.11 – Delay produced by programmable DEs implemented in 180nm: (a) MUX-DE @ nominal and (b) MUX-DE @ near, (c) DCCS-DE @ nominal and (d) DCCS-DE @ near, and (e) CMCS-DE @ nominal and (f) CMCS-DE @ near.

This indicates the need for compensating *VSDR* variation, in addition to compensating rise and fall delay mismatches.

Next, come the 65nm implementations. Figure 3.13 shows the generated delays as a function of the DE programming. The left column of graphs displays the delays produced by the DEs under nominal voltage. The right column graphs shows the near-threshold voltage delays. It is easy to note that VS impacts more intensely the difference between rise and fall delays for the DCCS-DE and CMCS-DE, when compared to the same DEs in the 180nm technology. Just as happened in the 180nm technology implementation, the CMCS-DE presents loss of monotonicity when operating in near-threshold.

Table 3.2 resume the maximum variations found in the data of the graphs from Figure 3.13. It is interesting to observe the difference between the rise and fall times for the MUX-DE. Under nominal voltage, the best implementation obtained in this range of delay requires a considerable delay margin for this DE (10.1%). Under near-threshold, however, there s a reduction in this variation (1.2%), due to the increase of the delay for the fall time above the increase of the delay for the rise time in this voltage. Besides, in the case of the

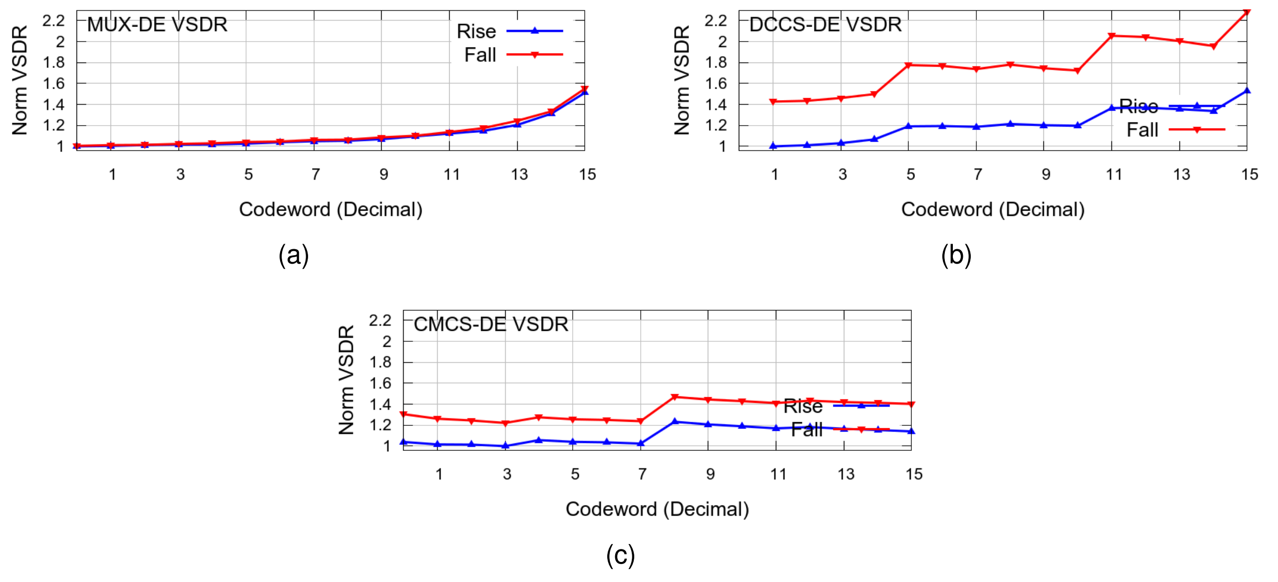


Figure 3.12 – Normalized *VSDR* values for DEs implemented in 180nm: (a) MUX-DE, (b) DCCS-DE and (c) CMCS-DE.

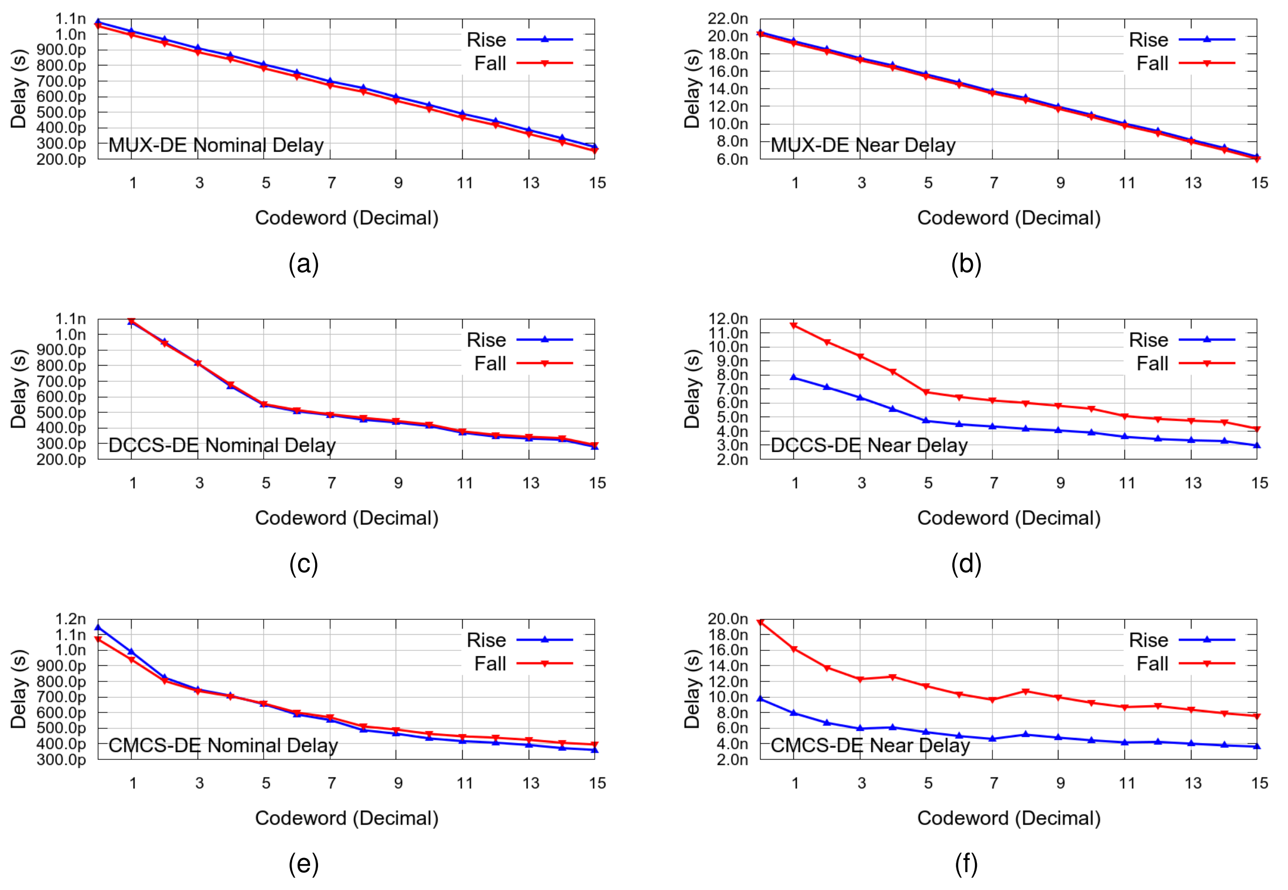
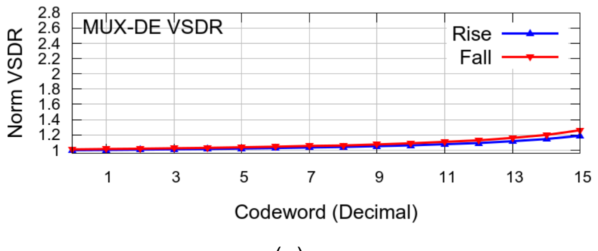


Figure 3.13 – Delay produced by the programmable DEs implemented in 65nm: (a) MUX-DE @ nominal and (b) MUX-DE @ near, (c) DCCS-DE @ nominal and (d) DCCS-DE @ near, and (e) CMCS-DE @ nominal and (f) CMCS-DE @ near.

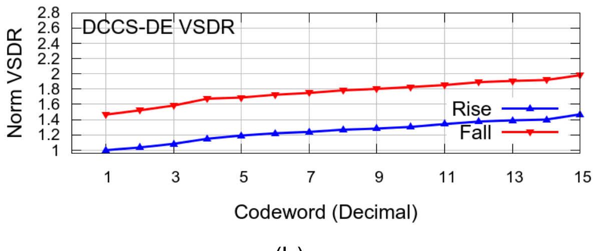
CMCS-DE, the difference between rise and fall times reaches alarming 101.3% under the near-threshold voltage.

Table 3.2 – Maximum difference between rise and fall delays in DEs implemented in 65nm.

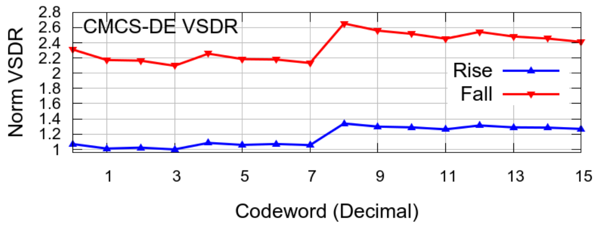
	@ Nominal	@ Near Threshold
MUX-DE	10.1%	1.2%
DCCS-DE	2.7%	47.9%
CMCS-DE	7.2%	101.3%



(a)



(b)



(c)

Figure 3.14 – Normalized *VSDR* values for DEs implemented in 65nm: (a) MUX-DE, (b) DCCS-DE and (c) CMCS-DE.

As for *VSDR* variations in the 65nm technology, it is possible to observe a considerably higher impact here than that obtained in the 180nm technology, in some cases. Figure 3.14 presents the normalized *VSDR* values, where it is possible to observe that the DEs that manipulate current (DCCS and CMCS) show *VSDR* variations between 98.1% for the DCCS-DE and 165.1% for the CMCS-DE. Although impacts on these DEs are more significant, the MUX-DE displays less *VSDR* variation, if compared to the same DE implemented in 180nm. In 65nm the maximum observed variation is 26.1%.

Finally, results for the DEs implemented in the 28nm FDSOI technology revealed some interesting characteristics with regard to voltage scaling. Figure 3.15 leaves clear that there are some fluctuations in rise and fall times, including in the MUX-DE. The way all DEs were implemented in this technology followed exactly the same steps, and variations can be noticed already at nominal supply. Here, the DCCS-DE show non-monotonic behavior for the delay under near-threshold voltage. These are easily identifiable between codewords 4 and 5 and between codewords 10 and 11 .

To facilitate the analysis of the rise and fall time differences of each implementation, Table 3.3 depicts these using maximum percentage variations. From this Table, it is possible to verify that both DCCS-DE and CMCS-DE present significant variations. DCCS-DE reaches a very large difference for near-threshold (i.e.224.1%), thus requiring an impractical delay margin.

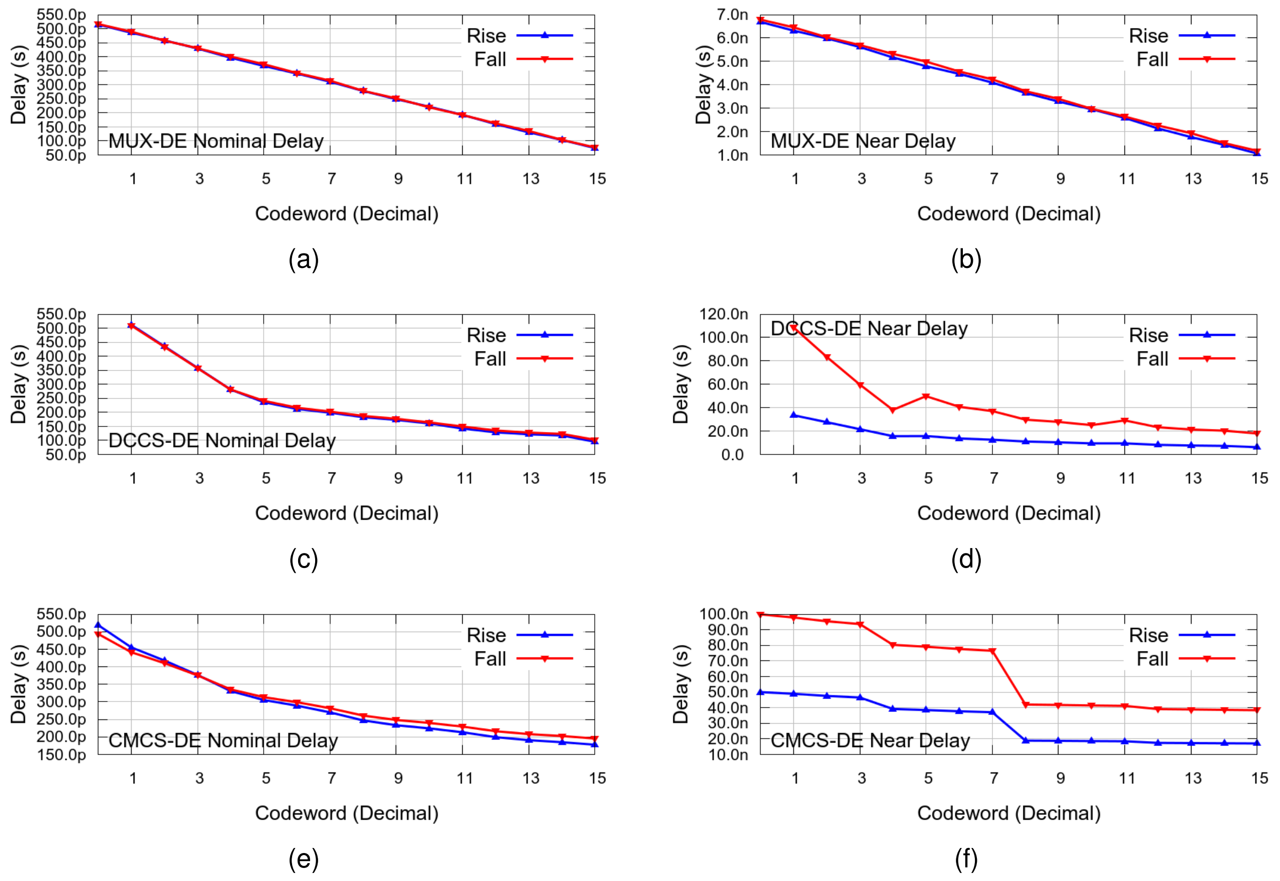


Figure 3.15 – Delay produced by programmable DEs implemented in 28nm: (a) MUX-DE @ nominal and (b) MUX-DE @ near, (c) DCCS-DE @ nominal and (d) DCCS-DE @ near, and (e) CMCS-DE @ nominal and (f) CMCS-DE @ near.

Table 3.3 – Maximum difference between rise and fall delays in DEs implemented in 28nm.

	@ Nominal	@ Near Threshold
MUX-DE	1.7%	4.0%
DCCS-DE	6.7%	224.1%
CMCS-DE	5.1%	99.6%

Concerning *VSDR* variations, Figure 3.16 state the relationships for all DEs. This time the *VSDR* of the DCCS-DE presents variations that are the largest surpassing 280%. Meanwhile, CMCS-DE also displays large variations reaching 255.2% between the rise time for codeword 8 and the fall time for codeword 7.

The discussion in this Section demonstrates that VS can considerably affect delays in DEs. DEs based on current control, like the DCCS-DE and the CMCS-DE, showed to be much more sensitive to VS in comparison with the MUX-DE. However, what is more salient in the experiments are the differences between the rise and fall time curves for both delay and *VSDR*. No matter how well dimensioned to operate under nominal supply the DEs are, they are deeply affected by VS operation. The differences between the rise and fall times and the observed *VSDR* variations in near-threshold point to the need to develop new

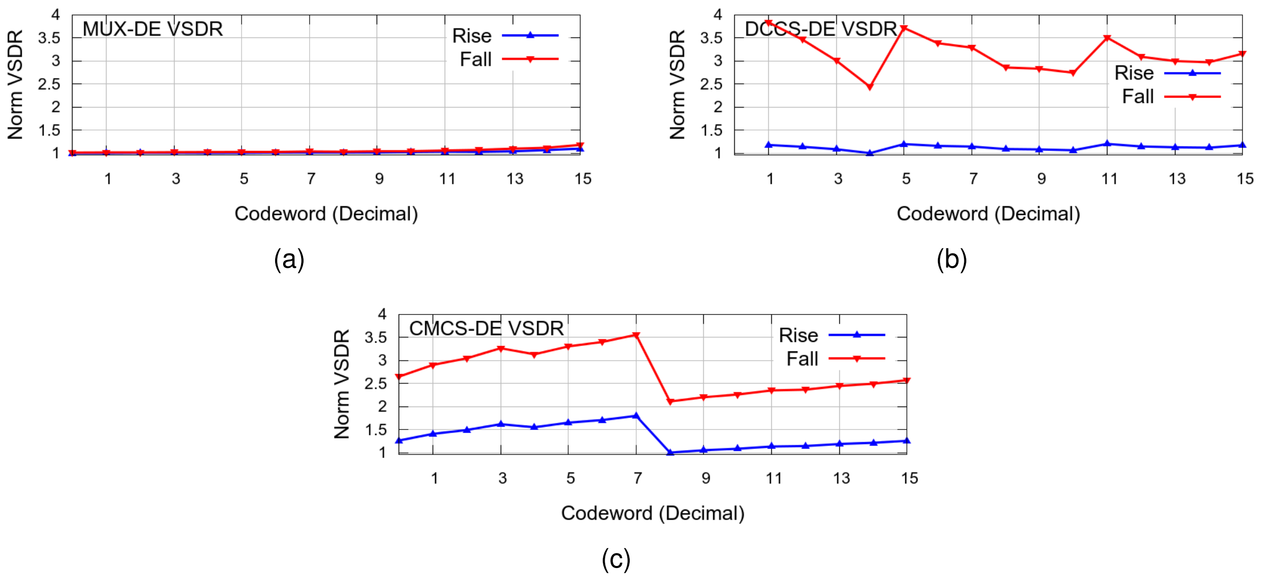


Figure 3.16 – Normalized *VSDR* values for DEs implemented in 28nm: (a) MUX-DE, (b) DCCS-DE and (c) CMCS-DE.

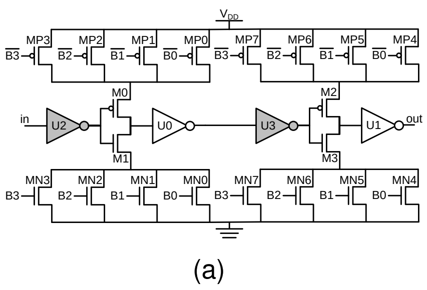
techniques to reduce such differences and to provide better performance to DEs intended for use in 2-phase BD circuits. The next Section provides a first overview of techniques to employ to this end.

3.3.2 Optimizing Delay Elements to Operate under Voltage Scaling

To reduce differences and variations between rise and fall delays and VSDRs, the proposition is to employ techniques similar to those described in Section 3.1.2. The idea is to force all signals to always traverse a same set of distinct transistor kinds, potentially balancing delays for every transition type. Meanwhile, some care is needed to maintain delay balancing close to ideal values.

It is important to emphasize that rise and fall time compensation is already part of the MUX-DE architecture. The employed design utilized inverters and inverting multiplexers specifically developed for clock distribution. These cells are usually designed and implemented to balance rise and fall times. Besides, consecutive signal inversions play the same role of optimizations suggested here.

Just as in the case of the single inverter, the proposal here is to replicate the whole original DE architecture in series, i.e. the output of a first instance of a DE is connected to the input of a second instance of the same DE. To guarantee that both DEs have similar delays, it is not enough to simply connect instances consecutively. The input signal ramp of the first instance must be identical to the input signal ramp of the second. To reinforce this characteristic both DEs receive inverters in their inputs. Another relevant item is to maintain



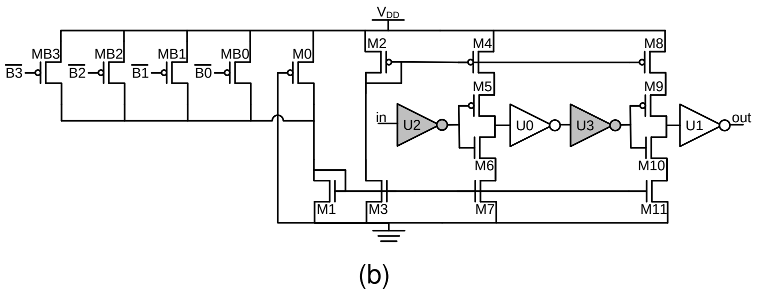


Figure 3.17 – Schematic of DEs after applying the balancing rise and fall times optimization: (a) Balanced DCCS-DE and (b) Balanced CMCS-DE.

a constant output load. Since both the DCCS-DE and the CMCS-DE already have inverters at their outputs, the output load is already identical in both these DEs. Thus, for these, it is only necessary the addition of the two input inverters. The final result is the optimized architectures depicted in Figure 3.17 where inverters *U2* and *U3*, marked as gray gates are the newly added components.

One of the advantages of the fact that CMCS-DE operating with current mirrors (transistors *M1* and *M2*) is the possibility of sharing the mirroring signal with other mirrors (e.g. *M8* and *M11*), eliminating the need to duplicate the control part of the DE. Nonetheless, the use of current mirrors causes a too high static power consumption, which may severely impair its use in practice. Regarding the DCCS-DE, the possibility of area reduction is not available and the optimized version doubles the occupied chip area. On the other hand, as the inverters formed by transistors *M0/M1* and *M2/M3* automatically disconnect the unused parts of the DE, the DCCS-DE static consumption is extremely reduced.

The implementation of both DCCS-DE and CMCS-DE in their revised versions as presented here followed criteria similar to that described previously for the non-optimized versions of the DEs. The simulation environment was configured to replicate the previous experiments *mutatis mutandis*. The addressed technologies were again 180nm and 65nm bulk and 28nm FDSOI, allowing to identify the benefits of the proposed optimizations.

Starting by the analysis of rise and fall time differences when applying VS to the 180nm optimized version of the DEs, it is possible to perceive significantly improved results, as Figure 3.18 demonstrates. Regardless of the improvements in delay differences, the lack of monotonicity in both DEs in the near-threshold supply is remarkable. Table 3.4 summarizes the maximum differences between rise and fall times for the DCCS-DE and the CMCS-DE implementations. When compared to previous related results for the non-optimized versions, there is a reduction of 44,7% maximum variation to only 2.2% in the DCCS-DE and from 19.3% to only 3.1% in the CMCS-DE.

Improvements also occur in the *VSDR* behavior. Figure 3.19 presents the *VSDR* variations for both modified DEs. It is clear that rise and fall *VSDRs* follow each other closely, indicating a considerable reduction in the implied margins. The DCCS-DE achieves a reduc-

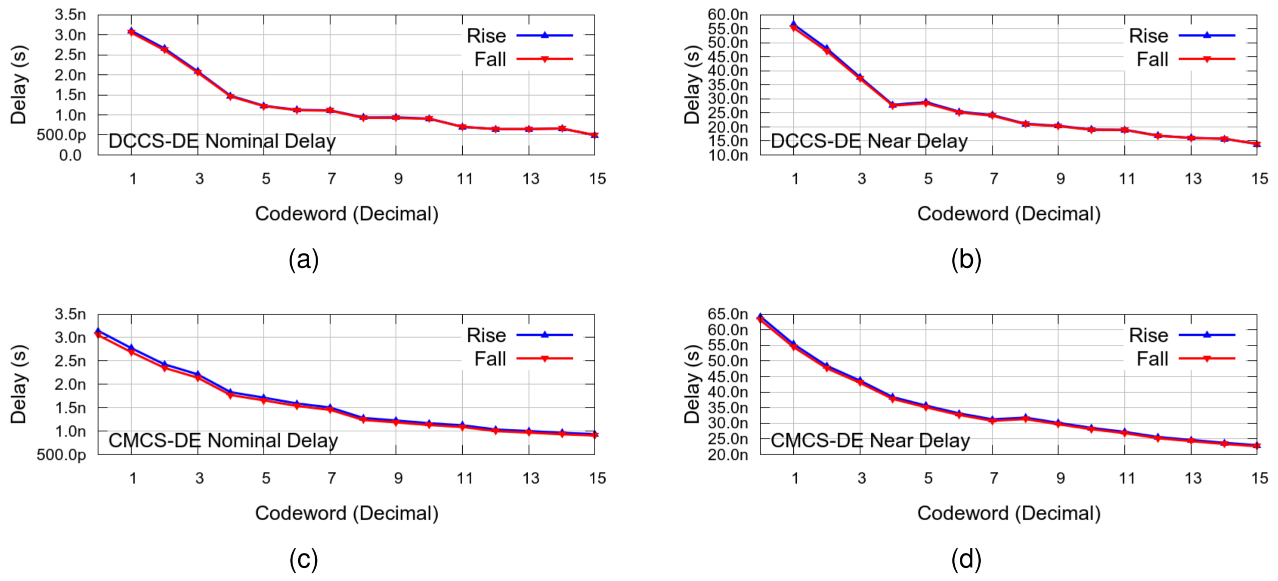


Figure 3.18 – Delay produced by programmable DEs implemented in 180nm using the proposed optimization technique: (a) DCCS-DE @ nominal and (b) DCCS-DE @ near, and (c) CMCS-DE @ nominal and (d) CMCS-DE @ near.

Table 3.4 – Maximum difference between rise and fall delays in DEs implemented in 180nm.

	@ Nominal	@ Near Threshold
DCCS-DE	1.6%	2.2%
CMCS-DE	3.1%	1.7%

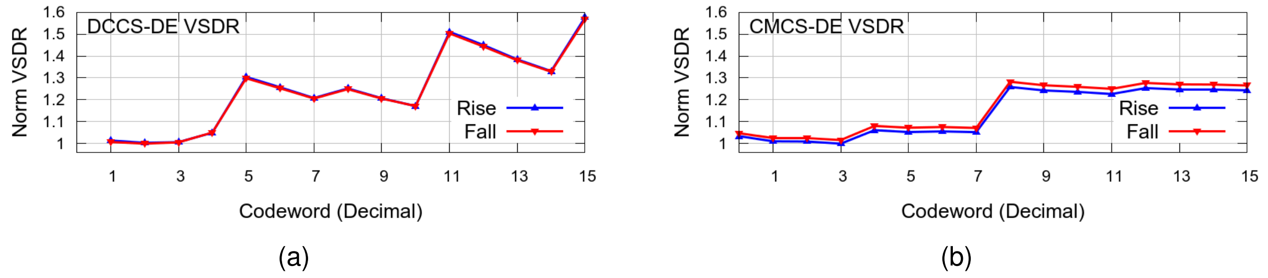


Figure 3.19 – Normalized *VSDR* for each DEs implemented in 180nm: (a) MUX-DE, (b) DCCS-DE and (c) CMCS-DE.

tion of 128.3% to 57.7% under the proposed optimization, cutting margins to less than half the previous values. The CMCS-DE also obtains improvements, going from 44.5% of maximum *VSDR* variation to only 28.2%.

In the 65nm technology, reductions are similar to those observed in 180nm. Figure 3.20 presents the delays produced for the several DE configurations. In this experiment, differences between rise and fall time are at most 6.4% for the DCCS-DE and 4.4% for the CMCS-DE, as Table 3.5 denotes. In the case of the DCCS-DE, there is a reduction of more than 86% in the delay margin. In the CMCS-DE, the reduction is still more pronounced,

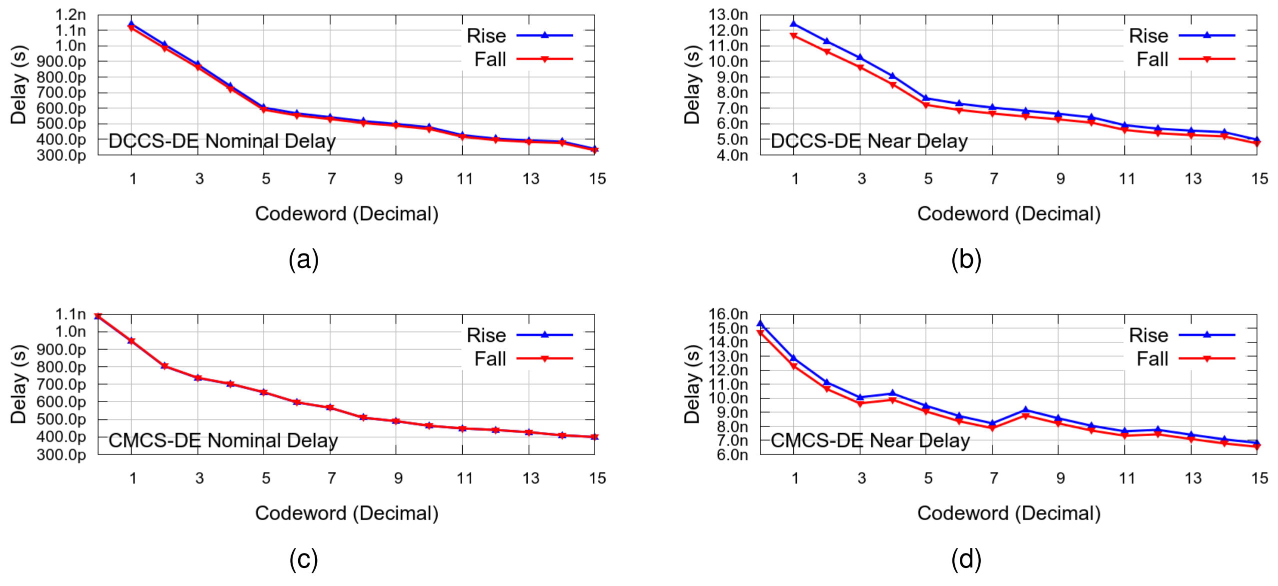


Figure 3.20 – Delay produced by programmable DEs implemented in 65nm using the proposed optimization technique: (a) DCCS-DE @ nominal and (b) DCCS-DE @ near, and (c) CMCS-DE @ nominal and (d) CMCS-DE @ near.

Table 3.5 – Maximum difference between rise and fall delays in DEs implemented in 65nm.

	@ Nominal	@ Near Threshold
DCCS-DE	2.3%	6.4%
CMCS-DE	0.3%	4.4%

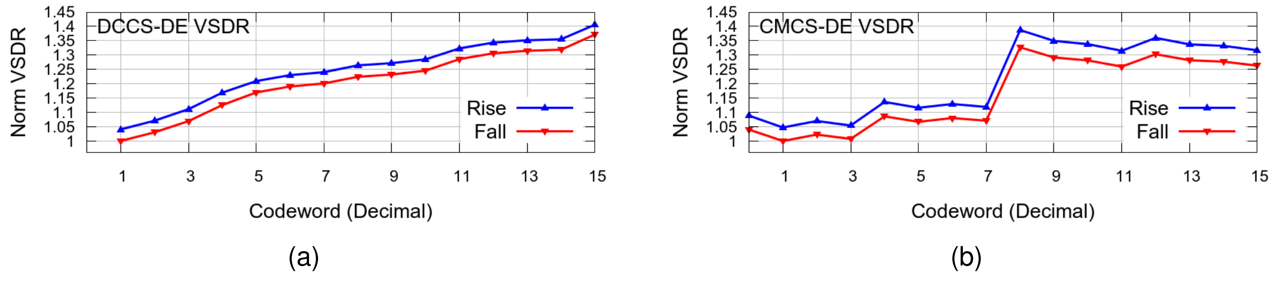


Figure 3.21 – Normalized *VSDR* for each DEs implemented in 65nm: (a) MUX-DE, (b) DCCS-DE and (c) CMCS-DE.

reaching 95%. In this technology (65nm bulk), only the CMCS-DE shows failures in DE monotonicity in near-threshold voltage.

The *VSDR* comparison results in 65nm appear in Figure 3.21. Here, variations are less than 40.6% for the DCCS-DE and less than 38.7% for the CMCS-DE. These values are respectively 58% and 76% better than the margins obtained for the non-optimized versions of DEs in this technology. This means a reduction to at least half of the previous *VSDR* margins.

As for the delay results in the 28nm technology, it can be stated that there is a considerable reduction in the difference between rise and fall times. Figure 3.22 details the

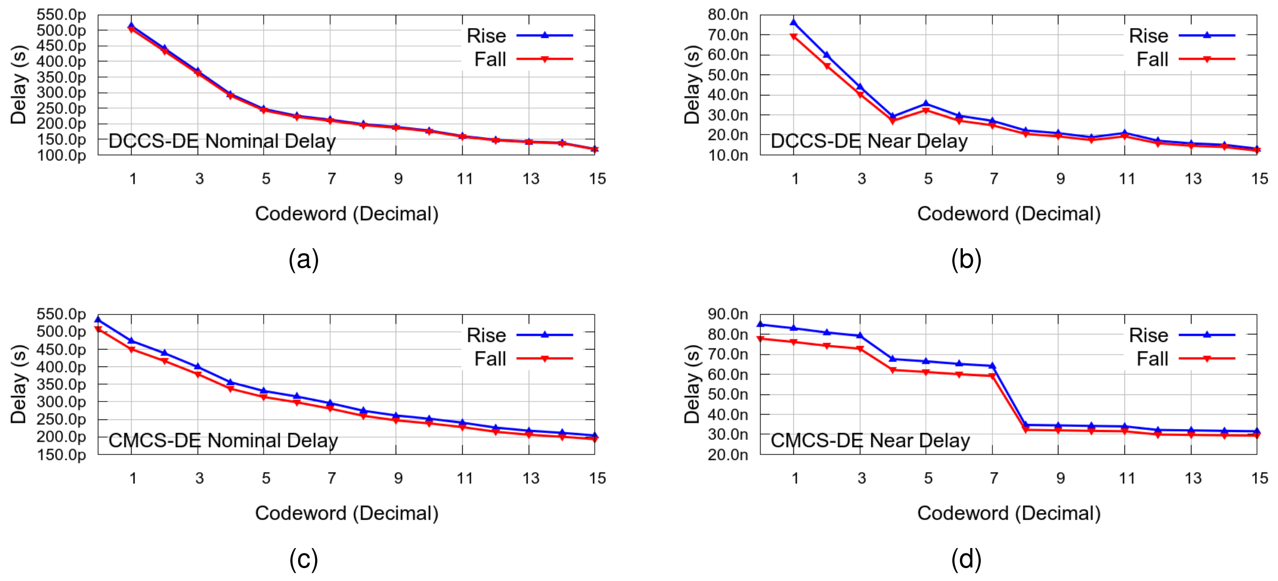


Figure 3.22 – Delay produced by programmable DEs implemented in 28nm using the proposed optimization technique: (a) DCCS-DE @ nominal and (b) DCCS-DE @ near, and (c) CMCS-DE @ nominal and (d) CMCS-DE @ near.

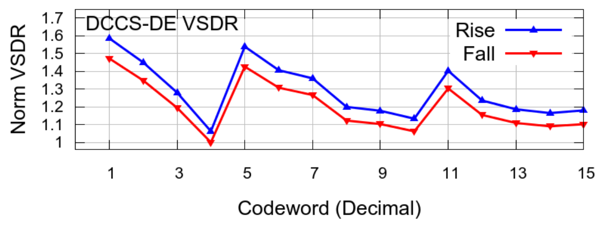
Table 3.6 – Maximum difference between rise and fall delays in DEs implemented in 28nm.

	@ Nominal	@ Near Threshold
DCCS-DE	2.0%	9.9%
CMCS-DE	5.1%	9.0%

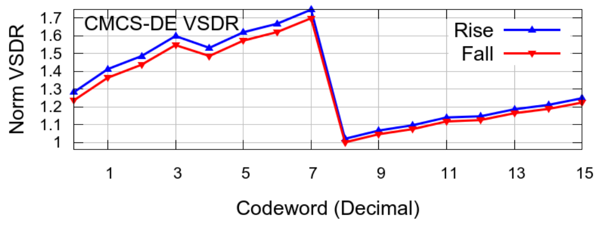
delays generated when at the nominal voltage and at the near-threshold voltage as well, for the optimized DE architectures. Table 3.6 transforms into values the differences observed in the graphs. With a difference of 9.9% for the DCCS-DE and of 9.0% for the CMCS-DE, it is possible to identify reductions of respectively more than 95% and more than 90% in the delay margins for the evaluated DEs, when these are compared to the previous implementations.

As expected, the *VSDR* margins are also affected. Figure 3.23 presents the *VSDR* variations for the optimized DCCS-DE and CMCS-DE in this technology. *VSDR* margins here are 59.9% for the DCCS-DE and 72% for the CMCS-DE. Regardless the fact that these are considerably high numbers, they are less than one third the margin obtained without optimizing the respective DEs. In fact, these numbers correspond to a reduction of more than 74% in the DCCS-DE margin and more than 76% in the CMCS-DE margin.

Even though considerable *VSDR* reduction in variations is achieved, neither the DCCS-DE nor the CMCS-DE are even close to the numbers obtained by the MUX-DE original implementation. Excluding the CMCS-DE *VSDR* in 180nm which is smaller with the optimization, the MUX-DE shows the smallest *VSDR* variations during reconfiguration. This is due to a basic characteristic of the MUX-DE: homogeneity. Both the DCCS-DE and the CMCS-DE employ current-based control circuits. To perform the control task, the dimensions of these transistors are specific and different for each control transistor. Each binary



(a)



(b)

Figure 3.23 – Normalized *VSDR* for each DEs implemented in 28nm: (a) MUX-DE, (b) DCCS-DE and (c) CMCS-DE.

input controls a transistor with larger L or smaller W than the subsequent input. As discussed in the start of this Chapter, in Section 3.1, the *VSDR* varies significantly with the transistor dimensions, which for the current-controlled DE produces *VSDR* fluctuations, depending on the specific programming codeword.

The MUX-DE is also sensitive to the just described effect. As the DE configuration approaches the minimum delay, less inverters are in the path between the input and the output. In fact, the least delay comprises a path that only contains multiplexers. Since these have a structure which is distinct from the inverter, their *VSDRs* vary differently as well. For example, in the 180nm technology multiplexers are implemented with NAND and NOR gates. The impact of these is significantly larger, since these gates have a structure very different from inverters.

Accordingly, an idea is to propose a new delay element that reduces *VSDR*. To this end, this new DE must be formed by the least possible number of distinct library cells to achieve the largest possible homogeneity. Ideally, thus, such a DE must be constructed using only one cell type and one single cell drive.

Starting from the MUX-DE architecture, the one presenting the best results overall so far, it can be observed that the cell responsible for the *VSDR* variations is the multiplexer. Since this is a device that cannot be avoided in this architecture, the suggestion is to employ it as the base to construct the new DE.

A multiplexer can be implemented in several different ways. However, the way that takes the least area, and which is the most commonly used in cell libraries is a topology that employs tristate gates. Figure 3.24(a) presents a multiplexer implementation for recent technologies. From this simple module, it is possible to design a new MUX-D, where inverter cells are substituted by tristates. To reduce power consumption, the multiplexer selection signal is also employed to activate and deactivate the tristates in the respective stage. Figure 3.24(b) shows the structure of the new, tristate-based MUX-DE.

With the new MUX-DE design, the results for differences between rise and fall times are not that distinct from the original MUX-DE. Figure 3.25 presents the delay graphs os graphs produced by the new MUX-DE in the three addressed technologies. Next, Table 3.7

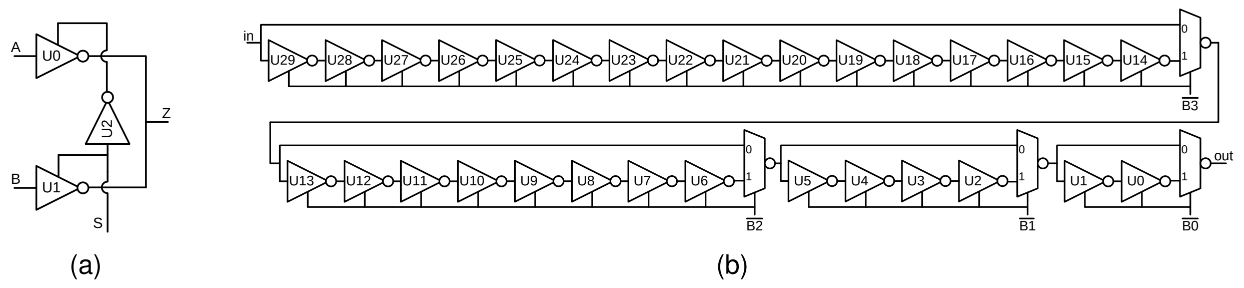


Figure 3.24 – (a) A multiplexer designed with tristates. (b) The new tristate-based MUX-DE.

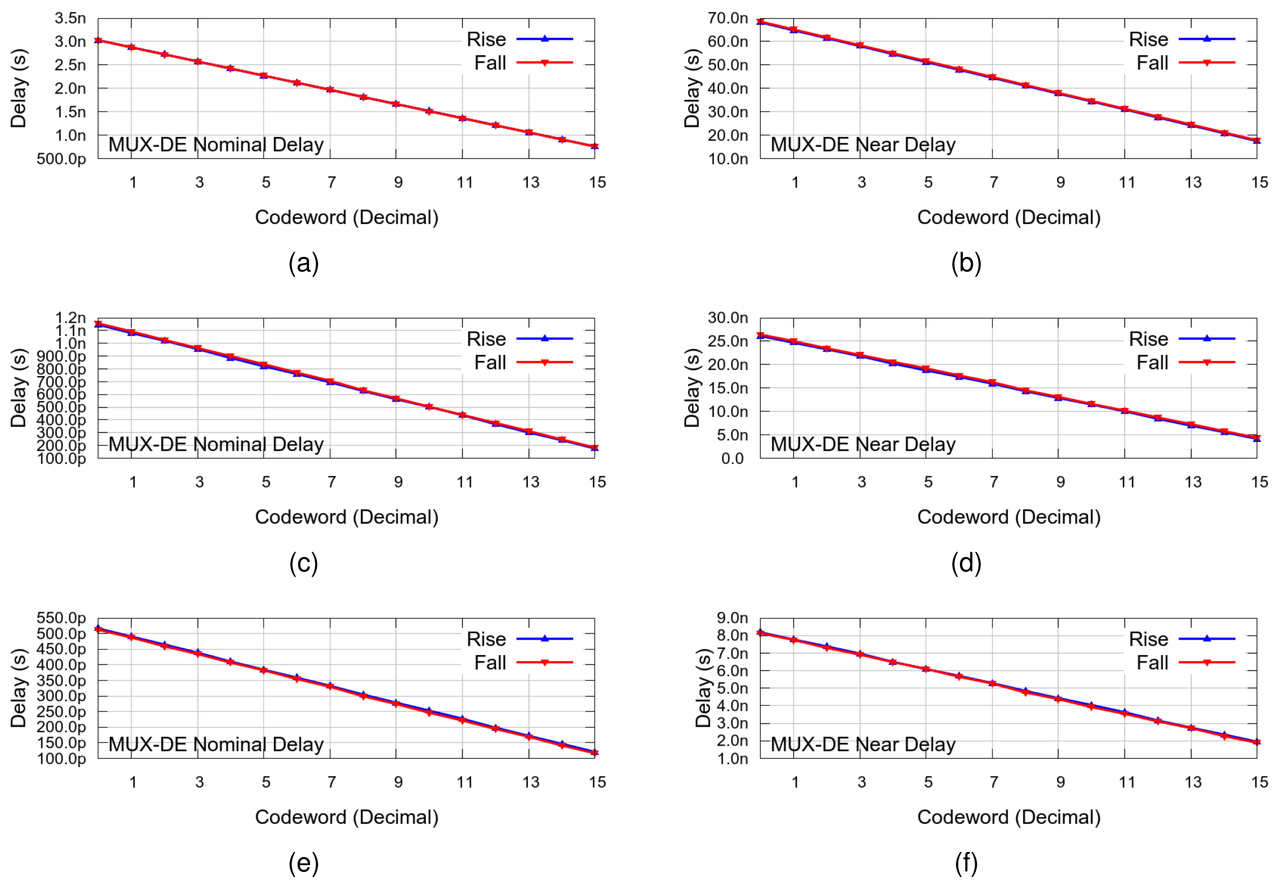


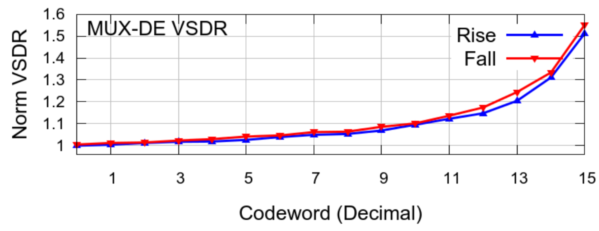
Figure 3.25 – Delay produced by new tristate-based MUX-DE implemented in: (a) 180nm @ nominal and (b) 180nm @ near, (c) 65nm @ nominal and (d) 65nm @ near, and (e) 28nm @ nominal and (f) 28nm @ near.

presents the maximum differences for rise and fall times for the new MUX-DE, comparing it to the original MUX-DE. From the Table, it is possible to more clearly approach the benefits of working with tristate-based MUX-DEs. There is a reduction of at least 45% in the margins, being the most significant reduction (almost 80%) observed in the 65nm bulk CMOS technology that passes from 10.1% to only 2.1% maximum variation.

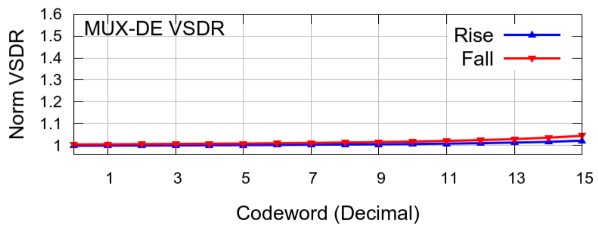
Although the improvement in reducing delay differences in rise and fall times, the largest benefit of the tristate-based MUX-DE is its behavior when submitted to VS. The graphs comparing the *VSDRs* of the original MUX-DE with the tristate-based MUX-DE are

Table 3.7 – Maximum difference between rise and fall delays in the original MUX-DE and in the tristate-based MUX-DE.

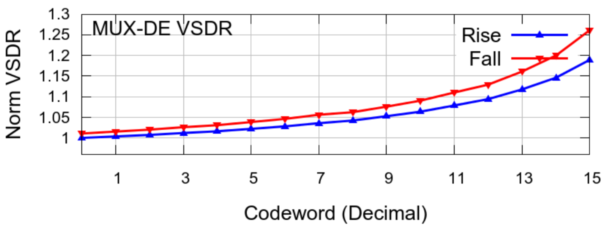
	Original MUX-DE		Tristate based MUX-DE	
	@ nominal	@ near	@ nominal	@ near
180nm	1.7%	3.2%	0.3%	0.8%
65nm	10.1%	1.2%	1.9%	2.1%
28nm	1.7%	4.0%	2.2%	1.8%



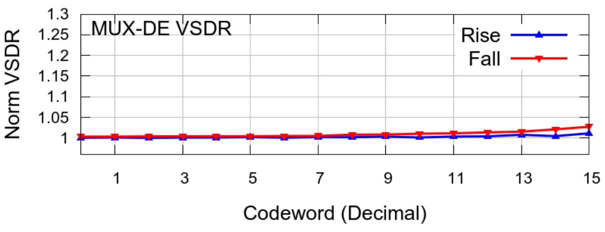
(a)



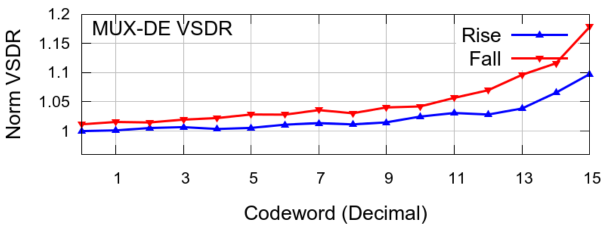
(b)



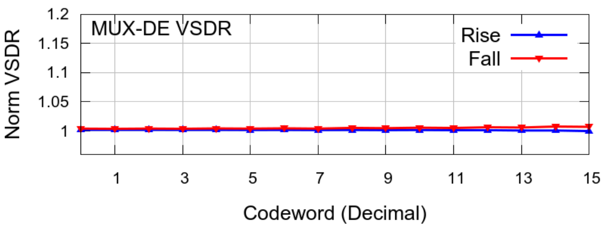
(c)



(d)



(e)



(f)

Figure 3.26 – The normalized *VSDR* variation values for: (a) the original MUX-DE @ 180nm; (b) the tristate-based MUX-DE @ 180nm; (c) the original MUX-DE @ 65nm; (d) the tristate-based MUX-DE @ 65nm; (e) the original MUX-DE @ 28nm; (f) the tristate-based MUX-DE @ 28nm.

depicted in Figure 3.26. The mere observation of the graphs already enables the identification of a considerable improvement in *VSDR* variations along the changes of delay programming values. The previous results in the 180nm technology had a margin of 55.2%, that passed to only 4.5% (a reduction of more than 90% in margins). In the 65nm implementation, the *VSDR* passed from 26.1% to just 2.7% in the tristate-based MUX-DE. Finally, in the 28nm technology, the *VSDR* reduced from 17.8% to just 0.8%, resulting in a reduction of more than 95% in margins.

Table 3.8 compiles most results presented in this Chapter. In this Table, it is possible to identify which delay element is more interesting for each design need. For example,

Table 3.8 – A comparison of programmable DEs relevant design parameters. Blue numbers indicate best values, while red numbers indicate worst values.

		Original MUX-DE	DCCS-DE	CMCS-DE	Tristate based MUX-DE
180nm	Active Area (um²)	26.064	25.931	6.545	36.763
	Avg. Idle Power @ nominal (nW)	2.088	0.46	360,729	9.36
	Avg. Idle Power @ near (nW)	0.336	0.076	1,945.26	3.42
	Avg. Energy per Transition @ nominal (fJ)	896.22	690.77	147.42	538.02
	Avg. Energy per Transition @ near (fJ)	85.56	22.99	6.36	44.76
	Rise/Fall Difference Margin	3.2%	2.2%	3.1%	0.8%
	VSDR Margin	55.2%	57.7%	28.2%	4.5%
65nm	Active Area (um²)	5.302	2.399	1.386	18.287
	Avg. Idle Power @ nominal (nW)	8.64	2.17	218,058	15.12
	Avg. Idle Power @ near (nW)	1.045	0.228	2924.02	3.465
	Avg. Energy per Transition @ nominal (fJ)	137.88	55.92	13.56	184.68
	Avg. Energy per Transition @ near (fJ)	27.77	4.81	1.98	33.71
	Rise/Fall Difference Margin	10.1%	6.4%	4.4%	2.1%
	VSDR Margin	26.1%	40.6%	38.7%	2.7%
28nm	Active Area (um²)	2.508	1.233	0.326	2.793
	Avg. Idle Power @ nominal (nW)	42.2	9.18	71,411	12.6
	Avg. Idle Power @ near (nW)	5.2	1.07	280.5	1.8
	Avg. Energy per Transition @ nominal (fJ)	66.9	12.6	1.4	14.2
	Avg. Energy per Transition @ near (fJ)	16.05	2.8	0.3	3.25
	Rise/Fall Difference Margin	1.5%	9.9%	9.0%	2.2%
	VSDR Margin	17.8%	59.9%	72%	0.8%

applications where area restrictions are most important can benefit from using a CMCS-DE. Concerning low power, on the other hand, the most adequate is the DCCS-DE, specially if static power is the main focus. However, if delay margins are the relevant parameter, the best option is using the MUX-DE, with preference for its tristate-based implementation, given its benefits, specially when operating under VS.

Some parallel studies have already derived from the work described in this Chapter. Singhvi et al. in [SMT+15] proposed techniques to enhance the capacity to generate delays in programmable DEs, through the polarization of the *body* terminal in FDSOI transistors. The impacts of these techniques on the *VSDR* are also discussed in this publication. Later, Tadros et al. in [THG+16] proposed a method to dimensioning delay elements with a specific target on acting over the *VSDR*. Further work on this subject is currently under elaboration and some results are discussed later in this document, in Chapter 5.

4. POST-SILICON TEST ANALYSIS ON BUNDLED-DATA DESIGNS

The main goal of manufacturing tests is to detect any defect that occurs in a fabricated circuit. However, several trade-offs are often necessary to obtain the required test coverage quality at a minimal cost [TPC11]. For large electronic systems, testing accounts for 30% or more of the total Integrated Circuit (IC) cost. Delay defects are specially important in asynchronous designs, where self-timing templates such as Bundled-Data (BD) are intrinsically time-dependent [URV11]. In this case, a suitable sequence of test vector pairs (V_1, V_2) has to be applied for testing time specifications.

The transition and path-delay fault models can guarantee good fault covering. However, the current complexity of circuits that derives from the integration capacity of current technologies together with the intrinsic variability of fabrication processes creates significant increases in the number of distinct test cases [TPC11]. Since the test cost is directly dependent on the test execution time, the increase in test cases has a direct impact on its cost [WST10]. This points to the need to improve coverage without significantly increment the number of test cases.

The Built-In Self-Test (BIST) technique reduces test costs, but delay defects are not easily detected by random inputs, requiring a large number of patterns [TPC11]. In this way, the alternative of using scan chains can lead to a better coverage, but the complexity involved in it need to be considered. However, with the trend to employ low power circuits, more test cases are added, due to the impact on the delay caused by techniques such as voltage scaling (VS), as already discussed in Chapter 3. That Chapter discusses research on the effects of VS on delay and points to other related works developed in parallel with the research described here, including works like [GMC15, THG+16]. Thus, it is indispensable to improve coverage without a significant increase in the number of tests. This is often achieved in synchronous circuits, under conventional operation using *At-Speed* (AS) tests.

An alternative to increase the coverage of delay defects and Small Delay Defects (SDDs) is to employ *Faster-Than-At-Speed* (FTAS) test techniques. FTAS test improves delay fault detection, increasing the low slack time detection [YHIF11]. In synchronous circuits increasing the operating frequency the same number of patterns results in larger detection of delay defects [XZVH09, YHIF11, TPC11]. However, applying this technique requires known operating frequencies to reduce the amount functional devices in these naturally more demanding tests [XZVH09].

If on the one hand, BD circuits are less sensitive to the influence of IR-Drop, responsible for a large part of complexity of applying FTAS tests, on the other hand, BD circuits do not have a reference signal such as the clock. The use of delay elements adds complexity to the test process exactly because of the missing reference signal. Delay elements are sensitive to process variations, which cannot be ignored during delay fault tests.

Algorithm 4.1 – Synchronous delay fault test.

```
1: $clock \leftarrow clock_{test}$              ▷ $clock_{test}$ is the frequency used during the test execution
2: for all $P$ patterns do                     ▷ $P$ is the number of patterns for synchronous delay fault test
3:     Run delay fault test pattern $(V_1, V_2)$                ▷ LOS, LOC or Enhanced
4:     Collect results
5: end for
```

In the next two Sections two circuit types are respectively analyzed: conventional BD and resilient BD architectures. The intention is to show the benefits of each implementation, contrasting these to the test complexity they incur. The complexity of the two models is investigated as well as the impact of low power implementations on their testability. This Chapter does not look for a solution to the test complexity problem. It merely brings some light about issues that can potentially make the fabrication of such circuits harder, or even impractical.

4.1 Conventional Bundled-Data Delay Test Analysis

One of the advantages of bundled-data design is its partial compatibility with conventional CAD tools [BOF10]. Because data representation relies on single-rail Boolean encoding, the synthesis of Combinational Logic (CL) takes place exactly in the same way as in conventional synchronous design. Thus, the test vector generation mechanisms conceived to test the CL of synchronous circuits can be employed to test BD circuits as well [SO15]. Consequently, it is possible to apply the classic approach of delay test presented in Section 2.5, i.e., employing (V_1, V_2) vector pairs.

To illustrate and as a reference, Algorithm 4.1 presents the execution of the delay fault test of a synchronous circuit this will be converted to a BD circuit. Here, the total time spent to cover delay faults (T_{total}) can be expressed by Equation 4.1.

$$T_{total} = P \times T_{onepattern}, \tag{4.1}$$

In Equation 4.1 P is the number of patterns for the synchronous delay fault test and $T_{onepattern}$ is the time to run one delay fault test pattern.

Current works usually approach the test of the control blocks in BD design [SM06, RGP+15, KA16, MCFB16]. In some cases, even the CL delay is investigated, as proposed e.g. by Sato and Ohtake [SO15]. However, just this single reference [SO15] could be found that addresses the delay difference between the CL and the associated delay element. Unfortunately, Sato and Ohtake propose just to readjust the DE based on the computed slack, without event testing it again to guarantee its correct functionality. Works like [SM06, MCFB16] simply assume their DEs operate appropriately, without testing them.

Different from what takes place in synchronous circuits, which have available at least one global reference, namely the clock, BD templates cannot rely on such an exact reference for its DEs, due to process variations. Even if DEs can be compensated using post-silicon evaluation based e.g. in measurements of in-chip ring oscillators, process variations are still a source of uncertainties in delay elements.

Externally assessing the DE delay would be a possibility to have available a reference equivalent to a clock. However, the cost of specific (analog) equipment to capture these values, and to measure the interference caused by additional gates and extra IC pads can point to the presence of defects that do not in fact exist. At this point, it is worth to consider if it still makes sense to assess the exact delay of DEs.

Since BD design techniques are self-contained (i.e. each CL of each stage has its own DE), if some CL is slower than expected, this does not necessarily imply a defect. If the DE delays also follows the CL delay and if the performance is not below the specified (maximum execution time), the circuit is classified as free from delay faults.

Another major point is the Small Delay Defect (SDD) analysis for more recent technologies. For example, one of the characteristics of conventional reconfigurable DEs (e.g. CMCS-DE, DCCS-DE and MUX-DE) is that there are selected delay configurations that may alter the subset of transistors and gates defining the delay by as much as 100% with regard to another neighbour configuration. For example, changing the configuration control word from "0111" to "1000" (in a scheme based on binary counting), the delay path in a MUX-DE completely changes. In situations like these process variations of a path can produce results much higher or much lower than the required one. Due to the uncertainties in the delay changes, Faster-Than-At-Speed (FTAS) tests can provide results out of the range required for yielding good chips [XZVH09]. For this reason, this work concludes that At-Speed (AS) is the best alternative for BD circuits.

4.1.1 Combinational Logic vs DE Delay

This Section deals with the complexity of AS tests considering DE variations. To facilitate the understanding of test complexity growth, the set of patterns (V_1, V_2) employed during the test will always be the same for all analyses. This abstracts which delay test model (transition or path-delay) is applied.

Sato and Ohtake proposed [SO15] to capture the CL delay to program a variable DE. To effectuate this kind of measurement these Authors use a Time-to-Digital Converter (TDC), based on scannable flip-flops parallel to a DE. During the test, the FF inputs are directly connected to selected intermediate DE positions. To detect a transition in the CL critical path, the TDC stops the signal propagation and the delay can be obtained by multiplying the number of FFs containing logic 1 by the period of the clock utilized in the capture process.

Analyzing the Sato-Ohtake proposal, it is possible to observe that the setup time of scannable FF is considerably larger than that of a simple logic gate (approximately equal to the propagation delay of two buffers, or four inverters). This reflects in the generated delay capture resolution. Besides, it is necessary to assume that a transition inside the FF setup period can occur. This implies admitting at least one reference clock cycle of margin to account for wrong readings, which adds margins to the DE delay. In addition, each DE configuration requires a different test execution, which considerably increases the test cost. Finally, it is still necessary to guarantee the absence of glitches in the evaluation path, which is a considerably hard action for single-rail encoded circuits.

From the previous discussion, a test alternative would be not to test the DE separately, but test it in conjunction with the associated CL and controller. In this context, there are three possibilities to capture data contained in registers: *expected data*, *reset* and *unexpected data*. For the *expected data* case, the DE possesses a delay adequate to the associated stage and the value acquired by the scan-chain will be the expected response to V_2. For the *reset* case, the DE takes too long for the controller to signal the capture in the register, acquiring through the scan-chain the response of the CL to V_1. Finally, in the *unexpected data* case, the delay generated by the DE is insufficient, signalling the capturing of data in the register before its complete arrival. In this last case, the response is different from the answers to both V_1 and V_2.

Even though the DE offers the expected duration to allow that the CL conclude its computation, a BD circuit can be sensitive to some of the DE intrinsic characteristics, depending on the selected protocol. In a 4-phase protocol, just one of the propagation delay times is considered, making it necessary to apply just one test. However, a BD circuit operating under a 2-phase protocol indistinctly employs both rise and fall propagation delays. Thus, two tests are needed for each (V_1, V_2) pattern, which adds at least one extra test.

In BD circuits, ring oscillators can be employed to detect process variations and provide the DE with delay adjustment parameters. This adjustment is expected to reduce margins and improve the circuit performance.

From the initial local adjustments, the circuit is able to pass through the delay fault test phase. The actions described in Algorithm 4.2 are a suggested sequence to execute an AS test for a BD circuit. In this case, a signal similar to that proposed in [SM06] can be used to lock the DE under the V_1 load and V_2 collect procedures. Since the case refers to a circuit operating under a 2-phase protocol, the procedure needs to be executed in the two configurations of the delay element, resulting in Algorithm 4.2.

In Algorithm 4.2 P is the number of delay defect tests usually employed for synchronous circuit, V_1 is the vector responsible for resetting/setting the CL and V_2 is the vector

Algorithm 4.2 – Bundled-Data delay test with fixed delay for a 2-phase circuit.

```
1: $delay_{codeword} \leftarrow \mathcal{I}$                ▷ $\mathcal{I}$ is the codeword supposed to produce the necessary delay
2: for all $P$ patterns do                                   ▷ $P$ is the number of patterns for synchronous delay fault test
3:     for $i = 0$; $i \leq 1$; $i++$ do
4:         $delay \leftarrow$ ‘$i$‘                          ▷ Logic value
5:         Run delay fault test pattern ($V_1$, $V_2$)       ▷ LOS, LOC or Enhanced
6:         Collect results
7:     end for
8: end for
```

that excites the path under test. Hence, the complexity considers the two DE states, resulting in Equation 4.2.

$$T_{total} = 2 \times P \times T_{onepattern} \tag{4.2}$$

Compared to a synchronous circuit, the (V_1, V_2) pattern pair is executed only at one of the edges of the clock, resulting in the execution of just P patterns. Clearly, 2-phase BD circuits require at least twice the time to execute the test.

4.1.2 Voltage Scaling Delay Test

From the results reported in Chapter 3 and from the *VSDR* results presented by Gibiluka in [Gib16], one reaches the conclusion that voltage variation can impact different circuits in different ways. In some cases, such as the TSMC 180nm CMOS technology, the *VSDR* step of increment can be distinct from that expected at intermediate voltage points. This is due to the impact of α in this technology, which causes minimum size gates to depend less on the supply voltage than gates with $3 \times L_{min}$ at intermediate voltage points. This can be verified by comparing Figures 3.5(b) and 3.7.

Varying L is unusual in standard cells, but an identical effect arises with stacked transistors, which are quite common in many standard cells [KNC02]. Thus, the possibility of *VSDR* variations in the combinational logic as well as in the delay element can lead to an unexpected behavior along several supply voltage values. Accordingly, one direct consequence to guarantee the correct circuit operation regarding delay values consists in testing the circuit at all defined supply levels. Starting from Algorithm 4.2, presented in Section 4.1.1, it can be observed that the resulting complexity grows in the same proportion as the defined voltage levels on which the circuit is designed to operate. Algorithm 4.3 demonstrates this increase.

This Algorithm does not make it explicit, but probably there will be another required set T of (V_1, V_2) patterns for each voltage level V. This has been shown by Gibiluka

Algorithm 4.3 – Bundled-Data delay test under voltage scaling with fixed delay for a 2-phase circuit.

```
1:  $delay_{codeword} \leftarrow \mathcal{I}$                ▷ $\mathcal{I}$ is the codeword supposed to produce the necessary delay
2:  for all $V$ voltages do                     ▷ $V$ is the number of operation voltages
3:      for all $P$ patterns do                 ▷ $P$ is the number of patterns for synchronous delay fault test
4:          for $i = 0; i \leq 1; i++$ do
5:              $delay \leftarrow$ ‘$i$‘                               ▷ Logic value
6:              Run delay fault test pattern ($V_1$, $V_2$)      ▷ LOS, LOC or Enhanced
7:              Collect results
8:          end for
9:      end for
10: end for
```

in [Gib16], who demonstrated the possibility that at distinct supply voltages the critical path of a circuit can differ among stages. As long as the number of patterns (P) remains constant, the analysis proposed here remains valid. Thus, to guarantee correct operation for a BD circuit operating under voltage scaling, it is required to multiply the number of tests by the number V of voltages under evaluation, resulting in the total test time described by Equation 4.3.

$$T_{total} = 2 \times V \times P \times T_{onepattern} \tag{4.3}$$

Nonetheless, in more recent technologies it is necessary to guarantee the covering of tests for small delay defects (SDDs). Using at-speed (AS) tests it is only possible to improve covering by increasing the number of patterns, which incurs in the increase of test time. One alternative is to employ faster-than-at-speed (FTAS) tests.

One of the fundamental advantages of using BD circuits consists in reduced switching activity compared to a synchronous counterpart. Also, due to the randomness of handshake protocols and by using DEs, BD circuits lead to less IR-Drop [CLM+16]. Thus, the problems arising from these parasitic effects occurring in synchronous circuits are reduced under the application of FTAS tests [TPC11].

Even if a good covering of SDDs is achieved, the problem to apply FTAS tests in BD circuits relies in the definition of the DE delays. Reducing the delay of an already unknown source by the uncertainties of the manufacturing process can compromise covering and yield in these cases [XZVH09]. The alternative of employing resilient BD circuits appears as a way to help bounding the uncertainties and improving the SDD covering predictability.

4.2 Blade Delay Test Analysis

As a resilient alternative to BD design, the Blade template allows circuits to operate with delay values smaller than the originally designed ones. By using an *error detection logic*

(EDL), stages can advance data propagation to a next CL while simultaneously evaluating if the result is not further modified during the current evaluation period. This characteristic can provide processing times that are smaller for faster-than-at-speed (FTAS) tests.

Despite naming it as *"error"*, the context of externalizing a defect does not fully apply to that detection mechanism. By identifying forced delay faults, but not propagating them, the term *error* is not the most appropriate here. However, because of the lack of a term that best defines a defect identified and corrected internally, this work chooses to use the term closest to the original definition.

On the one hand, if Blade allows applying FTAS tests, on the other hand, the mandatory use of two DEs by the template can cause a considerable increase on the number of test cases. This is due to the relative independence of the two DEs in each stage as well as to the ideal performance of Blade during its use.

4.2.1 Combinational Logic vs DE Delay

The resiliency mechanism based on the use of two DEs (δ and Δ) allows Blade circuits to operate with periods smaller than those expected for the worst case of the CL block. Nonetheless, the $\delta + \Delta$ period must be designed to guarantee that it covers the CL delay exactly in the worst case, just like in conventional BD circuits.

Just like in conventional BD circuits, the CL and controller test execution in the Blade template can detect defects in its DEs. However, there is a fundamental distinction between conventional BD circuits. Since Blade has both speculative delay (δ) and conventional delay ($\delta + \Delta$), it can be utilized to simultaneously execute FTAS and AS tests.

If, on one side Blade provides a range of benefits, on the other side some of Blade characteristics can add complexity. According to studies such as those described in [HHC+15, HMH+15], the ideal error rate must remain between 20% and 30% for Blade to reach its maximum performance. Wherefore, to make sure this characteristic is achieved, it is necessary to add a specific set of applications that help finding out if the ideal error rate is obtained. However, the need to produce an error rate around 20$\sim$30% to reach ideal performance requires the definition of a resiliency window Δ, which can end up by not being the ideal one for the FTAS test.

Ring oscillators strategically inserted in silicon can give information about process variations that during fabrication affect the final circuit functionality. With these, the codewords that produce the δ and Δ delays are estimated. From this discussion derives Algorithm 4.4, which is adapted from the BD test Algorithm 4.2 to apply FTAS tests. It is important to highlight that this case does not allow changing either δ or Δ values, requiring these to be fixed. Note that the original number of patterns remains the same but, thanks to the opera-

Algorithm 4.4 – Blade delay test with fixed $\delta + \Delta$.

```
1: $\delta_{codeword} \leftarrow \mathcal{I}$    ▷ $\mathcal{I}$ is the codeword supposed to produce the necessary $\delta$ delay for best performance
2: $\Delta_{codeword} \leftarrow \mathcal{J}$    ▷ $\mathcal{J}$ is the codeword supposed to produce the necessary $\Delta$ delay for best performance
3: for all $P$ patterns do    ▷ $P$ is the number of patterns for synchronous delay fault test
4:     for $i = 0; i \leq 1; i++$ do
5:         for $j = 0; j \leq 1; j++$ do
6:             $\delta \leftarrow$ ‘$i$‘    ▷ Logic value
7:             $\Delta \leftarrow$ ‘$j$‘    ▷ Logic value
8:             Run delay fault test pattern ($V_1$, $V_2$)    ▷ LOS, LOC or Enhanced
9:             Collect results (including error rate value)
10:        end for
11:    end for
12: end for
13: for all $N$ applications do
14:     Load memory
15:     Run app
16:     Collect results (including error rate value)
17: end for
```

tion below the original delay enabled by δ (FTAS), it is possible to obtain larger covering than that implied in Algorithm 4.2, for the case of SDDs. Add to this the capacity to simultaneously produce AS tests, as enabled by $\delta + \Delta$. With this, more precise results can be obtained for yield improvement.

In Algorithm 4.4 P is the number of patterns for synchronous delay fault test (here noted for reference only) and N is the number of applications employed to detect the error percentage rate. It makes sense to point out that applications must execute for a time adequate to detect the error rate (T_{oneapp}) and not only for a couple of cycles such as in the execution of a single (V_1, V_2) pair. It is possible to observe that even reducing the number of tests, the addition of a second DE has as consequence that the test needs to be executed four times, totaling a number of execution which can be computed by Equation 4.4.

$$T_{total} = 4 \times P \times T_{onepattern} + N \times T_{oneapp} \tag{4.4}$$

Observing Algorithm 4.4, it is easy to observe that the use of two DEs causes a cascading of tests. This is due to both the doubling of the number of DEs and to the use of a 2-phase protocol in Blade.

If using δ and Δ allows the use of FTAS tests and enhances Blade performance, these characteristics can intrinsically reduce the yield. This occurs because it is not trivial to tune the DE periods to reach maximum performance and, more importantly, because these values can be incompatible with those required to reach the best covering for FTAS tests. To reach this best covering situation and yet enable achieving the best performance, it is necessary to employ a scheme for dynamically adjusting δ and Δ.

Algorithm 4.5 – Blade delay test with multiple $\delta + \Delta$ combinations.

```
1: for all $L$ inputs do                          ▷ $L$ is the number of inputs at LUT
2:     $\delta_{codeword} \leftarrow \mathcal{I}$         ▷ $\mathcal{I}$ is the codeword supposed to produce the necessary $\delta$ delay for best performance
3:     $\Delta_{codeword} \leftarrow \mathcal{J}$         ▷ $\mathcal{J}$ is the codeword supposed to produce the necessary $\Delta$ delay for best performance
4:     for all $P$ patterns do                    ▷ $P$ is the number of patterns for synchronous delay fault test
5:         for $i = 0; i \leq 1; i++$ do
6:             for $j = 0; j \leq 1; j++$ do
7:                 $\delta \leftarrow$ ‘$i$‘                ▷ Logic value
8:                 $\Delta \leftarrow$ ‘$j$‘                ▷ Logic value
9:                 Run delay fault test pattern ($V_1$, $V_2$)    ▷ LOS, LOC or Enhanced
10:                Collect results (including error rate value)
11:            end for
12:        end for
13:    end for
14: end for
```

To allow the dynamic adjustment of δ and Δ to reach the best performance and yet enable good results in FTAS tests requires a number of supplementary tests. Here, results obtained from in-chip ring oscillators can allow to compute the probability of the possible (δ, Δ) configurations. The scan chain can then store a look-up table (LUT) with predefined combinations of (δ, Δ). Executing the tests in this scheme imposes two demands: (i) independently verifying each (δ, Δ) (i.e. considering the four possible combinations of phases for each δ and Δ - 2-phase); (ii) account for the (δ, Δ) LUT input variations. This leads to a complexity increase relative to these inputs. From this discussion arises the proposal of Algorithm 4.5.

Algorithm 4.5 enables adjusting the circuit for FTAS tests independently of the error rate, i.e. the error rate test (the loop on N in Algorithm 4.4) can be suppressed. However, the number of tests is dependent on the LUT size. Thus, the test complexity increase is proportional to the number of (δ, Δ) pairs in the LUT.

From the structure of Algorithm 4.5 it is viable to compute the total time required by the test. Equation 4.5 summarizes how this computation can be achieved. Note that besides the dependence on the two DEs, the number of LUT entries L is a multiplier of the number of test cases to execute. It is worth to remember that the target here is a solution that guarantees covering values identical, or very similar to what can be achieved with synchronous designs. This is why tests are required for all combinations of employed DE values.

$$T_{total} = 4 \times L \times P \times T_{onepattern} \tag{4.5}$$

Algorithm 4.6 – Blade Delay Test under Voltage Scaling with fixed $\delta + \Delta$.

```
1: $\delta_{codeword} \leftarrow \mathcal{I}$    ▷ $\mathcal{I}$ is the codeword supposed to produce the necessary $\delta$ delay to best performance
2: $\Delta_{codeword} \leftarrow \mathcal{J}$    ▷ $\mathcal{J}$ is the codeword supposed to produce the necessary $\Delta$ delay to best performance
3: for all $V$ voltages do    ▷ $V$ is the number of operation voltages
4:     for all $P$ patterns do    ▷ $P$ is the number of patterns for synchronous delay fault test
5:         for $i = 0; i \leq 1; i++$ do
6:             for $j = 0; j \leq 1; j++$ do
7:                 $\delta \leftarrow$ '$i$'    ▷ Logic value
8:                 $\Delta \leftarrow$ '$j$'    ▷ Logic value
9:                 Run delay fault test pattern ($V_1$, $V_2$)    ▷ LOS, LOC or Enhanced
10:                Collect results (including error rate value)
11:            end for
12:        end for
13:    end for
14:    if $V$ == nominal then
15:        for all $N$ applications do
16:            Load memory
17:            Run app
18:            Collect results (including error rate value)
19:        end for
20:    end if
21: end for
```

4.2.2 Voltage Scaling Delay Test

The Blade template can also take advantage of voltage scaling techniques for power reduction. However, as in conventional BD design, the impact of VS can affect DEs and CL differently, even by correcting the *VSDR*. Therefore, to ensure correct circuit operation, the delay fault test needs to be performed at each voltage level.

Starting with Algorithm 4.4 discussed in Section 4.2.1, it is possible to suggest the procedure presented in Algorithm 4.6. Note here that the delay test repeats for each voltage level.

Similar to what happens in the delay test case with VS in conventional BD designs, the set P of delay test patterns (V_1, V_2) can change for each voltage level V. To facilitate the analysis, it is assumed here the number of tests for each voltage does not change. Since DEs are not dynamically adjustable in this version, the test to determine the error rate in N applications only needs to be executed at the nominal voltage.

As for the algorithmic complexity growth, observe the proportionality of the number of voltage levels (V) used in VS operation. Complexity can accordingly be computed by Equation 4.6.

$$T_{total} = 4 \times V \times P \times T_{onepattern} + N \times T_{oneapp} \tag{4.6}$$

Algorithm 4.7 – Blade Delay Test under Voltage Scaling with multiple $\delta + \Delta$.

```
1: for all $L$ inputs do                                  ▷ $L$ is the number of inputs at LUT
2:     $\delta_{codeword} \leftarrow \mathcal{I}$    ▷ $\mathcal{I}$ is the codeword supposed to produce the necessary $\delta$ delay to best performance
3:     $\Delta_{codeword} \leftarrow \mathcal{J}$    ▷ $\mathcal{J}$ is the codeword supposed to produce the necessary $\Delta$ delay to best performance
4:     for all $V$ voltages do                            ▷ $V$ is the number of operation voltages
5:         for all $P$ patterns do        ▷ $P$ is the number of patterns for synchronous delay fault test
6:             for $i = 0; i \leq 1; i++$ do
7:                 for $j = 0; j \leq 1; j++$ do
8:                     $\delta \leftarrow$ ‘$i$‘                                    ▷ Logic value
9:                     $\Delta \leftarrow$ ‘$j$‘                                    ▷ Logic value
10:                    Run delay fault test pattern ($V_1$, $V_2$)       ▷ LOS, LOC or Enhanced
11:                    Collect results (including error rate value)
12:                end for
13:            end for
14:        end for
15:    end for
16: end for
```

Still, in the LUT-based implementation presented in Algorithm 4.5 the increase in delay test complexity is even more striking. With the LUT addition, each combination of (δ, Δ) values needs to be considered. Thus, Algorithm 4.7 arises.

Following the same previously presented reasoning, at each voltage level considered, new patterns (V_1, V_2) can be used. To simplify the analysis, the test pattern set (with cardinality P) remains constant for each voltage. To this also adds the fact that the (δ, Δ) pair selection can be adjusted later, e.g. at runtime[1], to achieve the required Blade performance levels.

Analyzing Algorithm 4.7, it is possible to get the expression of its execution, using Equation 4.7.

$$T_{total} = 4 \times L \times V \times P \times T_{onepattern} \tag{4.7}$$

In Equation 4.7 L is the number of pairs (δ, Δ) stored in the LUT, V is the number of employed supply voltages and P is the number of reference delay test patterns (based in a synchronous test).

By comparing the test time increase of all proposed algorithms, it is possible to see the influence caused by DEs. It should be clear that each added DE increases $2\times$ the number of tests to the algorithm, resulting in a $2^{|DEset|}$ growth in complexity. This points to the need to reduce the number of test runs in Blade circuits. In Blade, there is an interdependence between the two DEs, which leads to motivation to develop a new delay element, where the DE architecture reflects this interdependence and, at the same time achieves the reduction of delay margins caused by process variability. This is the subject of Chapter 5, the main contribution of this Thesis.

[1]This means that values of (δ, Δ) can be changed during circuit operation in the field.

5. THE COMPLEMENTARY DELAY ELEMENT

The test analysis presented in Chapter 4 motivates the development of a Delay Element (DE) targeting the ability to test the circuit. This Chapter presents an alternative to the current state of the art in DEs, which comprises the use of a series connection of two completely independent and separate delay elements.

Assume a delay element commonly used in asynchronous Bundled-Data (BD) templates. The use of this DE, illustrated in Figure 5.1(a) assumes as a design requirement that a request signal only reaches the controller after the Combinational Logic (CL) stage execution has accomplished its task. The Blade template, described in Section 2.2.3 and depicted in Figure 5.1(b), divides its DE into two parts, δ e Δ, deemed respectively to propagate and evaluate data produced by the CL. However, if the two DEs are independent and both have a variable delay, variations of process, voltage and/or temperature can influence each DE in distinct ways, which can in turn affect the expected global behavior.

To reduce process variation effects, it would be ideal that Blade DEs were in fact complementary parts of a single DE. Observing Figure 5.2(a) that shows a classic BD delay element, a solution would be to split this single DE into two DEs (δ and Δ), the sum of which produces the maximum stage delay. The resulting DEs would then share a same set of logic gates. Given that δ and Δ are expected to be variable delay DEs, another requirement is that the overall DE be sectionable at virtually any point, such that without changing the sum $\delta + \Delta$ component DEs have variable but directly related values, as expected.

The proposal of the Complementary-DE presented herein aims precisely to achieve the previously described behavior. From an overall delay element, the Complementary-DE derives two delay elements and still provide compensation for delay margin reduction. In addition, the new DE is more robust against process variations when compared to a classical MUX-DE alternative.

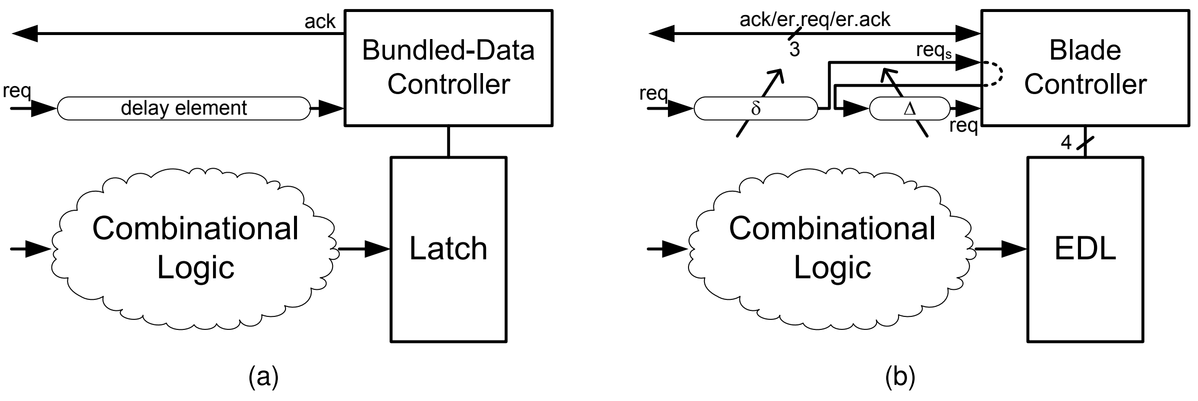


Figure 5.1 – Delay elements in distinct BD templates: (a) Bundled-data; (b) Blade.

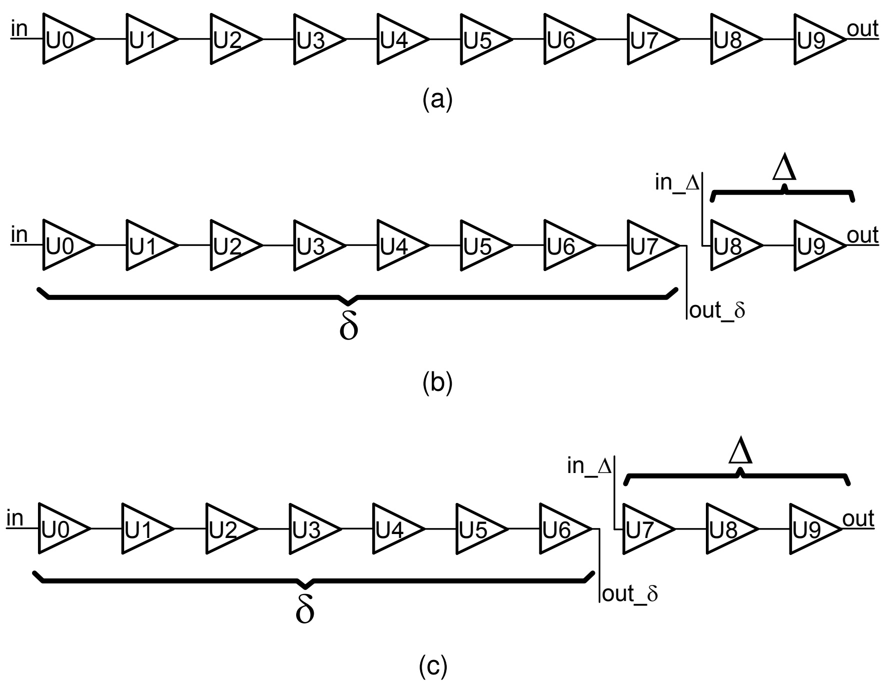


Figure 5.2 – Examples of delay element in: (a) Bundled-data; (b) Blade (example 1); (c) Blade (example 2).

The next Section presents the characteristics of the design problem and the proposed structure to implement Complementary-DE concept. Section 5.2 shows the design flow developed for the Complementary-DE Intellectual Property (IP) generation. The Section goes down to layout details of the IP, employing an example technology. Although not shown all results described here were fully reproduced in more than one technology[1]. Next, Section 5.3 presents and discusses simulations comparing the Complementary-DE with the classical MUX-DE approach, corroborating the advantages of employing the new DE. The simulations are conducted in all three technologies already explored in Chapter 3, i.e. two bulk technologies (TSMC 180nm CMOS and STM 65nm CMOS) and one FDSOI technology (STM 28nm FDSOI). Finally, Section 5.4 presents an alternative application of the new DE, which is its use for clock generation in circuits based on Razor template [EKD+03, DTP09, FFKP13] from the proposed DE.

5.1 The Complementary Delay Element

Due to the Blade template characteristics (described in Section 2.2.3), it requires the design of two distinct, adjustable delay elements at each logic stage. However, the sum of these DEs must guarantee the correct CL execution in the respective stage. One of

[1]Unfortunately, Non-Disclosure Agreement (NDA) constraints do not allow disclosing here all possible details and used technologies.

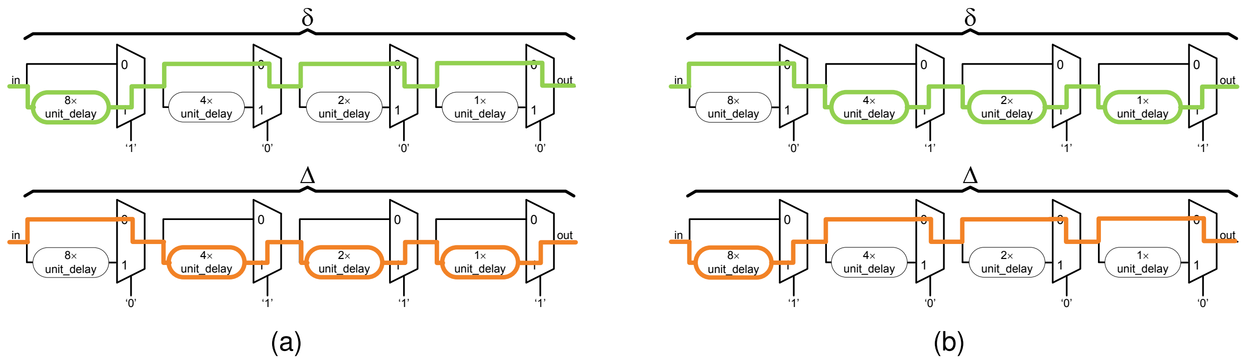


Figure 5.3 – Example of Blade DEs consecutive paths: (a) δ = "1000" and Δ = "0111"; (b) δ = "0111" and Δ = "1000".

the most common delay elements that provide these features is the MUX-DE. Figure 5.3(a) presents two possible settings for implementing δ and Δ with mux-DEs.

As Figure 5.3(a) shows, a possible configuration for δ operation comprises the selection value "1000" and, for Δ, "0111". If the error rate achieved with this configuration is lower than expected, a reconfiguration of the DEs can take place, said δ can be reconfigured with "0111" and Δ can be changed to "1000", as Figure 5.3(b) shows. In this example, it is easy to identify the path changes associated to the reconfiguration process, resulting in DE path changes of 100% between the two configurations. This path variations may cause unexpected behavior due to process variability.

Despite the fact that the Blade template operates by speculating a reduced delay to a specific stage (through δ), it still requires the amount $\delta + \Delta$ available, to make CL able to correctly receive and process data in the worst case. This implies that the total DE margin is still relevant and reducing it is a design requirement.

In addition, the use of two DEs significantly increases the test delay time, as Section 4.2 discussed. The uncertainties in delay margins in the DEs' delay sum require the verification of all possible delay combinations of δ and Δ.

The Complementary-DE is a single delay element that allows the reordering of its inner interconnections without drastically changing the originally interconnected cells. For this, a new DE structure is proposed and developed. This structure allows the signal to pass through the conventional path from the primary input to the primary output (*in*$\rightarrow$*out*) as shown for single module of the DE, depicted in Figure 5.4(a). This module allows sectioning the primary input signal and redirect it to a secondary output (*in*$\rightarrow$*out*') at the same time that it directs another input signal to the primary output (*in*'$\rightarrow$*out*), as Figure 5.4(b) shows. Accordingly, this basic module is called the Pass-Section (PS) DE, which describes its operation. The details of this new DE module are explored in Section 5.1.1.

Based on the MUX-DE architecture and using the Pass-Section DE introduced above, the Complementary-DE, by sectioning a delay element at one of several intermediate

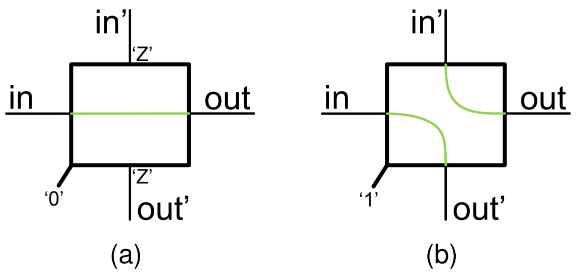


Figure 5.4 – The functionality of the Pass-Section DE module in its two possible configurations: (a) Pass and (b) Section.

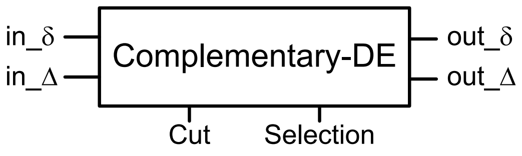


Figure 5.5 – The Complementary-DE external interface.

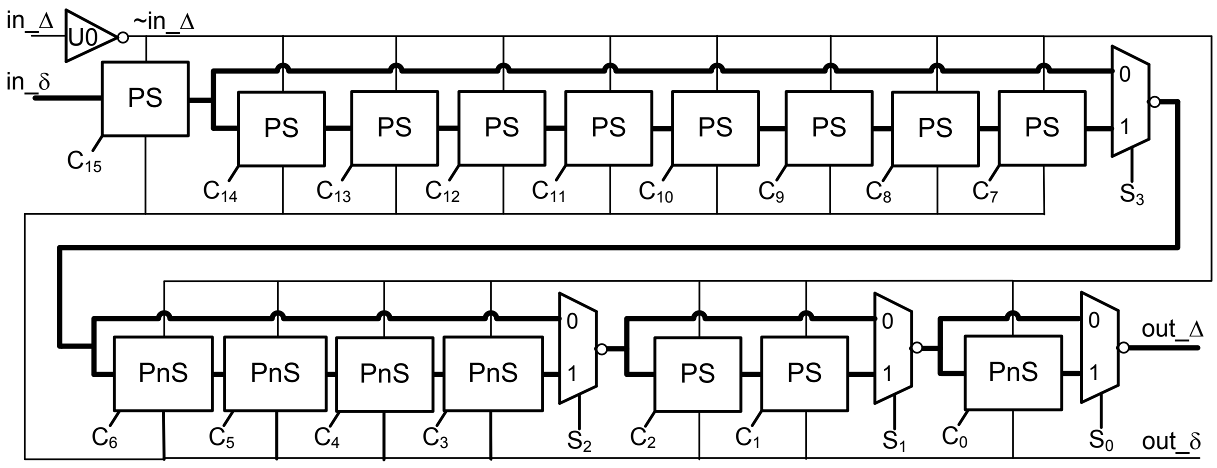
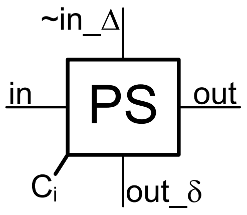


Figure 5.6 – The Complementary-DE structure, using a mux-DE implementation approach.

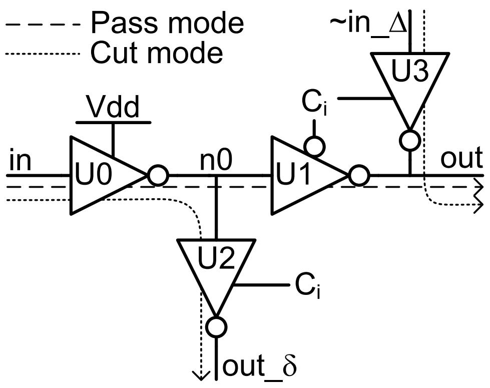
points creates two related DEs. Figure 5.5 displays the Complementary-DE top interface. Its internal interconnection, composed by Pass-Section Delay Elements (PS-DE), is illustrated in Figure 5.6. Note that this is not the only possible implementation of a Complementary-DE. More options are explored in following Sections.

5.1.1 The Pass-Section Delay Element

To enable a delay element section, the basic Pass-Section (PS) DE module is proposed here. As Figure 5.7(a) shows, the Pass-Section DE has two entries (*in* and $\sim$*in*_Δ), two outputs (*out* and *out*_δ) and a control signal (C_i), responsible for setting the overall DE configuration. To enable hardware reconfiguration and redirection of connections, the Pass-Section DE employs tristate elements, cells commonly found in cell libraries of vari-



(a)



(b)

Figure 5.7 – The Pass-Section DE: (a) external interface; (b) schematic. In the Pass mode the element connects *in* to *out* (works like a buffer), while in Cut mode it derives *in* to *out*_δ while connects $\sim$*in*_Δ to *out*

ous technologies. Figure 5.7(b) details the Pass-Section DE external interface and internal structure.

The control signal C_i is responsible for switching between *Pass mode* and *Cut mode*. In the *Pass mode*, when C_i has logical value '0', the input *in* is directed to the output *out* passing through tristates $U0$ and $U1$. In this configuration, the Pass-Section DE operates as a conventional delay element, in other words, part of δ or Δ.

If, on the other hand, the logical value '1' is applied to C_i, the Pass-Section DE enters in the *Cut mode*. Here, the Pass-Section DE generates the final part of δ DE (its output), connecting *in* to *out*_δ through $U0$ and $U2$. At the same time, $\sim$*in*_Δ generates the starting part of Δ, directing $\sim$*in*_Δ to *out* through $U3$. In the case, $U1$ isolates the nodes $n0$ and *out*, opening the DE.

There is a variant of the Pass-Section DE to achieve phase inversion as Figure 5.8(a) describes: the Pass-nSection (PnS) DE. Inverting multiplexers induce a phase inversion in the delay signal to the next stage. To derive the δ signal in phase inversion, it is necessary to restore its original phase. For this, the tristate $U0$ is repositioned to the output node, resulting in the configuration of Figure 5.8(b). Note that the control signal C_i operates in the same way as the original Pass-Section DE.

5.1.2 The Complementary-DE Process Variations Compensation

The compensation of circuit process variations is a mechanism to reduce margins (in time, power, etc.) that must be applied to ensure correct circuit operation even in the worst operating conditions admissible by the design. As discussed earlier, in Section 3.2, process variations can be compensated for in post-silicon steps. In the case of BD circuits,

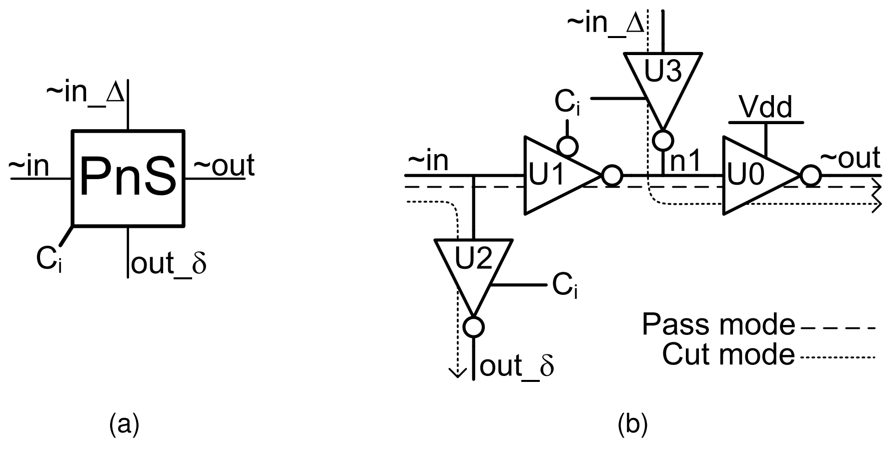


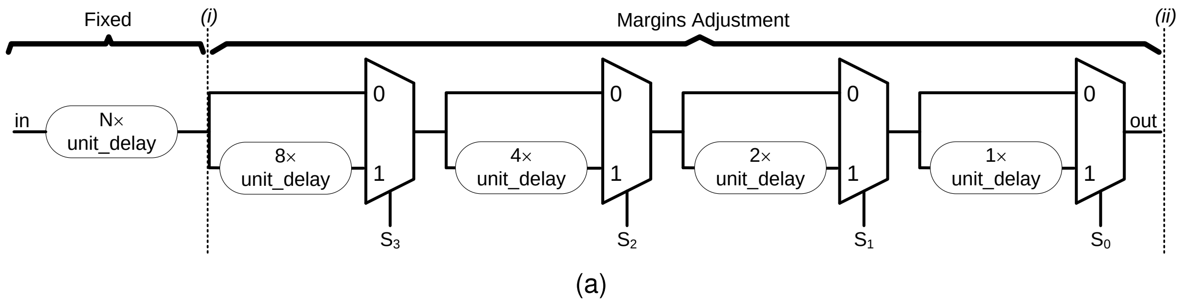
Figure 5.8 – The Pass-nSection DE: (a) external interface; (b) schematic.

a compensation mechanism is used to adjust the existing delay margin to increase circuit performance.

When analyzing the effect of PVT variations in the delay of a conventional 2-Phase BD circuit, two time constraints can be set for the CL conclusion: minimum and maximum delay. These delay values vary according to the corners selected by the designer. To ensure proper operation, the DE must have a delay identical to or a little higher than the CL delay. Variations due to the DE itself are added at this point, which increases the original delay margin, to ensure that the desired delay is obtained even in its best case operation. Excessive margins cause circuit performance degradation, reducing the BD efficiency.

A typical delay element for 2-phase BD circuits can be divided into two parts: one with fixed delay and an adjustable delay. As Figure 5.9(a) exposes, the fixed DE comprises the CL best case (minimum delay) and the worst case DE operation (maximum delay). The adjustable DE part is placed in series with the fixed one and provides delay values between points (i) (the minimum CL delay), and (ii) (the maximum DE delay). Thus, after capturing the process variability in the post-silicon step (usually checking frequencies produced by internal ring oscillators [XZVH09]), the delay between (i) and (ii) is adjusted to obtain a satisfactory delay that minimizes margin.

The Blade template, on the other hand, comprises the resiliency window peculiarity. For this reason, the fixed delay point previously established for conventional BD templates such as (i), now presents a shorter time. Being also defined by the designer, this delay takes into account not only the physical corners, but also the statistic application behavior that will be executed. In this way, the fixed delay (i‘) is established in Figure 5.10(a). The resiliency window adjustment that determines the Δ delay is located between points (i‘) and (ii). Finally, the adjustment of the total DE delay (the sum of delays $\delta + \Delta$) is defined between points (i) and (ii), exactly as in conventional BD design.



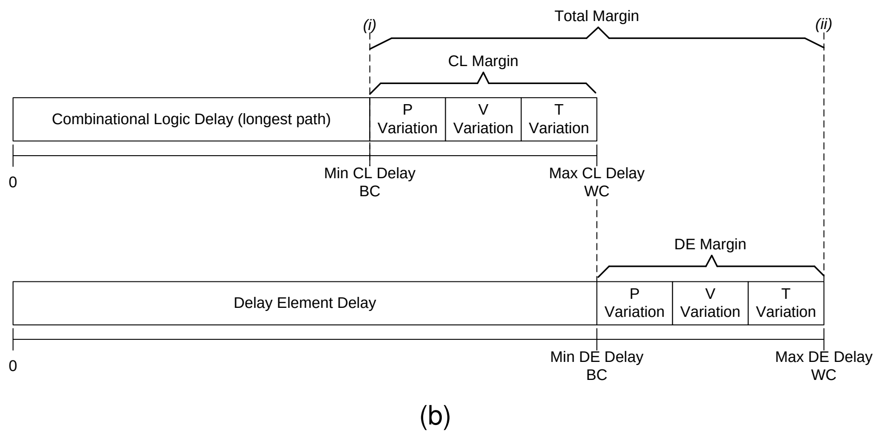
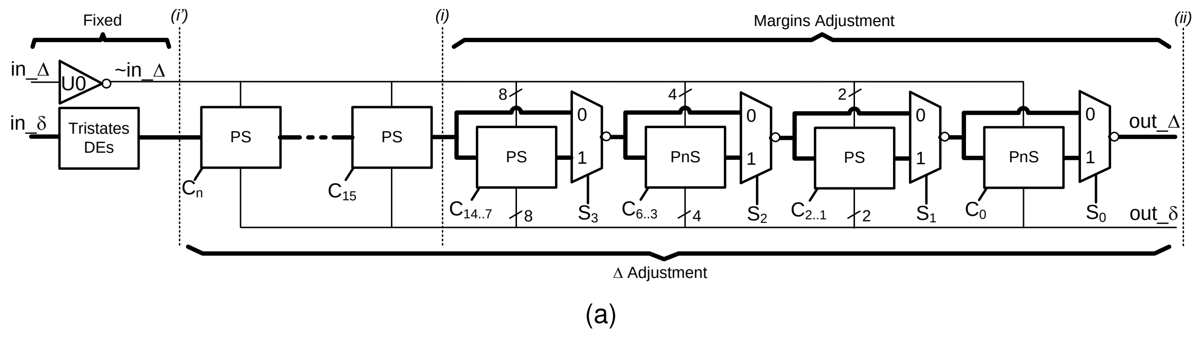


Figure 5.9 – BD delay adjustments: (a) Example of a conventional BD DE; (b) BD timing diagrams (BC = Best Case, WC = Worst Case).



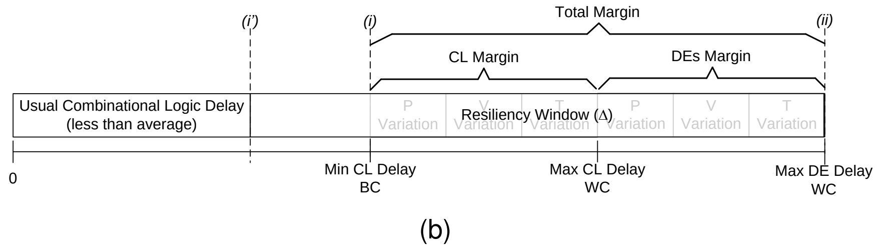


Figure 5.10 – Blade delays: (a) Example of Complementary-DE adjustments; (b) Blade timing diagram (BC=Best Case, WC=Worst Case).

It is possible to note the selection signals (*S*) are intended to adjust the sum of the delay elements δ and Δ. This signal is set in a post-silicon step, at test time. Once adjusted, resiliency window size changes can be performed through the control signals (*C*) dynamically, at execution time. This mechanism allows altering the resiliency window with minimum changes in the total delay element, which guarantees correct Blade operation.

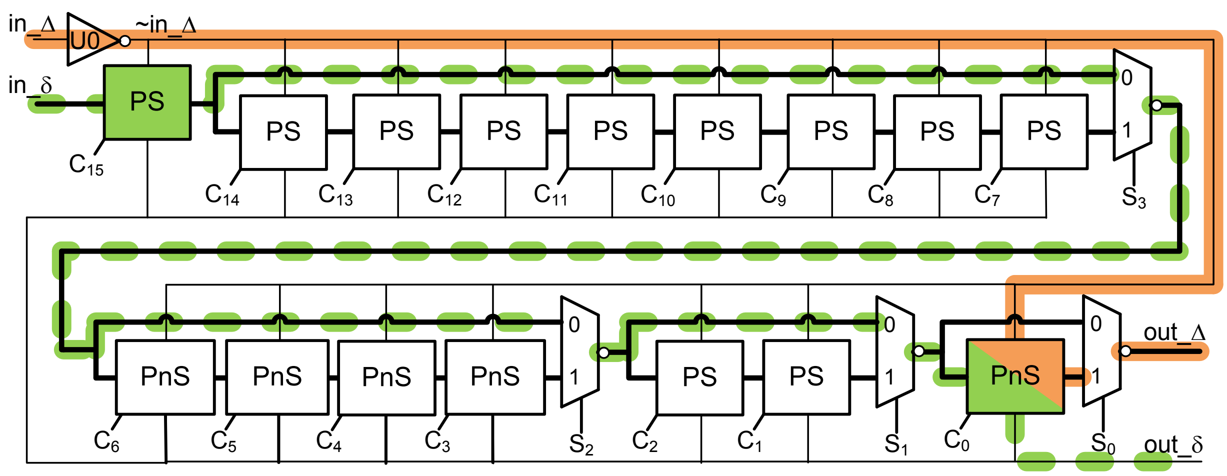


Figure 5.11 – Complementary-DE configuration example (*control* = ”0000000000000001” and *selection* = ”0001”.

Observing Figure 5.10(a), sectioning the Complementary-DE takes place at exactly one position. Thus, the encoding of the control signal (C) can benefit from the use of *one-hot* codes. Selection (S) to adjust for process variations can benefit from using conventional binary coding. It is also important to inhibit the action of the control signals (C) in the stages disabled by the selection signal S. Because they are off the main path, these points should be eliminated by the Complementary-DE controller. Note also that despite the use of a one-hot code, C cannot be changed without previously pausing the circuit. Otherwise, glitches can propagate to Blade controllers and may cause unwanted transitions or even deadlock.

To illustrate the Complementary-DE operation, a configuration example appears in Figure 5.11. First, PVT setting takes place by fixing the value of the *selection* signal ($S_3..S_0$). In the example, it receives the second lowest possible value (*selection* = ”0001”). Next, signal *control* ($C_{15}..C_0$) is set, to define the Complementary-DE sectioning point. In the example, this occurs in the PnS closest to output out_Δ (*control* = ”0000000000000001). Finally, the dashed (green) line shows the δ delay path and the continuous (orange) line shows the Δ delay path.

5.2 The Complementary-DE Design Flow

Due to the Complementary-DE design complexity and the need of disposing of a soft IP to integration this DE in circuits, it is justified the development of an specific design flow for the DE. Thus, a set of scripts is proposed here to promote such flow. Although the flow assumes an architecture based on the use of standard cells, interconnections and gates instances require specific attention to reduce sources of interference such as wire delays and crosstalk, in order to reduce delay margins. This flow is also applicable to other delay elements, such as the MUX-DE alternative.

To achieve the above stated requirements, the flow is divided into three stages: logical synthesis based on a parameterizable Verilog code, physical synthesis and parasitics extraction. The logical synthesis relies on a parameterizable Verilog code. This code allows abstracting the Complementary-DE interconnection complexity to the point of just requiring selection of the technology library gates to employ and the specification of the number of stages constituting the DE instance to generate. In the physical synthesis, details are presented for the cell arrangements, for the signal distribution and for the layout IP generation necessary to allow the Complementary-DE inclusion in the Blade design flow. The parasitics extraction aims at increasing the measurements accuracy, to reflect the real physical implementation.

5.2.1 The Parameterizable Complementary-DE

Currently, any digital circuit correctly described in structural Register-Transfer Level (RTL) of some Hardware Description Language (HDL) can be interpreted and transcribed to logic gates during logical synthesis. However, delay elements are not usually employed directly in synchronous circuits. Design flows like *ACDC* [GMC15] and Blade [HMH+15] environments elaborate the DE during the BD design, but none of these were capable of instantiating adjustable DEs so far.

Due to the characteristics of the Complementary-DE, there are interconnections not usually done, such as the tristate connections inside PS/PnS DEs, which prevent the correct interpretation of the description specified at a high abstraction level in some HDL. A parameterizable low level HDL is then proposed and developed to simulate and generate a Complementary-DE specific instance. The DE spec is finally produced as a structural Verilog description. The process requires four gates from the technology standard cell library and the specification of three other parameters which determine: the number of buffers, the PS/PnS modules and the multiplexers.

Because the employed gate cells (a tristate, two inverters and an inverter multiplexer) are technology dependent, these are selected by the designer and instantiated in the Verilog code (in the *Cells.v* file). There are two distinct inverters, because one is used to invert the control signal in the PS-DEs and PnS-DEs and the other is used to distribute the *in*_Δ signal. The latter is employed here to improve the distribution of the $\sim$*in*_Δ signal that could be interpreted as a clock tree because of its distribution across multiple nodes. Currently, the setting of this inverter size is manual, since part of the parasitic extraction takes place manually, which prevents its automation.

After selecting the four cells from the library, the designer employs a parameter to define the number of elements of each part of the Complementary-DE. Basically, there are three parameters that help in defining the number of elements: the *minimum delay*, the

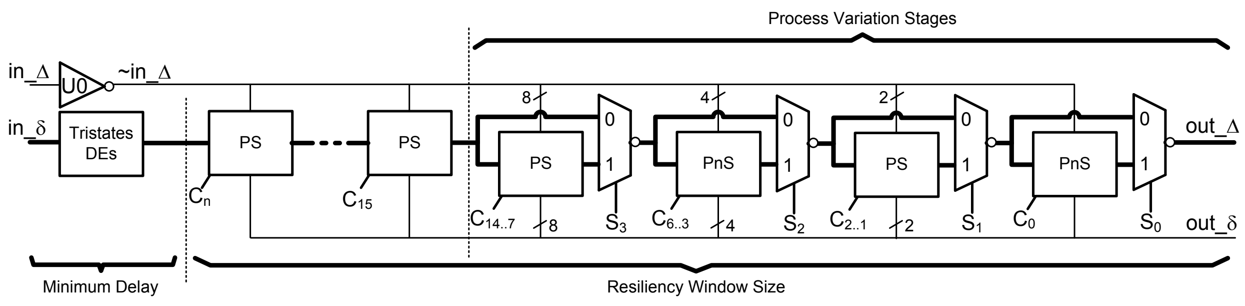


Figure 5.12 – Complementary-DE parameters for HDL input file.

resiliency window size and the *process variation stages*. Figure 5.12 shows in detail how each parameter impacts the design. The *minimum delay* defines the number of elements to produce the fixed delay, i.e. the point of minimum δ. The *resiliency window size* defines the number of PS/PnS DEs that can section the main delay element to generate δ and Δ. Finally, the *process variation stages* determines how many multiplexers will be instantiated to allow adjustment of process variability ($\delta + \Delta$).

Originally, the Verilog code is designed to work with pairs of *process variation stages*. This ensures a balance between the rise and fall propagation delays as presented in Chapter 3. However, if an odd number of stages is chosen, the code uses a permanently active tristate connected at the Complementary-DE output, to avoid the use of non-inverting multiplexers. The use of such a cell can accumulate different rise and fall delays, rather than compensating them.

Being described in Verilog, the Complementary-DE input file can be instantiated in any HDL circuit for simulation and validation, as long as there is access to the PDK behavioral Verilog. In this way, it is possible to identify possible Complementary-DE errors, either at its interface or of behavioral nature.

5.2.2 Synthesis, Extraction and Instantiation of Compllementary-DEs

Starting with the parameterizable Verilog described in the previous Section, it is possible to perform logical synthesis and then physical synthesis of the Complementary-DE IP. Scripts are available for Cadence CAD tools, some compatible with older versions of the framework (RTL Compiler and Encounter) and some compatible with more recent versions (Genus and Innovus). These last two tools are available in beta versions from the Cadence University Program. Therefore, future changes to full compatibility with the latter may be required.

The Complementary-DE uses its design flow to produce an IP independent of the rest of the circuit in which it will be instantiated. The purpose of this choice is to reduce the distance between cells and to isolate the DE to avoid interference from nearby circuits. In this

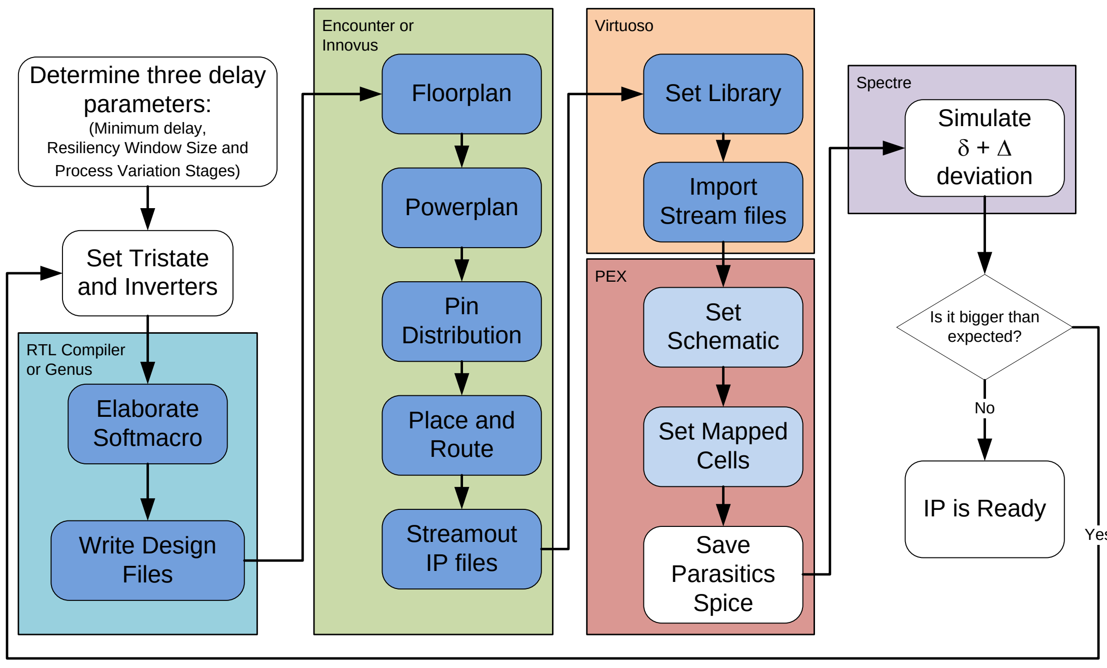


Figure 5.13 – The design flow for Complementary-DE IP production.

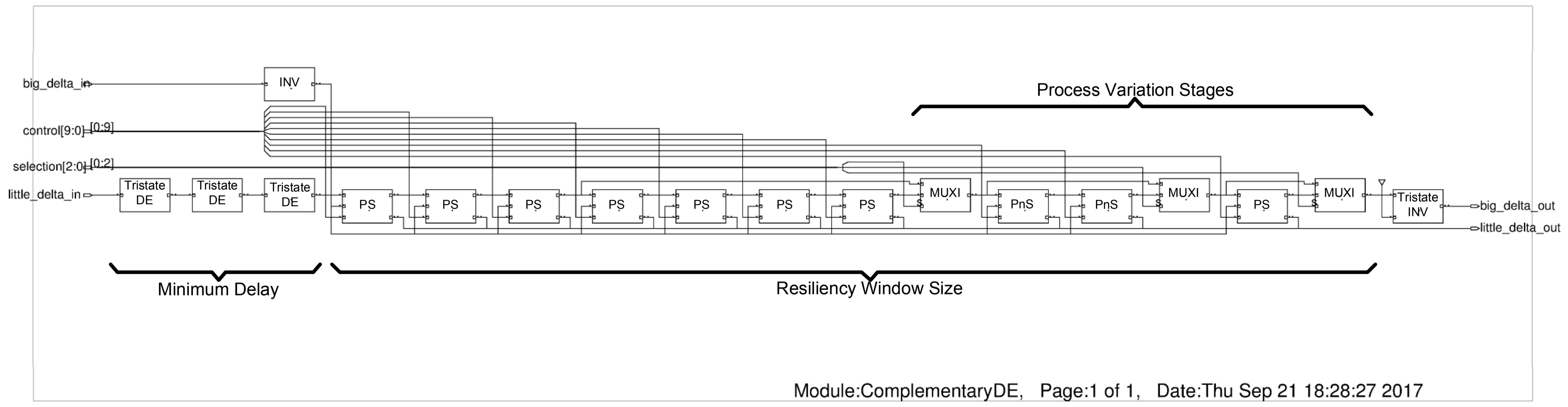
way, the impact of wire delay is reduced and the supply voltage variation is less significant. In addition, the flow allows to instantiate the Complementary-DE IP closer to the combinational circuit that employs it, ensuring that voltage variations are kept somehow proportional in the DE and in the combinational logic.

Figure 5.13 displays the design flow. The tool choices used at each step are highlighted. The shaded blue or light blue items are respectively fully or partially scripted. White steps still require designer manual intervention.

Since it is developed starting with a parameterizable Verilog, the Complementary-DE is only elaborated during the logical synthesis in RTL Compiler or Genus. This Verilog code has direct mapping commands to cells using the *elaborate* command. In this way, conventional steps of logical synthesis such as area or performance optimization must not be applied here. However, this step is essential both for obtaining the physical Verilog and for preparing the environment for physical synthesis in Encounter or Innovus.

Figure 5.14 shows an instance of synthesized IP, based on parameters *minimum delay* = 3, *resiliency window size* = 10 and *process variation stages* = 3. Due to the modularization process, the original symbols (multiplexer, tristate, inverter) are replaced by boxes. Note that by using 3 multiplexer stages (an odd number), a permanently active inverter tristate (Tristate INV) is added after the output multiplexer.

After calling Genus (or RTL Compiler) to write the design files, the script launches the Innovus tool (or Encounter) to begin the Complementary-DE physical synthesis. The physical synthesis scripts have been automated to allow the IP elaboration without any user



Cadence Schematics, Copyright 1997-2006

Figure 5.14 – Illustration of a Complementary-DE synthesized in Cadence Genus using the proposed parameterizable Verilog.

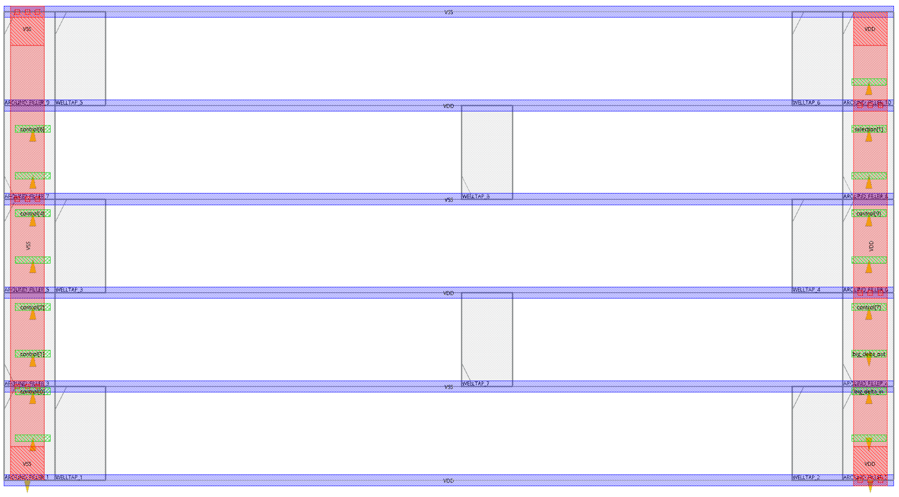


Figure 5.15 – An example of power plan and pin distribution for the Complementary-DE IP.

intervention. However, the designer can still act on it, if he so desires. This step was divided into Floorplan, Powerplan, Pin Distribution, Place and Route, Fillers and Streamout IP files.

Floorplan initializes the environment, loading the libraries and files generated by the Genus in the logical synthesis step. The area density selected for the Complementary-DE is 80% and the ratio for Complementary-DE dimensions is 0.6. The occupation was restricted to this value, due to the need for instantiating well cells (known as *Welltaps*) in more recent technologies.

Powerplan distributes the power needed to bind the Complementary-DE cells. To avoid the need for stripes inside the IP, it also includes stripes on the circuit edge. This is done to reduce the voltage variation impact between power rows. In this step Welltap cells are also inserted at appropriate intervals. The script strategically adds fillers below the power stripes. This eliminates the cell placement in border regions and also provides an appropriate location for the inclusion of IP interface pins.

Pin Distribution is performed by a script that automates the placement of all IP interface pins on the sides of the layout. Due to the peculiarity of being directly employed as an IP within Blade circuits, this is sought to prevent possible placement problems at the Complementary-DE boundaries, including in the CL routing. To do this, the script places the distributed pins on layout sides, properly spaced and slightly moved into the IP as can be seen in Figure 5.15. As Metal 3 is usually employed for horizontal routing, this metal layer is also chosen for the side pins. It is important to emphasize that the pin positions are determined independently of the number of elements chosen by the designer, since their distribution is dynamically defined.

The Place and Route step positions cells according to defined constraints. The option was for activating every possible optimization without changing the cells already in use. Concerning wire routing, the script restricts metals above metal 3, as Figure 5.16 makes clear. No congestion and no unsolved routing were identified with these limitations

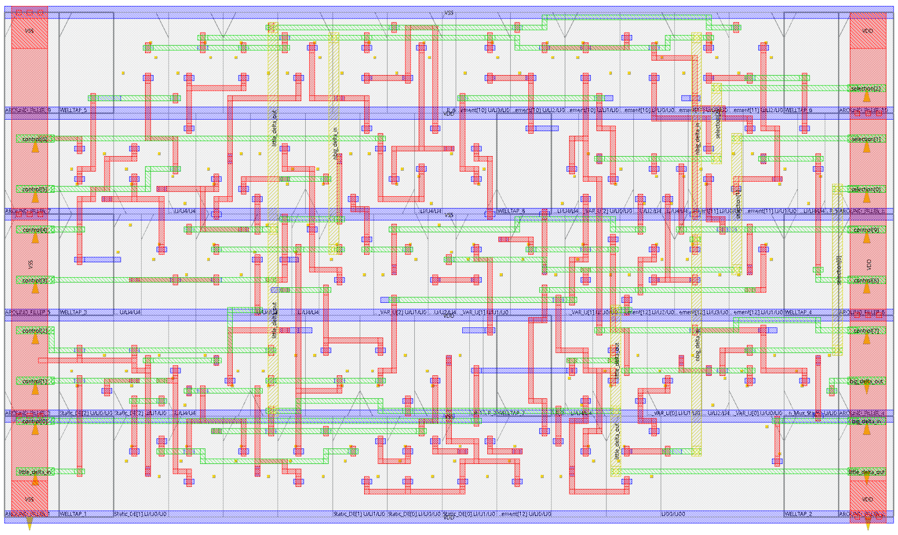


Figure 5.16 – An example of the final layout for the Complementary-DE IP, with a limited number of layout layers displayed.

applied in the used technologies, which contributes to wire delay reduction. After the Place and Route step, the design is completed by applying fillers in the layout empty spaces.

Finally, the script writes the IP Complementary-DE output files. Since the intention is to avoid future interference of circuits that instantiate the IP, within the IP definition the routing of wires inside the circuit is limited. In this way, it is possible to route metals over the Complementary-DE just above Metal 4[2]. During the streamout step, the script produces two physical Verilog files, one with power pins and another without these pins, a Standard Delay Format (SDF) file, a Graphic Database System (GDS) file, an Open Artwork System Interchange Standard (OASIS) file, a Library Exchange File (LEF) file for library and abstract, and finally a Design Exchange Format (DEF) file. The SDF, GDS, LEF and DEF files can be used to instantiate the IP in some Blade circuit. However, to obtain more precise measurements and mainly to investigate the impact of process variations, it is necessary extract the IP parasitics. In this case, the PEX tool from Mentor Graphics is employed.

Although PEX allows the parasitic extraction without the library cells, it was not possible to make the tool operate without library integration. Thus, the script calls Cadence Virtuoso tool to integrate the designed IP with library cells. To facilitate this design inclusion, the script also configures and adds libraries, and imports the IP design from Innovus (or Encounter).

The integration with technology libraries can be a non-trivial task in some cases. The vast majority of libraries do not provide a full layout view of cells. It is common for foundries to only release access to the metal views, to avoid undue routing in these layers, but hiding other layers for intellectual property protection. It is important to mention the possibility of library files themselves come with changed layers or even without the layers

[2]Metal 4 is reserved for the purpose of shielding, as will be explored later in this Thesis

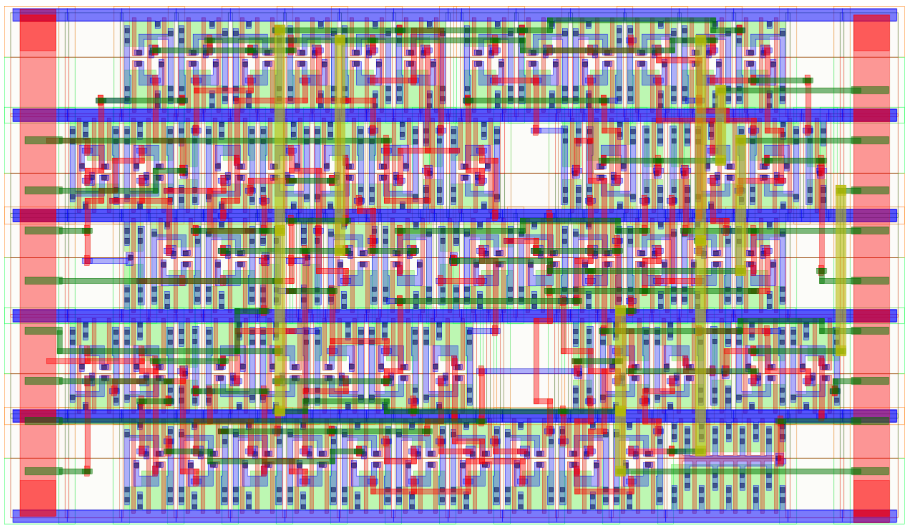


Figure 5.17 – An example Complementary-DE IP layout, as imported in Virtuoso.

essential for layout integration. An example is the abstract view in some libraries used to synthesize the Complementary-DE, which came with a pin layer different from what is defined in the technology layermap. This caused confusion and a long delay in the process of parasitic extraction. As the cell available layers are usually limited, Authors usually recommend converting the layout view to an abstract view to complete the integration and proceed to extraction. An example result of library adaptations and Complementary-DE IP integration can be seen in Figure 5.17.

Despite integration with the Virtuoso tool, PEX still needs two input files to be properly configured. The first file is the Complementary-DE schematic used for Layout Versus Schematic (LVS) check. A script-generated Spice file, obtained from the physical Verilog produced by Innovus (or Encounter), is then used. Without this step, PEX cannot extract the circuit internal nodes properly. It is important to highlight that there are some discrepancies between pin designations in the physical Verilog generation, which are script-handled. The second file is the list of library cells that will be skipped during extraction. This list maps cells identified in the layout to a name that will be used, instead of original names in the extracted circuit Spice file.

The PEX execution is the single step that has not been automated so far. Until now, no commands have been found that would allow accessing the Virtuoso database without using a graphical interface. Thus, to obtain the Complementary-DE IP Spice description, this step still requires the user interference through a graphical interface interaction. Fortunately, with all the changes made via scripts, the Spice files can be used to simulate the extracted circuit without further changes.

Finally, the generated Spice file is simulated to see if the impact in the rise/fall difference caused by $\sim\in\Delta$ signal distribution is acceptable. Currently, the inverter defined in the initial Verilog file that sets this optimization is done manually. Therefore, if the variation is

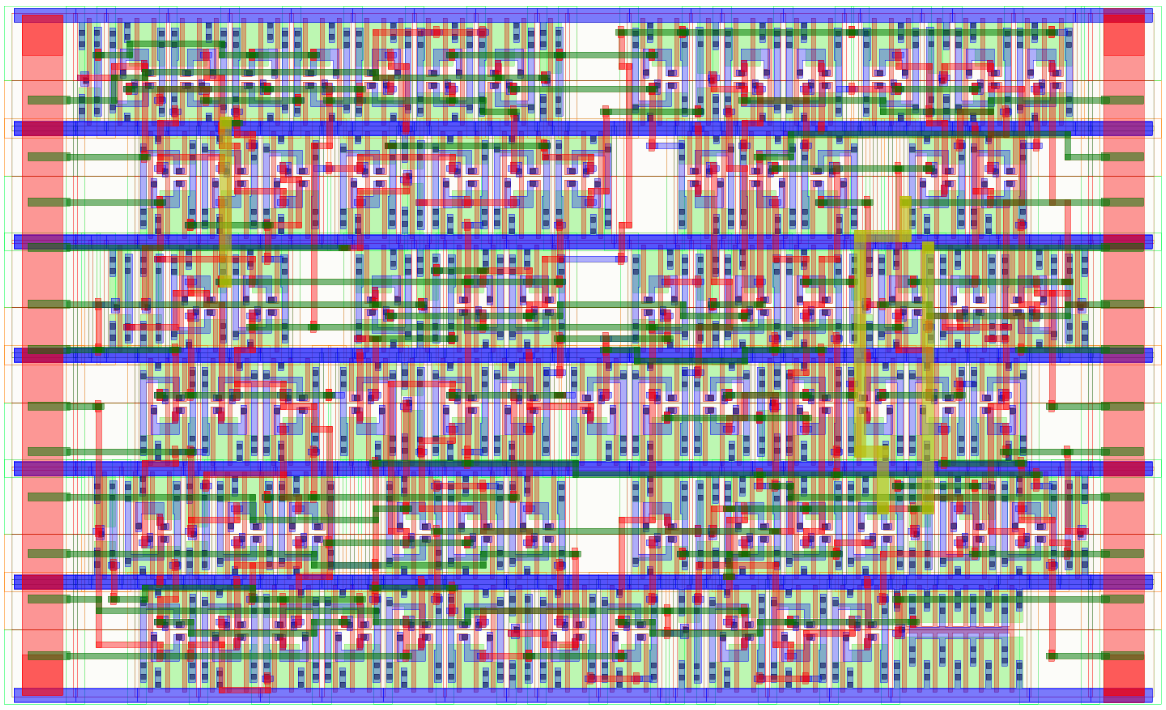
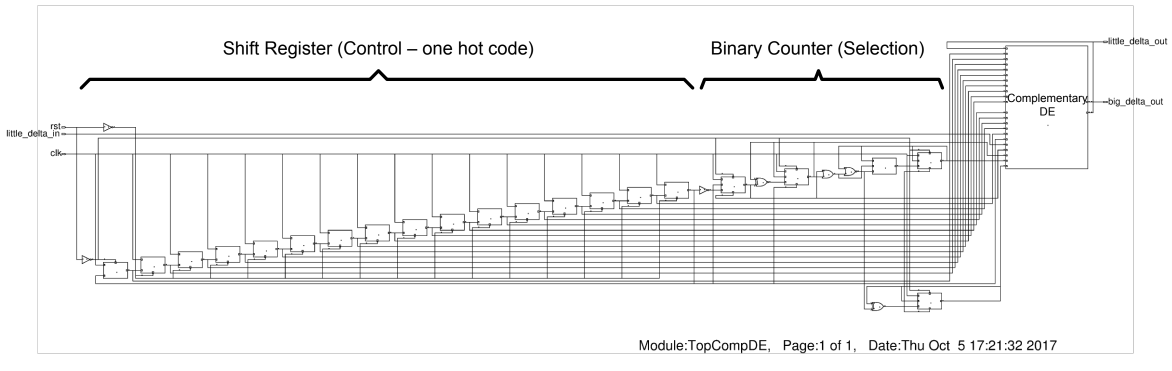


Figure 5.18 – Example final layout for a Complementary-DE IP with Minimum Delay = 0, Resiliency Window Size = 16, and Process Variation Stages = 4.

below that tolerated by the designer, a new inverter cell needs to be selected and the entire flow is re-executed.

The Author has also developed a small circuit tester, capable of operating with any Complementary-DE instance. Its purpose is to extract essential information to validate the design flow presented here and to evaluate the benefits of the new delay element. In addition to the power pins, this tester circuit has only five interface pins (*rst*, *clk*, *little_delta_in*, *little_delta_out* and *big_delta_out*). Consisting of a shift register and a binary counter, the tester sweeps all combinations of *control* with one-hot coding using the *clk* signal transition and, for each complete control value sweep, it increments the binary counter responsible for producing the *selection* input. Although some *control* settings do not operate properly (specific selections can inhibit PS or PnS elements from the main path like when *control* = "0000000010000000" in the example presented in Figure 5.11), they do not short-circuit or destroy the device. In these cases, the delay should be disregarded, since it produces an invalid combination.

For the validation addressed here, the Complementary-DE configuration is as follows: (i) without *minimum delay* (no Tristate buffers); (ii) *resiliency window size* of 16 PS/PnS DEs and (iii) *process variation stages* of 4 bits. Figure 5.18 displays the result of a Complementary-DE physical synthesis with these parameters. The test circuit instantiating the Complementary-DE in the logic synthesis can be seen in Figure 5.19. Figure 5.20 and Figure 5.21 respectively present the physical synthesis of the finished circuit in Innovus and the final circuit imported in Virtuoso.


Shift Register (Control – one hot code)
Binary Counter (Selection)
rst
little_delta_in
clk
Complementary DE
little_delta_out
big_delta_out
Module:TopCompDE, Page:1 of 1, Date:Thu Oct 5 17:21:32 2017

Cadence Schematics, Copyright 1997-2006

Figure 5.19 – A schematic view of a testbench for the Complementary-DE, with the results of the testbench logic synthesis, containing the test circuit and an instace of the Complementary-DE.

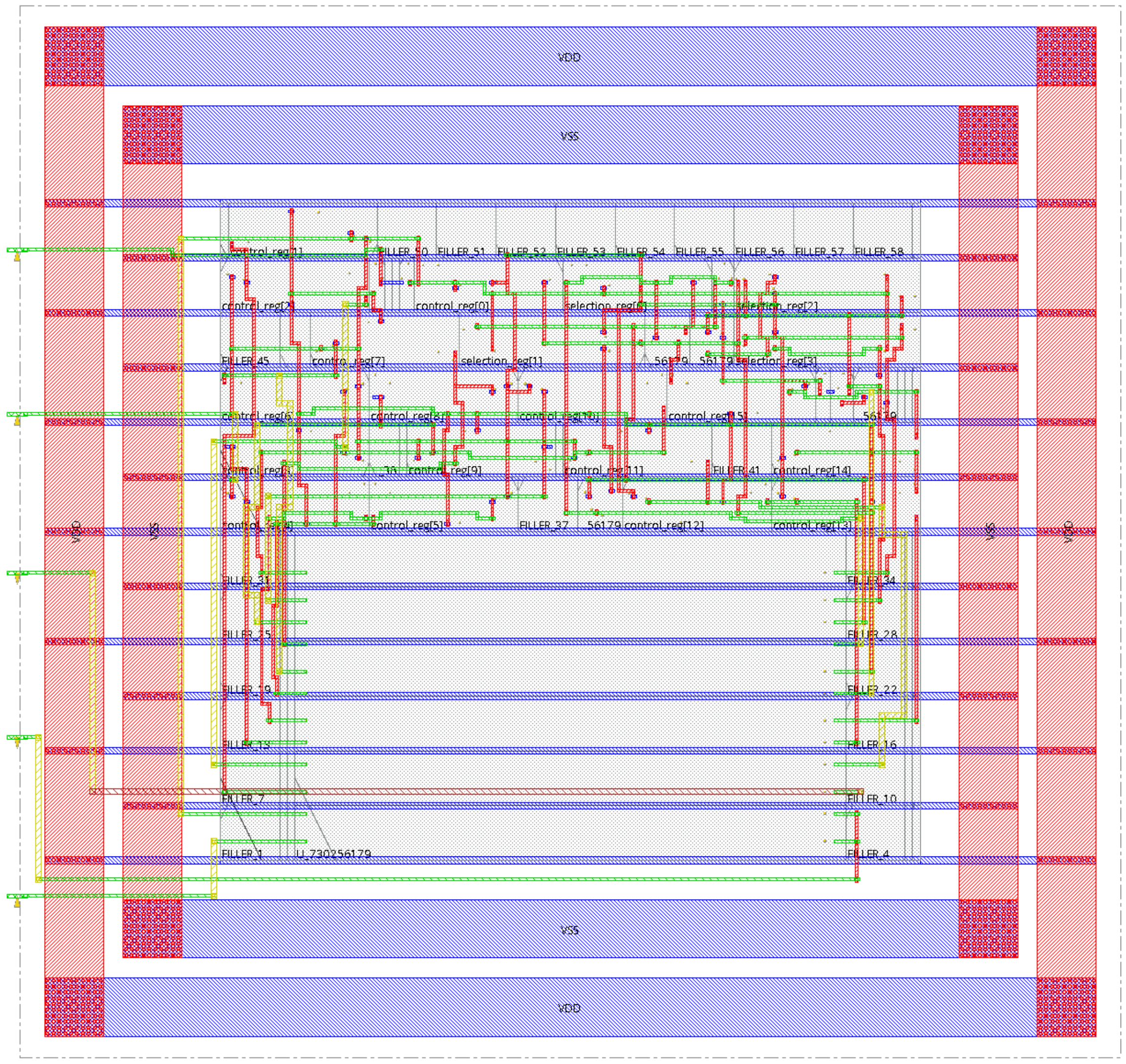


Figure 5.20 – Example result of the test circuit physical synthesis, with the Complementary-DE occupied area as an empty box at the bottom center of the layout.

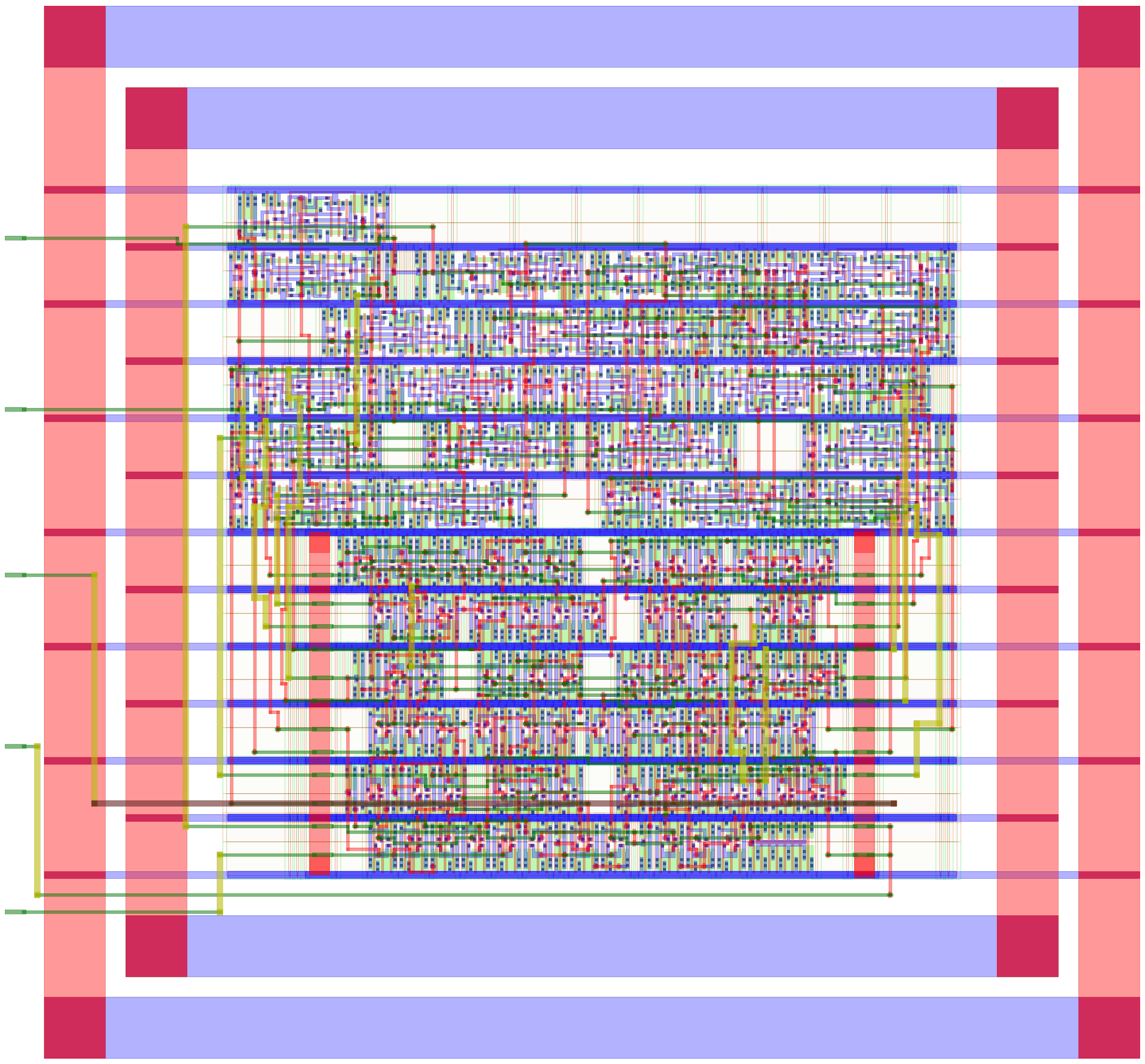


Figure 5.21 – A complete layout example of a testbench circuit instantiating a Complementary-DE IP.

5.3 Complementary-DE Simulation and Comparison with MUX-DE

This Section aims to present simulations that validate the concepts developed in previous two sections. Three CMOS technologies to which the Graduate Program in Computer Science at PUCRS has access were selected. The University also provides access to all the necessary tools for circuit elaboration and simulation through University Program agreements with Cadence and Mentor Graphics. The experiments put the focus on showing the advantages and disadvantages of the proposed DE in addition to comparing parasitic variations along technologies.

The three PDKs used are from the 180nm bulk CMOS technology of Taiwan Semiconductor Manufacturing Company (TSMC), from the 65nm bulk CMOS technology of STMicroelectronics (STM), and from the 28nm Fully Depleted Silicon On Insulator (FDSOI) also from STM. These PDKs are all commercial ones, i.e. based on actual technologies currently in use to fabricate commercial ICs. In this way, it is expected that the parasitic effects found in reality are reflected in an accurate way.

As for the development environment, it is important to highlight some of the used tools. For circuit front-end and back-end elaboration, Genus and Innovus (both from Cadence) are respectively used, following the design flow presented in Section 5.2. On the other hand, the integration with cell layouts and libraries is achieved using the Virtuoso environment, also from Cadence. The parasitic extraction is performed with the PEX tool from Mentor Graphics.

All circuit simulations employ the Cadence SPECTRE tool. Conventional delay and *VSDR* experiments were performed on a workstation with an Intel Xeon W3670, 3.2GHz processor, with 18GB of RAM. Monte Carlo simulations were performed in a mixed environment involving machines with Intel Xeon and Intel i7 processors of several models, each with at least 8GB of RAM.

5.3.1 The Complementary-DE vs MUX-DE Comparison

In order to identify the advantages and disadvantages of using the Complementary-DE it is compared here to the classical MUX-DE in terms of behavior and produced delay variations. To enable similar functionality of the two DEs provided by the Complementary-DE, two MUX-DEs are connected in series as required.

Both circuits were developed to generate 16 different intermediate delay settings (δ), keeping the sum of the two DEs (δ+Δ) constant. Due to timing constraints, the number of configurations was limited to 16 because it represents an expressive value of configurations without incurring in too long simulation times.

The Complementary-DE is configured with 4 Process Variation Stages, Resiliency Window Size of 16 elements and 0 Minimum Delay elements. Figure 5.22 shows the resulting circuit diagram. In itself, process variation stages could be eliminated, but the Author prefers to include them to make the experiments akin to actual applications. To make viable achieving 16 distinct configurations, *selection* (*S_i‘s*) is kept with a fixed value, "1111". Thus, all PS/PnS DEs can be used.

To produce the same 16 different delay settings for δ, two 4-multiplexer-stage MUX-DEs are instantiated. The resulting circuit diagram can be seen in Figure 5.23. The only difference between these two MUX-DEs is the *selection* input where the first one receives inverted values.

Both circuits were developed with the same parameters and extracted in the same way for the already described 180nm and 65nm technologies. This is essential to turn the comparison as fair as possible.

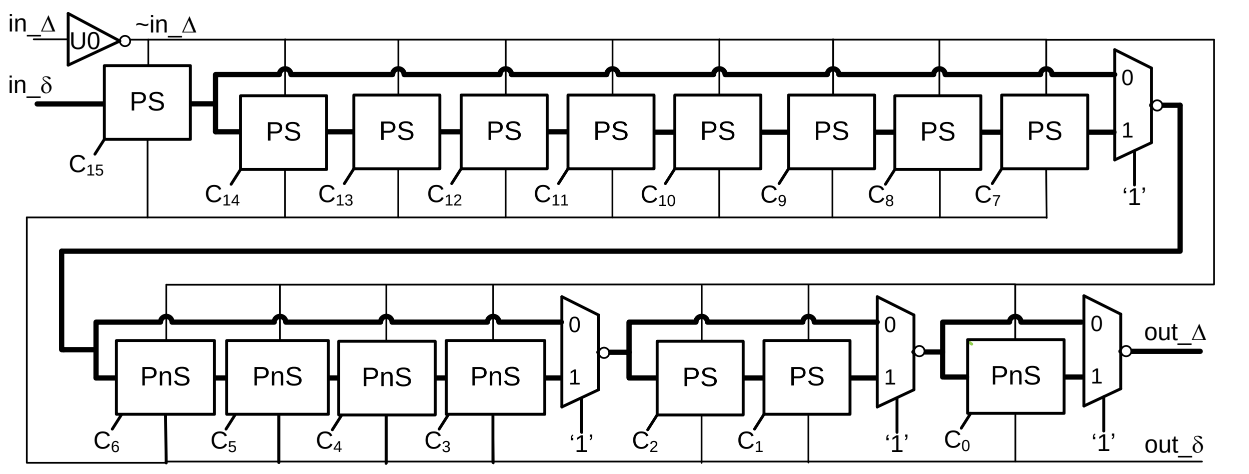


Figure 5.22 – The Complementary-DE configuration used for comparison simulations.

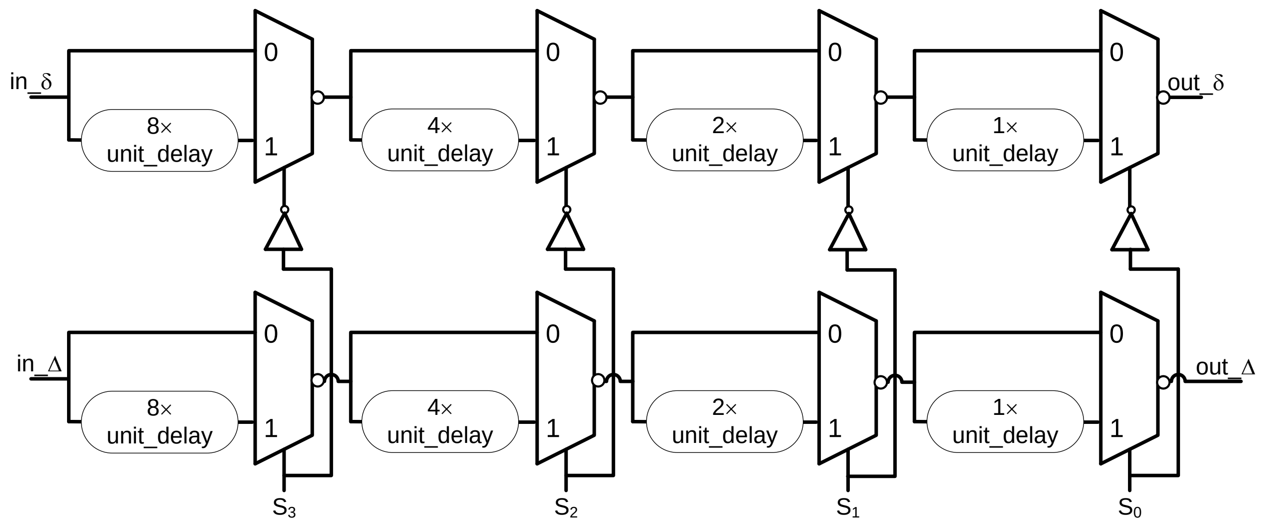


Figure 5.23 – The MUX-DE structure for comparison purposes.

Unfortunately, for the 28nm FDSOI technology it was not possible to elaborate the Complementary-DE due to the lack of tristate cells in the available library. Although the Complementary-DE reconfiguration mechanism is adaptable to other cells, such as transmission gates, this PDK does not provide any cell that produces high impedance output ('Z'). For this reason, the results presented for this technology do not have parasitic effects for cells or wires. To overcome the lack of tristates cell, one was sized following the minimum technology multiplexer. These cells use tristates internally for their construction.

5.3.2 Delay Margin Analysis

The comparison between the Complementary-DE and MUX-DE solutions focuses on three key points: generated delay (intermediate and total), impact of process variability

Table 5.1 – Comparison between MUX-DE and Complementary-DE (180nm).

	MUX-DE	Complementary-DE
Number of steps	16	16
Area (μm^2)	2351.76	1776.3
Max dynamic energy (*fJ*)	2518.1	2299.0
Max leakage power (*nW*)	4.54	93.7
Delay range δ (*ps*)	3474.3 $\sim$ 310.7	3271.3 $\sim$ 269.3
Delay range Δ (*ps*)	350.0 $\sim$ 3462.7	464.0 $\sim$ 3455.0
Total delay $\delta + \Delta$ (average) (*ps*)	3776.9	3717.9
Total delay $\delta + \Delta$ variation (*ps*)	92.05 (2.44%)	29.47 (0.79%)

Table 5.2 – Comparison between MUX-DE and Complementary-DE (65nm).

	MUX-DE	Complementary-DE
Number of steps	16	16
Area (μm^2)	366.6	444.6
Max dynamic energy (*fJ*)	390.32	498.90
Max leakage power (*nW*)	114.91	222.77
Delay range δ (*ps*)	1511.9 $\sim$ 283.4	1538.8 $\sim$ 122.2
Delay range Δ (*ps*)	296.5 $\sim$ 1537.5	237.5 $\sim$ 1657.6
Total delay $\delta + \Delta$ (average) (*ps*)	1814.4	1778.8
Total delay $\delta + \Delta$ variation (*ps*)	13.82 (0.76%)	4.01 (0.22%)

Table 5.3 – Comparison between MUX-DE and Complementary-DE (28nm).

	MUX-DE	Complementary-DE
Number of steps	16	16
Active area (transistor area) (μm^2)	2.84	2.94
Max dynamic energy (*fJ*)	67.5	31.8
Max leakage power (*nW*)	42.4	47.2
Delay range δ (*ps*)	457.1 $\sim$ 76.9	482.91 $\sim$ 38.1
Delay range Δ (*ps*)	84.9783 $\sim$ 463.795	70.5 $\sim$ 513.5
Total delay $\delta + \Delta$ (average) (*ps*)	535.8	553.4
Total delay $\delta + \Delta$ variation (*ps*)	18.59 (3.4%)	2.01 (0.4%)

on delay margins (between rise and fall), and *VSDR*. These points aim to present the advantages and disadvantages of each DE mainly focusing on their testability.

The intermediate or total delay comparison target to present the basic characteristics of each DE. Table 5.1, Table 5.2 and Table 5.3 briefly compare the DEs in 180nm, 65nm and 28nm technologies, respectively. These tables aim not only to summarize the differences of each DE, but also how technology influences this difference.

To render the comparison comparable in terms of delay, the circuits were dimensioned to produce similar delay ranges for the same technology. Thus, δ, Δ, and δ+Δ ranges are as close as possible in both DEs.

Because they have gates smaller than the tristates, the area used by the MUX-DE is smaller than that of the Complementary-DE, with the only exception being in the 180nm technology. This is due to the larger number of inverters needed to generate a delay comparable to that of the PS-DE. In other technologies, the MUX-DE takes less area than the Complementary-DE.

Concerning power, the Complementary-DE presents lower energy consumption per transition in 180nm and 28nm. However, the Complementary-DE leakage is always superior to that of the MUX-DE. This result reflects the fact that half of the tristates are always on, with their propagation signal active ($Enable = 1$). This increases leakage, as described in the technology manual for this cell. The Complementary-DE leakage at 180nm already extrapolates the expectations. The difference of more than twenty times in this parameter could indicate some short circuit, but is due to the characteristic of tristate implementation. In this technology, tristate transistor networks have the input acting on the transistors closest to power supply. Since the enable signal is always active ($Enable = 1$), the intermediate nodes of the tristate are also charged, causing the current to be consumed until reaching a $V_{dd} - V_t$ voltage. One solution to this over-consumption can be to change the tristate input to transistors closest to the tristate output and sifting the enable signal (*Enable*) to the transistors closest to power supply. This would eliminate the charge and discharge of the tristate intermediate nodes during the change in input values. Designing a new cell with these characteristics is a future work.

The generated delay ranges of each DE in the Tables can be seen more easily in Figure 5.24. These graphs make explicit the deviations in each configuration generated by both DEs in the three technologies. Note that pd_r and pd_f are contractions of propagation delay rise and propagation delay fall respectively.

It is important to highlight the isometric behavior among MUX-DE simulations. Because they are identical DEs, the ranges for δ and Δ are similar. The Complementary-DE shows a slight variation in the linearity of the evolution of values for both, δ and Δ. These occur due to variations in the capacitances of certain Complementary-DE nodes. In these cases, capacitances at the multiplexer selection nodes (multiplexer input '0') have a different value from the other nodes, slightly altering the behavior in these control values. Despite this small non-linearity, the sum of delays $\delta + \Delta$ is not affected, as it is compensated by the reduction of capacitances in the complementary path that ends up compensating the effect described.

By approaching process variability and its impact on the $\delta + \Delta$ sum, a considerable reduction in the sum variation is expected. To do that, each execution is done by scanning the configurations of δ and Δ in order to keep the sum constant. The variation obtained between the lowest and the highest delay from this sum is expressed as a percentage, based on the lowest delay. This allows to observe how much margin should be imposed on the delay element and, consequently, what is the impact of this margin on reconfiguration, while keeping the sum $\delta + \Delta$ constant.

In this way, both evaluated DEs are simulated in $10,000$ Monte Carlo runs varying process (slow, typical and fast), voltage ($Vdd - 10\%$, Vdd and $Vdd + 10\%$) and temper-

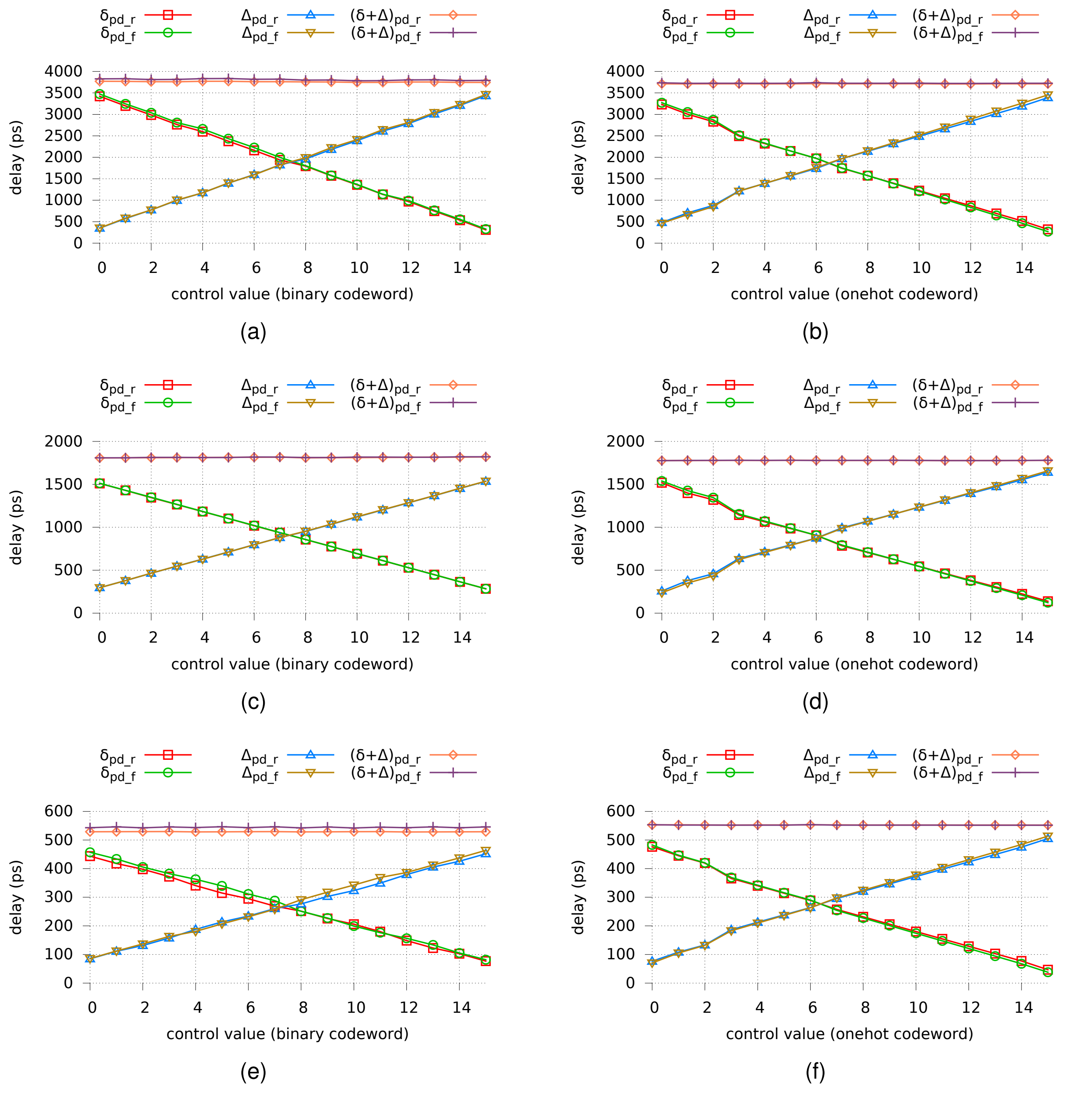
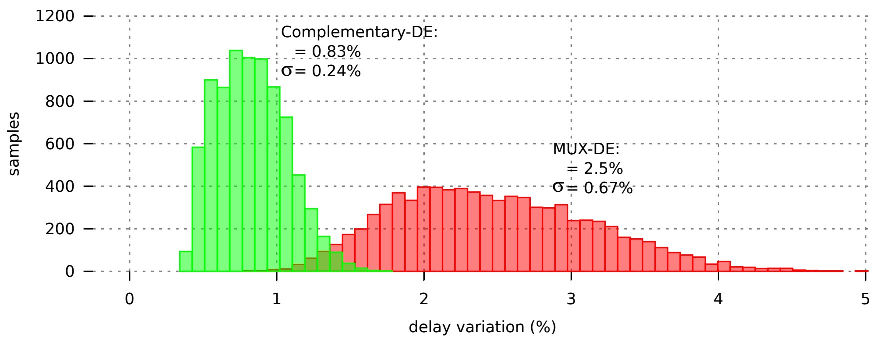


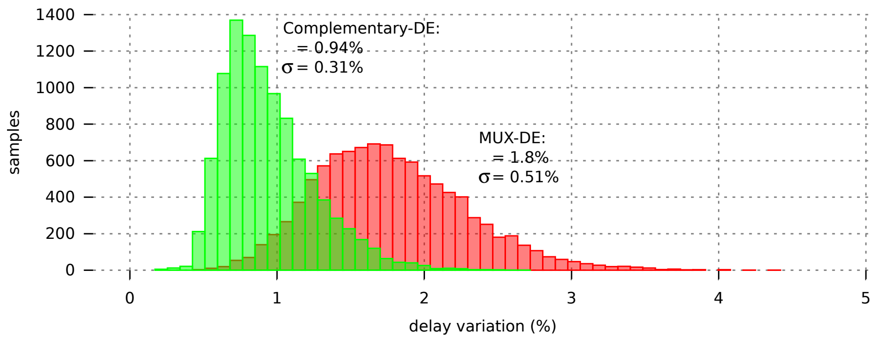
Figure 5.24 – Produced delays and delay sums in: (a) MUX-DE 180nm, (b) Complementary-DE 180nm, (c) MUX-DE 65nm and (d) Complementary-DE 65nm, (e) MUX-DE 28nm and (f) Complementary-DE 28nm. 28nm results do not take wiring effects into account.

ature (-55℃, 25℃ and 125℃). The charts in Figure 5.25 summarize the results of these simulations.

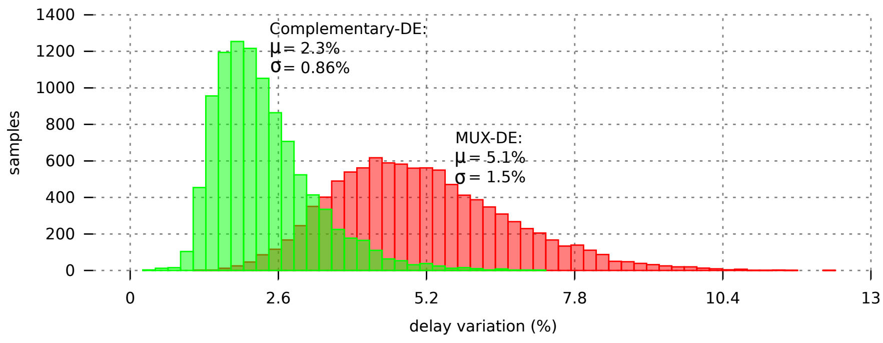
Observing the histograms, it is possible to notice that in all cases the Complementary-DE obtains considerable improvement, reducing the variations in $\delta + \Delta$. As discussed in Section 2.5, 99.9% of cases are present in 3σ of variation in a normal distribution [WST10]. Results in 180nm, presented in Figure 5.25(a) show 64% reduction in the Complementary-DE standard deviation, implying 65.6% in total delay margin reduction when compared with the MUX-DE. For 65nm, Figure 5.25(b) shows a standard deviation reduction in Complemen-



(a)



(b)

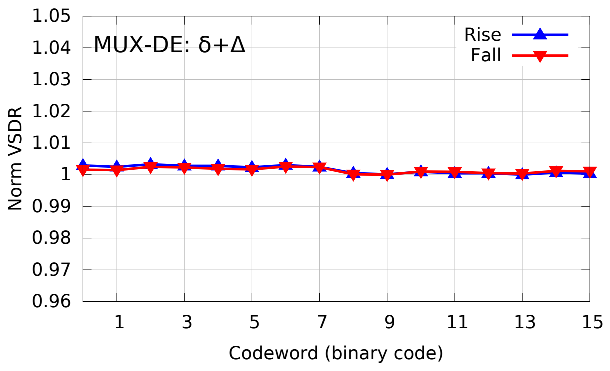


(c)

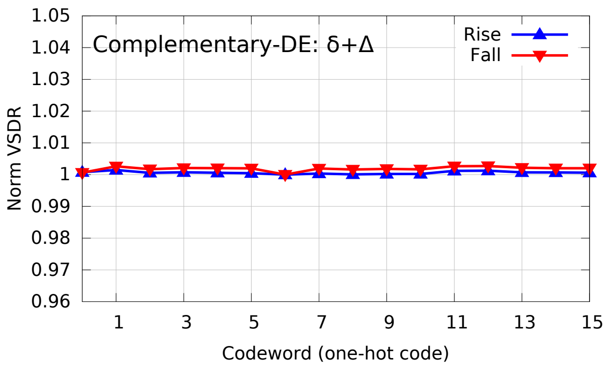
Figure 5.25 – Impact in the $\delta + \Delta$ sum delay margin considering PVT variations for: (a) 180nm, (b) 65nm, and (c) 28nm.

tary-DE of 39.2%, implying 43.8% reduction in total delay margin when compared with the MUX-DE. In 28nm, as Figure 5.25(c) shows, the reduction is of 42.7% in standard deviation, resulting 49.2% in delay margin reduction. It is important to remember that both results in 180nm and 65nm technologies include the parasitic effects on cells and wires. However, it is observable that there is not much discrepancy on these velues when compared with the 28nm technology results, where such parasitics are disregarded.

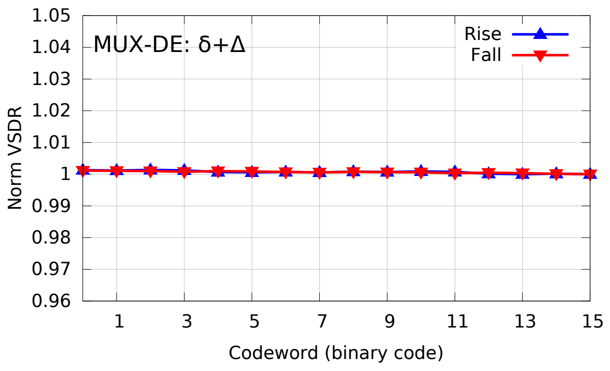
In the *VSDR* comparison, both DEs have margins close to 0%, as can be seen in Figure 5.26. One of the benefits of working with the Blade template is the *VSDR* compensation. If what is removed from the δ path is incremented in Δ, the sum structure of



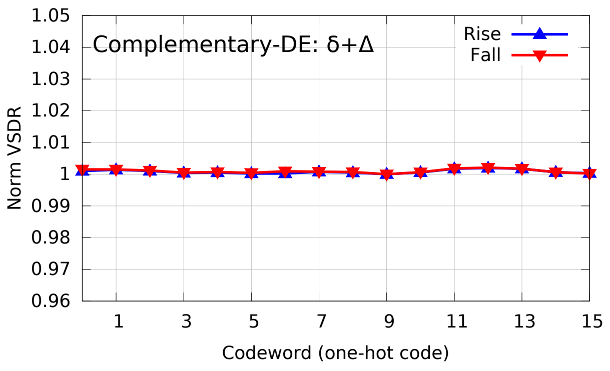
(a)



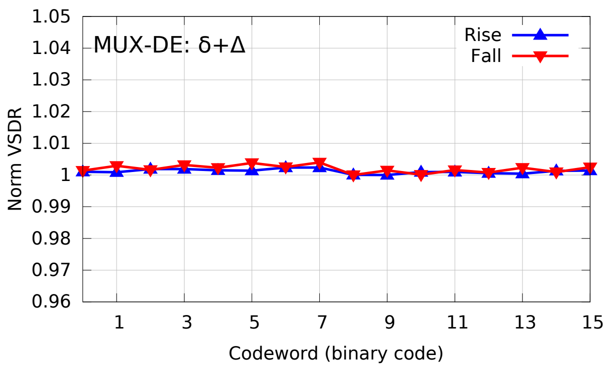
(b)



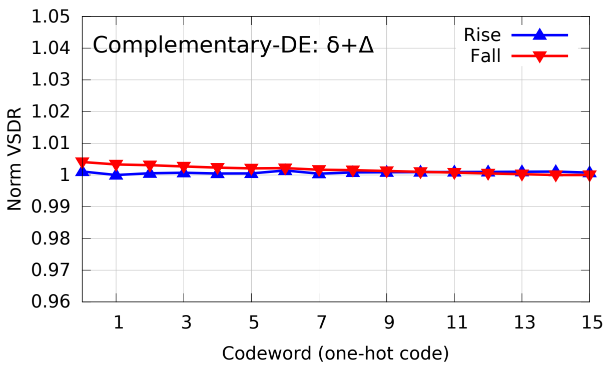
(c)



(d)



(e)



(f)

Figure 5.26 – The *VSDR* variation in $\delta + \Delta$: (a) MUX-DE in 180nm, (b) Complementary-DE in 180nm, (c) MUX-DE in 65nm, (d) Complementary-DE in 65nm, (e) MUX-DE in 28nm, and (f) Complementary-DE in 28nm.

the two elements remains the same, causing *VSDR* to remain pretty much constant. However, it is important to emphasize that the adjustment of process variation in the MUX-DE implementation must act on both DEs, and not only on one of them. This maintains the homogeneity in changes in δ and Δ configuration, guaranteeing that the same number of cells eliminated/added in one DE is added/eliminated in the other DE, keeping the *VSDR* values mostly unchanged. The Complementary-DE does this automatically with its Pass-Section elements.

Although the results point to *VSDR* values that are always very close in both alternatives, it is important to remember that, since the *VSDR* depends on V_t, process variations tend to interfere more in the MUX-DE than in the Complementary-DE. However, Monte Carlo simulations to extract this feature require a lot of time due to voltage scaling. It was intended to add these results to the final text of this Thesis, but due to lack of time, it was not possible to add it in this first version.

5.4 Razor Clock Generation based on the Complementary-DE

Because it takes some of its inspiration from the Razor [EKD+03] template, the Blade template uses similar techniques to propose the resiliency window concept over the full CL delay. Thus, in some ways the delay element used in Blade can also be used to generate frequencies for Razor templates. By a simple modification, the pair of delay elements δ and Δ can be converted, originating an adjustable frequency generator for Razor templates.

The Razor templates, as described in Section 2.1.2, employ as base a frequency higher than nominally supported by the design and a copy of this frequency is delayed to generate the resilience period. However, in spite of operating at a higher frequency, the period between the rising edge of the main clock and the rising edge of the delayed clock guarantees an appropriate execution time of CL even for the CL worst case delay, according to Figure 5.27. Thus, for example, a circuit designed to operate nominally at 1GHz (1000ps period) with a 1.0V supply and at 0.9GHz (1111.2ps) with a 0.9V supply, can run at 1GHz (1000ps period - Faster Clock) with a 0.9V supply, as long as the Delayed Clock is delayed by at least 111.2ps to total 1111.2ps (0.9GHz is the nominal clock for the considered supply). During this delay, the circuit propagates data from the Main FFD in advance to the next stage while the Shadow Latch evaluates its result.

Based on the particularity of the Delayed Clock of circuits designed with a Razor template, it is possible to make a small adaptation in the Complementary-DE to allow the generation of Razor frequency. Both Complementary-DE delays (δ and Δ) have the feature of keep constant the delay sum (the Required Period on Figure 5.27). Setting δ it is possible produce a shorter delay, which could be made equivalent to the smaller period of a Faster Clock Period in a Razor template. To generate the oscillation, therefore, only one inverter is strategically placed between the output of δ and the input of δ itself, as Figure 5.28 shows. To generate the Delayed Clock, it is only necessary to plug the δ output into the Δ input. In this way, it is possible to vary the frequency of the Faster Clock according to the required Razor error rate, keeping the Required Period with the help of Δ.

The Complementary-DE has an intrinsic feature that assists the clock generation for Razor templates. The process variation compensation system allows the template to adjust the required CL period in this case. Thus, after adjusting the Complementary-DE *selection*

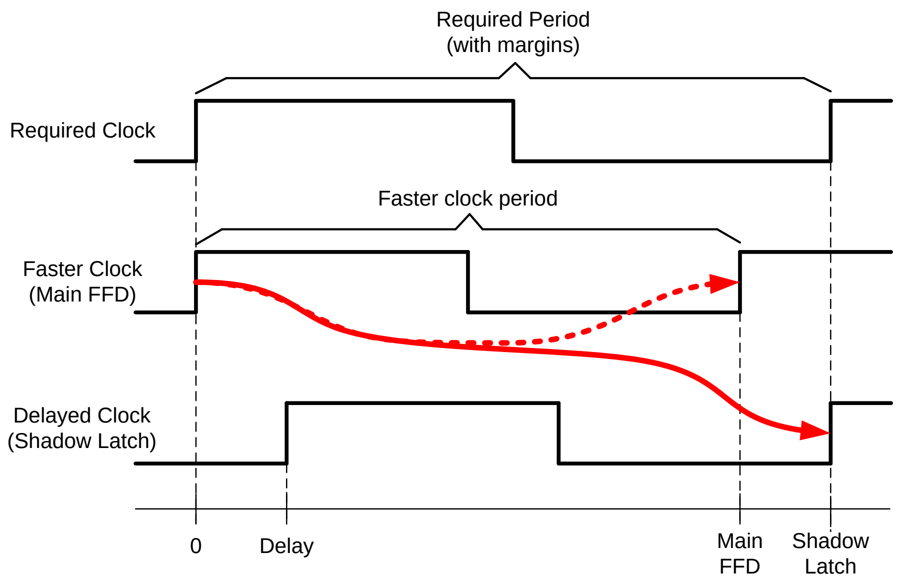


Figure 5.27 – The clock operation in a Razor template.

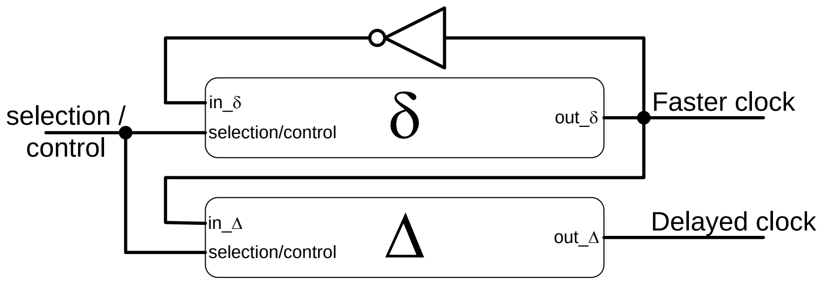


Figure 5.28 – A proposal of clock generator for Razor templates, based on the Complementary-DE.

signal (Required Period), it would be possible vary the *n control* entries to generate *n* faster frequencies. However, it is important to note that the period generated in δ should not be less than Δ. This can cause cycle overlapping leading to anomalous behavior in a Razor template.

6. CONCLUSIONS AND FURTHER WORK

There is a large set of works in the contemporary literature claiming that asynchronous circuits can help solve problems related to modern IC fabrication processes and application requirements. The lack of CAD tools support and characteristics such as high area overhead and power consumption make Quasi-Delay Insensitive (QDI) design a costly choice today. The alternative of using Bundled-Data (BD) design templates is still the easiest way to design an asynchronous circuit. However, the required delay margins in BD templates can result in unacceptable performance degradation compared to synchronous alternatives.

If QDI design is more robust to Voltage Scaling (VS), BD design can also take advantage of the latter technique to reduce power. However, the extra delay margins required to properly operate programmable Delay Elements (DEs) and their configuration under multiple voltages can significantly reduce performance. This Thesis addressed relevant issues in these scenarios, mainly focusing in DE delay variations. It constitutes a step forward in the state of the art to design 2-phase BD asynchronous circuits. This Chapter summarizes the original contributions of the Thesis, putting them into perspective with regard to the latest developments in asynchronous circuit research. The Chapter also proposes a set of directions for future work.

6.1 Thesis Original Contributions

Section 6.1.1 describes this Thesis contributions. During the development of this Thesis, one collaboration was established with another asynchronous research group besides the Author group, as described in the same Section. Also, along the Thesis contributions to asynchronous research fields other than that of the Thesis have also been conducted, as described in Section 6.1.2.

6.1.1 Thesis Contributions

It is possible to list and discuss six original contributions of this Thesis:

1. *Analysis of VS in programmable DEs:* Chapter 3 presented an analysis of delay variation under VS in programmable DEs. It presented the proposition of new metric, called *VSDR* that is useful to model the VS impact on logic circuits. The origin of changes in *VSDR*, caused by transistor sizing and by specific technology characteristics is highlighted and investigated in depth. This work was developed in collaboration

with the Asynchronous CAD/VLSI research group of the University of Southern California (USC) in Los Angeles (United States of America). The *VSDR* concept was first presented in the VLSI Design conference in 2015 [HHS+15]. The Thesis presents an extension of the published research. Four different DEs were compared in three technologies and the results show how DEs can be responsible for more delay margin, if they are not properly designed. The research originally proposed and developed contributed to at least three other independent research works, as described in [Gib16], [SMT+15] and [THG+16].

2. *A novel programmable DE architecture for VS:* As a result from the previously described analysis, one novel programmable DE was proposed for VS applications. With regard to *VSDR* effects, the new DE is more homogeneous when compared with previously proposed DE architectures. It can be reconfigured with low *VSDR* variation, which reduces the required delay margins. This work is not yet published, a journal paper is under writing, extending concepts from [HHS+15] and including this novel Programmable DE as another original contribution.

3. *Analysis of delay test complexity for BD and resilient BD circuits operating under VS:* Chapter 4 depicts a brief analysis approaching VS effects in BD templates. As far as the Author could verify no previous research has investigated delay tests in 2-phase BD circuits operating under VS. It is relevant to show the importance of minimizing the number of test runs in this case and include consideration of coverage for Small Delay Defects (SDDs). It is the Author view that the main advantage of resilient BD templates like Blade consists in providing the possibility of covering SDDs while also including a delay margin analysis through error rate measurements. This Chapter stands out as a starting point in this direction.

4. *Proposal of a new delay element - the Pass-Section DE:* Another contribution is the novel elementary DE called here Pass-Section DE. This DE is capable of changing two outputs according with two inputs. Here, it is only used to pass an signal or section the signal to allow another signal to be propagated in the Complementary-DE. However, other uses can be found in the future to take advantage of the characteristics of this new DE. A journal paper is under writing, describing the Pass-Section DE along with the Complementary-DE, another original contribution of this work described next.

5. *Proposal of a novel DE to reduce sensitivity to process variations impacts in resilient designs - the Complementary-DE:* Probably the main contribution of this Thesis is the proposal of the Complementary-DE, destined to use within Blade and similar resilient BD templates. The Complementary-DE is the first DE to adequately keep the requirement of Blade to dispose of a DE formed by two distinct DEs, the sum of which is constant under most operating conditions, including operation under VS. The peculiarity of this architecture and the novelty in this proposition is at the heart of the originality

of this DE. This DE was idealized during the sandwich stage of the Author in the USC. The Complementary-DE is the object of a paper currently under writing.

6. *Proposal of a novel clock generator for resilient designs:* Another original contribution of this Thesis consists in the conversion of the resilient BD DE (the Complementary-DE) to operate as a clock generator for synchronous resilient templates. This is a late development topic of this research and requires further development work to unfold its full potential. However it is an original work, since no previously proposed clock generator employs a similar component, or displays similar properties.

6.1.2 Other Contributions

During his Thesis work, the Author also worked on other parallel projects, described herein. All works mentioned here are out of main scope of this volume, but have relevance in asynchronous design and/or on the design of Globally Asynchronous, Locally Synchronous (GALS) circuits.

All synchronous circuits require an external clock. However, GALS circuits can benefit from frequency scaling, working with several different clock sources. Each synchronous module in a GALS circuit has an asynchronous interface. The Author designed a low area Digitally Controlled Oscillator (DCO) for synchronous modules. This DCO is robust to PVT variations and provides glitch-free operation. It allows 256 configuration steps to compensate PVT variations and another 16 glitch free selections of frequencies between 100MHz and 1GHz. Two mechanisms to pause and gate the clock are available for controller compensation actions as well as for power reduction. This work was first published in the Latin American Symposium in Circuits and Systems (LASCAS) 2015 [HHM+15]. Having been classified among the best papers in that conference edition, an extension of this work was invited for submission to the Analog Integrated Circuits and Signal Processing (AICSP) 2016, and was published there [HHM+16].

Another two works counted with the Author co-authorship, both in Quasi-Delay Insensitive (QDI) circuit design. The first approaches Null Convention Logic (NCL) design (a class of QDI design) [GHMC14] and is about the possibility of jeopardy to the functionality of such circuits when subject to transients caused by Single Event Effects (SEEs). The work evaluates the effect of using Schmitt triggers on output inverters to help mitigating such problems in NCL gates, investigating the sufficiency of this approach. The second work compares classical minterm and maxterm synthesis logic styles [MGHC14] in QDI design. These are logic styles based on the use of return-to-one 4-phase protocols. The results in this paper show improvement of over 300% in C-Element tolerance to transient faults.

6.2 Future Work

This Thesis is a starting point for further research work involving delay elements in BD templates. The document discussed and proposed a set of new ideas, analyses and developments, but of course the subject was not exhausted by this research effort. This Section explores some possible directions for additional research to conduct on the subject.

1. *VSDR equation counterproof:* According to good evidence practices, an equation can only be proven with proof and counterproof. The *VSDR* equation presented in Section 3.1 only proves the threshold voltage influence by physical simulation. In this case, the mathematical validation based on transistor model values is the counterproof required to validate the *VSDR* modeling.

2. *VSDR evaluation under process variations:* One of the problems pointed by [TPC11] is that process variations can change the threshold voltage by values that reach around 30%. It is interesting to run Monte Carlo simulations in programmable DEs to evaluate this impact in *VSDR*. Since tristate devices contain stacked transistors, the probability of process variations interfering in the delay of these cells is higher than those affecting inverters or buffers.

3. *VSDR modeling covering the subthreshold supply voltage region:* This is an interesting point this research did not exploit. The knowledge of *VSDR* in the subthreshold region can help designers predicting the delay cell behavior and, from that, design DEs for really wide ranges of VS in BD. In this way, it will be possible to investigate if BD design can achieve VS robustness competitive or comparable to QDI design.

4. *Prototype and validate data for Complementary-DE and MUX-DE:* The Complementary-DE deserves a silicon implementation to validate its proposition. A project has been submitted to obtain area to fabricate a chip in the Europractice mini@SIC program, using the TSMC 180nm CMOS technology. The Complementary-DE and the MUX-DE can be easily implemented, because both IPs for 180nm are already developed up to the layout level. These modules have a small area footprint and require only a small number of external IC pins. From the prototype, it will be possible to compare both DEs physically and check if the results match with the obtained post-layout simulation results described here.

5. *Implement a 28nm tristate cell:* The complete test for the Complementary-DE in FDSOI 28nm technology requires the layout of at least one tristate cell. Unfortunately, this technology does not provide a single gate with high impedance output (required for Pass-Section DE interconnection). It is interesting to set the ASTRAN tool [ZR14] to automatically generate such cells, contributing to enhance the ASCEnD asynchronous

library [MOPC11]. However various issues require treatment before this is possible, including: (i) ASTRAN is not yet enabled to deal with the complexities of FDSOI layout or even for CMOS nodes below 45nm; (ii) The GAPH research group has not yet addressed 28nm cell layout generation either manually or automatically, except through the use of provided standard cell libraries; (iii) The cell characterization process for the 28nm FDSOI technology has yet to be addressed.

6. *Implement DE IP shields automatically:* One problem involving DEs is the crosstalk between wires. Tehranipoor et al. [TPC11] show variations of up to 225ps in propagation delay due to this effect. Thus, to reduce crosstalk, a DE IP needs a shield in some higher metal level. Unfortunately, the power plan automatically creates vias in all designs to connect the shield to ground or supply source, possibly causing short circuits in the routed wires. Within the time available for investigating this issue, the Author could not find a way to automatically produce the DE shield correctly.

7. *Implement isochronic output for out_δ:* It is clear to the Author that there are two node problems in the Complementary-DE design: *out_*δ and $\sim$*in_*Δ (refer to Figure 5.12. These nodes share a large number of gates in their respective fanouts. $\sim$*in_*Δ is easily solved declaring it as a clock signal. The clock tree is generated, reducing the wire delay difference between gates. However, the output *out_*δ problem is not that easy to solve. The isochronic signal distribution is a classical problem in asynchronous design [BOF10].

8. *Define a flow for the complete parasitics extraction to automate the Complementary-DE/Mux-DE flow design to integrate in Blade:* One problem discovered by the Author is the integration required between the Virtuoso database and PEX. The only manner found to obtain the parasitic extraction needs human interference. If this step is automated, it is possible to fully automate the DE IP implementation. Here, what is required is a list of gates and a script that executes the design flow, extract parasitics, simulate and do corrections to achieve the required delay. In this way, the design flow could be integrated to Blade do generate its DEs.

9. *Conduct an evaluation of the Complementary-DE VSDR under process variations:* Just as in the first item in this list, it is interesting to investigate how *VSDR* impacts the Complementary-DE under process variations. The time required to run this evaluation under low supply voltage, the large number of nodes where to compute parasitics and the number of DE configurations makes this simulation excessively long. As tristate devices could be more sensitive to process variations, probably Complementary-DE is, too. However, the characteristic of this DE of keeping the main path partially unchanged could be an advantage for this kind of variations. MUX-DE needs also to be simulated to compare the results with those of the Complementary-DE.

10. *Employ error rates to identify fault-free devices and delay margins:* Another future work can be identified in the context of the Blade resilient template: the error rate evaluation. As Blade produces an error rate value at runtime, during tests this feature can be used to identify delay defects, constituting an extra parameter to employ during test. As an example, if the error rate is bigger than expected, probably the delay margins are lower than those required by the circuit. The Author sees a significant opportunity for future work in this topic.

REFERENCES

[AESK+12] Al-Eryani, J.; Stanitzki, A.; Konrad, K.; Tavangaran, N.; Bruckmann, D.; Kokozinski, R. "Low power area-efficient delay element with a wide delay range". In: IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2012, pp. 717–720.

[AFE+05] Amde, M.; Felicijan, T.; Efthymiou, A.; Edwards, D.; Lavagno, L. "Asynchronous on-chip networks", *IEE Proceedings - Computers and Digital Techniques*, vol. 152–2, Mar 2005, pp. 273–283.

[AYI17] Agwa, S.; Yahya, E.; Ismail, Y. "Power efficient AES core for IoT constrained devices implemented in 130nm CMOS". In: IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 521–524.

[BA04] Bushnell, M.; Agrawal, V. "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits". Springer US, 2004, 690p.

[Bak10] Baker, R. J. "CMOS: Circuit Design, Layout, and Simulation". Wiley, 2010, 944p.

[BOF10] Beerel, P. A.; Ozdag, R. O.; Ferretti, M. "A Designer's Guide to Asynchronous VLSI". Cambridge University Press, 2010, 339p.

[BR07] Beerel, P.; Roncken, M. "Low Power and Energy Efficient Asynchronous Design", *Journal of Low Power Electronics*, vol. 3–3, 2007, pp. 234–253.

[Cal98] Calazans, N. "Projeto Lógico Automatizado de Sistemas Digitais Sequenciais". Imprinta, 1998, 318p.

[CCGC10] Chen, J.; Chong, K. S.; Gwee, B. H.; Chang, J. S. "An ultra-low power asynchronous quasi-delay-insensitive (QDI) sub-threshold memory with bit-interleaving and completion detection". In: IEEE International New Circuits and Systems Conference (NEWCAS), 2010, pp. 117–120.

[CDS+08] Chakraborty, A.; Duraisami, K.; Sathanur, A.; Sithambaram, P.; Benini, L.; Macii, A.; Macii, E.; Poncino, M. "Dynamic Thermal Clock Skew Compensation using Tunable Delay Buffers", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16–6, Jun 2008, pp. 639–649.

[CL10] Chang, X.; Lian, Y. "A Quasi-Delay-Insensitive Dual-Rail low-pass filter working in subthreshold region". In: Biomedical Circuits and Systems Conference (BioCAS), 2010, pp. 214–217.

[CLM+16] Cortadella, J.; Lupon, M.; Moreno, A.; Roca, A.; Sapatnekar, S. S. “Ring Oscillator Clocks and Margins”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2016, pp. 19–26.

[Col04] Colinge, J. “Silicon-on-Insulator Technology: Materials to VLSI”. Springer US, 2004, 366p.

[CPR10] Chang, I. J.; Park, S. P.; Roy, K. “Exploring Asynchronous Design Techniques for Process-tolerant and Energy-efficient Subthreshold Operation”, *IEEE Journal of Solid-State Circuits*, vol. 45–2, 2010, pp. 401–410.

[CVG07] Chelcea, T.; Venkataramani, G.; Goldstein, S. C. “Area Optimizations for Dual-Rail Circuits Using Relative-Timing Analysis”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2007, pp. 117–128.

[CVPS11] Chen, J.; Vasudevan, D.; Popovici, E.; Schellekens, M. “Design of a Low Power, Sub-Threshold, Asynchronous Arithmetic Logic Unit Using a Bidirectional Adder”. In: EUROMICRO Conference on Digital System Design (DSD), 2011, pp. 301–308.

[DS94] Deokar, R. B.; Sapatnekar, S. S. “A graph-theoretic approach to clock skew optimization”. In: IEEE International Symposium on Circuits and Systems (ISCAS), 1994, pp. 407–410.

[DTP09] Das, S.; Tokunaga, C.; Pant, S. “RazorII: 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: IEEE/ACM International Symposium on Microarchitecture (MICRO), 2003, pp. 7–18.

[Elr11] Elrabaa, M. E. S. “Robust Two-Phase RZ Asynchronous SoC Interconnects”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19–6, Jun 2011, pp. 1086–1089.

[FFKP13] Fojtik, M.; Fick, D.; Kim, Y.; Pinckney, N. “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.

[Gaz15] GazettaByte. “Altera’s 30 billion transistor FPGA”. Source: http://www.gazettabyte.com/home/2015/6/28/alteras-30-billion-transistor-fpga.html, 2016-11-17.

[GBN13] Ghiribaldi, A.; Bertozzi, D.; Nowick, S. M. “A Transition-Signaling Bundled Data NoC Switch Architecture for Cost-effective GALS Multicore Systems”. In: Design, Automation & Test in Europe Conference and Exhibition (DATE), 2013, pp. 332–337.

[GHMC14] Guazzelli, R.; Heck, G.; Moreira, M.; Calazans, N. “Schmitt trigger on output inverters of NCL gates for soft error hardening: Is it enough?” In: Latin American Test Workshop (LATW), 2014, pp. 1–5.

[Gib16] Gibiluka, M. “Analysis of Voltage Scaling Effects in the Design of Resilient Circuits”, Master’s Thesis, PPGCC-Pontifical Catholic University of Rio Grande do Sul, 2016, 103p.

[GJZ+17] Gong, F.; Ju, L.; Zhang, D.; Zhao, M.; Jia, Z. “Cooperative DVFS for energy-efficient HEVC decoding on embedded CPU-GPU architecture”. In: ACM/IEEE Design Automation Conference (DAC), 2017, pp. 1–6.

[GMC15] Gibiluka, M.; Moreira, M.; Calazans, N. “A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework”. In: EUROMICRO Conference on Digital System Design (DSD), 2015, pp. 79–86.

[GMMC15] Gibiluka, M.; Moreira, M. T.; Moraes, F. G.; Calazans, N. L. V. “BAT-Hermes: A transition-signaling bundled-data NoC router”. In: IEEE Latin American Symposium on Circuits Systems (LASCAS), 2015, pp. 1–4.

[HB13] Hamon, J.; Beigné, E. “Automatic Leakage Control for Wide Range Performance QDI Asynchronous Circuits in FD-SOI Technology”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2013, pp. 142–149.

[HCGC15] Ho, W. G.; Chong, K. S.; Gwee, B. H.; Chang, J. S. “Low power sub-threshold asynchronous quasi-delay-insensitive 32-bit arithmetic and logic unit based on autonomous signal-validity half-buffer”, *IET Circuits, Devices Systems*, vol. 9–4, 2015, pp. 309–318.

[HHC+15] Hand, D.; H., H.; Chang, 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.

[HHM+15] Heck, G.; Heck, L. S.; Moreira, M. T.; Moraes, F. G.; Calazans, N. L. V. “A digitally controlled oscillator for fine-grained local clock generators in MPSoCs”. In: IEEE Latin American Symposium on Circuits Systems (LASCAS), 2015, pp. 1–4.

[HHM+16] Heck, G.; Heck, L. S.; Moreira, M. T.; Moraes, F. G.; Calazans, N. L. V. “A new local clock generator for globally asynchronous locally synchronous MPSoCs”, *Analog Integrated Circuits and Signal Processing*, vol. 89–3, Dec 2016, pp. 631–640.

[HHS+15] Heck, G.; Heck, L.; Singhvi, A.; Moreira, M. T.; Beerel, P.; Calazans, N. L. V. “Analysis and Optimization of Programmable Delay Elements for 2-Phase Bundled-Data Circuits”. In: International Conference on VLSI Design (VLSID), 2015, pp. 321–326.

[HMH+15] Hand, D.; Moreira, M. T.; H., H.; Chen, D.; Butzke, F.; 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.

[HSL17] Huang, K. Y.; Shen, T. Y.; Li, C. M. “Test methodology for dual-rail asynchronous circuits”. In: ACM/IEEE Design Automation Conference (DAC), 2017, pp. 1–6.

[Ito09] Itoh, K. “Adaptive circuits for the 0.5-V nanoscale CMOS era”. In: IEEE International Solid-State Circuits Conference (ISSCC), 2009, pp. 14–20.

[ITR] ITRS. “The International Technology Roadmap for Semiconductors (ITRS)”. Source: http://www.itrs.net/, 2015-06-06.

[JKMK13] Jasielski, J.; Kuta, S.; Machowski, W.; Kołodziejski, W. “An analog dual delay locked loop using coarse and fine programmable delay elements”. In: International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES), 2013, pp. 185–190.

[KA16] Kuentzer, F. A.; Amory, A. M. “Fault Classification of the Error Detection Logic in the Blade Resilient Template”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2016, pp. 37–42.

[KHS07] Konishi, T.; Hamada, N.; Saito, H. “A Control Circuit Synthesis Method for Asynchronous Circuits in Bundled-Data Implementation”. In: IEEE International Conference on Computer and Information Technology (CIT), 2007, pp. 847–852.

[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 (ISSCC), 2013, pp. 264–265.

[KKJ12] Kołodziejski, W.; Kuta, S.; Jasiełski, J. “Current controlled delay line elements’ improvement study”. In: International Conference on Signals and Electronic Systems (ICSES), 2012, pp. 1–4.

[KNC02] Kao, J.; Narendra, S.; Chandrakasan, A. “Subthreshold leakage modeling and reduction techniques”. In: IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2002, pp. 141–148.

[KY09] Kobenge, S. B.; Yang, H. “A Power Efficient Digitally Programmable Delay Element for Low-Power VLSI Applications”. In: Asia Symposium on Quality Electronic Design (ASQED), 2009, pp. 83–87.

[LNS13] Liu, J.; Nowick, S. M.; Seok, M. “Soft MOUSETRAP: A Bundled-Data Asynchronous Pipeline Scheme Tolerant to Random Variations at Ultra-Low Supply Voltages”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2013, pp. 1–7.

[LSGB11] Ludovici, D.; Strano, A.; Gaydadjiev, G. N.; Bertozzi, D. “Mesochronous NoC Technology for Power-efficient GALS MPSoCs”. In: ACM International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC), 2011, pp. 27–30.

[MAGC14] Moreira, M. T.; Arendt, M. E.; Guazzelli, R. A.; Calazans, N. L. V. “A New CMOS Topology for Low-Voltage Null Convention Logic Gates Design”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2014, pp. 93–100.

[Mar90] Martin, A. J. “The Limitations to Delay-insensitivity in Asynchronous Circuits”. In: MIT Conference on Advanced Research in VLSI, 1990, pp. 263–278.

[MCA12] Meenakshi, J.; Chowdary, G. R.; Aditya, A. L. G. N. “Implementations of DPDE for Delay Locked Loop for High Frequency Clock of 2.5GHz High Speed Applications”, *International Journal of Soft Computing and Engineering*, vol. 2–2, May 2012, pp. 2231–2307.

[MCFB16] Miorandi, G.; Celin, A.; Favalli, M.; Bertozzi, D. “A built-in self-testing framework for asynchronous bundled-data NoC switches resilient to delay variations”. In: IEEE/ACM International Symposium on Networks-on-Chip (NOCS), 2016, pp. 1–8.

[MGHC14] Moreira, M. T.; Guazzelli, R. A.; Heck, G.; Calazans, N. L. V. “Hardening QDI Circuits Against Transient Faults Using Delay-insensitive Maxterm Synthesis”. In: ACM Great Lakes Symposium on VLSI (GLSVLSI), 2014, pp. 3–8.

[MGT00] Mahapatra, N. R.; Garimella, S. V.; Tareen, A. “An Empirical and Analytical Comparison of Delay Elements and a New Delay Element Design”. In: IEEE Computer Society Workshop on VLSI (WVLSI), 2000, pp. 81–86.

[MNS03] Maymandi-Nejad, M.; Sachdev, M. “A digitally programmable delay element: design and analysis”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11–5, Oct 2003, pp. 871–878.

[MNS05] Maymandi-Nejad, M.; Sachdev, M. “A monotonic digitally controlled delay element”, *IEEE Journal of Solid-State Circuits*, vol. 40–11, Nov 2005, pp. 2212–2219.

[MOPC11] Moreira, M.; Oliveira, B.; Pontes, J.; Calazans, N. “A 65nm standard cell set and flow dedicated to automated asynchronous circuits design”. In: IEEE International System-on-Chip Conference (SOCC), 2011, pp. 99–104.

[MRSM17] Martins, A.; Ruaro, M.; Santana, A.; Moraes, F. G. “Runtime energy management under real-time constraints in MPSoCs”. In: IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1–4.

[Rab] Rabaey, J. “For Lower Power, Re-Think Computing”. Source: https://community.cadence.com/cadence_blogs_8/b/ii/posts/jan-rabaey-keynote-for-lower-power-re-think-computing, 2017-12-06.

[RCN03] Rabaey, J. M.; Chandrakasan, A.; Nikolic, B. “Digital Integrated Circuits: A Design Perspective”. Prentice Hall, 2003, 761p.

[RDH+17] Russell, P.; Döge, J.; Hoppe, C.; Preußer, T. B.; Reichel, P.; Schneider, P. “Implementation of an asynchronous bundled-data router for a GALS NoC in the context of a VSoC”. In: IEEE International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), 2017, pp. 195–200.

[RGP+15] Roncken, M.; Gilla, S. M.; Park, H.; Jamadagni, N.; Cowan, C.; Sutherland, I. “Naturalized Communication and Testing”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2015, pp. 77–84.

[RLCM12] Rosa, T. R.; Larréa, V.; Calazans, N.; Moraes, F. G. “Power consumption reduction in MPSoCs through DFS”. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2012, pp. 1–6.

[SF13] Sparsø, J.; Furber, S. “Principles of Asynchronous Circuit Design: A Systems Perspective”. Springer US, 2013, 337p.

[SGB11] Stevens, K. S.; Golani, P.; Beerel, P. A. “Energy and Performance Models for Synchronous and Asynchronous Communication”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19–3, Mar 2011, pp. 369–382.

[SGY+09] Stevens, K. S.; Gebhardt, D.; You, J.; Xu, Y.; Vij, V.; Das, S.; Desai, K. “The Future of Formal Methods and GALS Design”, *Electronic Notes in Theoretical Computer Science*, vol. 245, Aug 2009, pp. 115–134.

[SHB+14] Saifhashemi, A.; Hand, D.; Beerel, P. A.; Koven, W.; Wang, H. “Performance and Area Optimization of a Bundled-Data Intel Processor through Resynthesis”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2014, pp. 110–111.

[SLK+00] Sidiropoulos, S.; Liu, D.; Kim, J.; Wei, G.; Horowitz, M. “Adaptive Bandwidth DLLs and PLLs using Regulated Supply CMOS Buffers”. In: Symposium on VLSI Circuits (VLSIC), 2000, pp. 124–127.

[SLKR13] Sayyed, A.; Lavagno, L.; Khalid, S.; Rahman, N. U. “Implementation and performance analysis of variable latency adders”. In: IEEE International SOC Conference (SOCC), 2013, pp. 267–272.

[SM06] Shi, F.; Makris, Y. “Testing Delay Faults in Asynchronous Handshake Circuits”. In: IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2006, pp. 193–197.

[SMT+15] Singhvi, A.; Moreira, M. T.; Tadros, R.; Calazans, N. L. V.; Beerel, P. A. “A Fine-Grained, Uniform, Energy-Efficient Delay Element for FD-SOI Technologies”. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2015, pp. 27–32.

[SO15] Sato, S.; Ohtake, S. “A Delay Measurement Mechanism for Asynchronous Circuits of Bundled-Data Model”. In: IEEE International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), 2015, pp. 243–248.

[SRG+01] Stevens, K. S.; Rotem, S.; Ginosar, R.; Beerel, P.; Myers, C. J.; Yun, K. Y.; Koi, R.; Dike, C.; Roncken, M. “An asynchronous instruction length decoder”, *IEEE Journal of Solid-State Circuits*, vol. 36–2, Feb 2001, pp. 217–228.

[TB02] Tugsiriavisut, S.; Beerel, P. A. “Control circuit templates for asynchronous bundled-data pipelines”. In: Design, Automation & Test in Europe Conference and Exhibition (DATE), 2002, pp. 1098.

[TBW+09] Tschanz, J.; Bowman, K.; Walstra, S.; Agostinelli, M.; Karnik, T.; De, V. “Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance”. In: Symposium on VLSI Circuits (VLSIC), 2009, pp. 112–113.

[THG+16] Tadros, R. N.; Hua, W.; Gibiluka, M.; Moreira, M. T.; Calazans, N. L. V.; Beerel, P. A. “Analysis and Design of Delay Lines for Dynamic Voltage Scaling Applications”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2016, pp. 11–18.

[THK+05] Tugsinavisut, S.; Hong, Y.; Kim, D.; Kim, K.; Beerel, P. A. “Efficient asynchronous bundled-data pipelines for DCT matrix-vector multiplication”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13–4, Apr 2005, pp. 448–461.

[TM11] Tsividis, Y.; McAndrew, C. “Operation and Modeling of the MOS Transistor”. Oxford University Press, 2011, 752p.

[TPC11] Tehranipoor, M.; Peng, K.; Chakrabarty, K. “Test and Diagnosis for Small-Delay Defects”. Springer US, 2011, 212p.

[URV11] Ubar, R.; Raik, J.; Vierhaus, H. “Design and Test Technology for Dependable Systems-on-chip”. Information Science Reference, 2011, 550p.

[VTN12] Vezyrtzis, C.; Tsividis, Y.; Nowick, S. M. “Designing Pipelined Delay Lines with Dynamically-Adaptive Granularity for Low-Energy Applications”. In: IEEE International Conference on Computer Design (ICCD), 2012, pp. 329–336.

[WH11] Weste, N.; Harris, D. “CMOS VLSI Design: A Circuits and Systems Perspective”. Pearson Education, 2011, 864p.

[WK17] Waugaman, M.; Koven, W. “Sharp - A Resilient Asynchronous Template”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2017, pp. 83–84.

[Wol05] Wolf, W. “Computers as Components: Principles of Embedded Computing System Design”. Morgan Kaufmann, 2005, 656p.

[WST10] Wang, L.; Stroud, C.; Touba, N. “System-on-Chip Test Architectures: Nanometer Design for Testability”. Elsevier Science, 2010, 896p.

[WWW06] Wang, L.; Wu, C.; Wen, X. “VLSI Test Principles and Architectures: Design for Testability”. Elsevier Science, 2006, 808p.

[XZVH09] Xiong, J.; Zolotov, V.; Visweswariah, C.; Habitz, P. A. “Optimal Test Margin Computation for At-Speed Structural Test”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28–9, Sep 2009, pp. 1414–1423.

[YHIF11] Yoneda, T.; Hori, K.; Inoue, M.; Fujiwara, H. “Faster-than-at-speed test for increased test quality and in-field reliability”. In: IEEE International Test Conference (ITC), 2011, pp. 1–9.

[YLM+10] Yang, W.; Lin, C.-H.; Morshed, T. H.; Lu, D.; Niknejad, A.; Hu, C. “BSIMSOIv4.4 MOSFET model users’ manual”, Technical Report, The Regents of the University of California, 2010, 128p.

[ZR14] Ziesemer Jr., A.; Reis, R. “Simultaneous Two-Dimensional Cell Layout Compaction Using MILP with ASTRAN”. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2014, pp. 350–355.

[ZSD10] Zhou, L.; Smith, S. C.; Di, J. “Bit-Wise MTNCL: An ultra-low power bit-wise pipelined asynchronous circuit design methodology”. In: IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2010, pp. 217–220.

APPENDIX A – THE PARAMETERIZABLE COMPLEMENTARY-DE CODE

This Appendix presents the Verilog code used as input to the Complementary-DE synthesis. It relies on templates available in the svn of the design flow, in the link:

https://corfu.pucrs.br/svn/delaylines/tags/compde.

The Verilog used here is in the *rtl* folder of the design database.

The file *Cells.v* defines the cells that are used for the Complementary-DE elaboration. Here, the user (or script) inserts the respective tristates, inverters, buffers and multiplexers in the respective modules.

```
module Tristate (
in ,
en ,
out
);
//Input declaration
input in;
input en;
//Ouput declaration
output out;
//Port Data types
wire  in;
wire  en;
wire  out;

TECH_ITX U0 (.A (in), .Z (out), .E (en));

endmodule


module Inverter_control (
in ,
out
);
//Input declaration
input in;
//Ouput declaration
output out;
//Port Data types
wire  in;
wire  out;
```

```
TECH_IVX_MIN    U4(.A (in), .Z (out));

endmodule


module Inverter_DELTA (
in  ,
out
);
//Input declaration
input in;
//Ouput declaration
output out;
//Port Data types
wire  in;
wire  out;

TECH_IVX_X U00(.Z (out), .A (in));

endmodule


module Mux (
in0  ,
in1  ,
s   ,
out
);
//Input declaration
input in0;
input in1;
input s;
//Ouput declaration
output out;
//Port Data types
wire  in0;
wire  in1;
wire  s;
wire  ns;
wire  out;
```

```
Tristate U0(.in (in0), .out (out), .en (ns));
Tristate U1(.in (in1), .out (out), .en (s));
Inverter_control U2(.in (s), .out (ns));

//TECH_MUXI21  U(.D1 (in1), .D0 (in0), .S0 (s), .Z (out) );

endmodule


module Tristate_buffer (
in  ,
out
);
//Input declaration
input in;
//Ouput declaration
output out;
//Port Data types
wire  in;
wire  n0;
wire  out;

Tristate U0(.in (in), .out (n0), .en (1'b1));
Tristate U1(.in (n0), .out (out), .en (1'b1));

endmodule
```

The file *PSDE.v* has the Pass-Section and Pass-nSection DEs.

```
//  Dependencies:
//  > ./Cells.v
//
//  Interface description:
//
//  PS—DE
//
//                                         n_in_bdelta
//                                               |
//                                             __|__
//                     vdd          control__\U3 /
//                      |               |       \ /
//                   |  \|           |  \O        O
//                   |   \    n0     |   \        |
//      in_ps ----|U0 O----+----|U1 O------+----- out_ps
//                |  /     |    |  /
//                | /      |    | /
//                       __|__
//             control__\U2 /
//                        \ /
//                         O
//                         |
//                         |
//                     out_ldelta

module PSDE (
in_ps,
n_in_bdelta,
control,
out_ps,
out_ldelta
);
//Input declaration
input in_ps;
input n_in_bdelta;
input control;
//Ouput declaration
output out_ps;
output out_ldelta;
//Port Data types
wire  in_ps;
wire  n_in_bdelta;
wire  control;
wire  ncontrol;
wire  out_ps;
wire  out_ldelta;
wire  n0;

Tristate        U0 (.in (in_ps), .out (n0), .en (1'b1));
Tristate        U1 (.in (n0), .out (out_ps), .en (ncontrol));
Tristate        U2 (.in (n0), .out (out_ldelta), .en (control));
Tristate        U3 (.in (n_in_bdelta), .out (out_ps), .en (control));
Inverter_control U4(.in (control), .out (ncontrol));

endmodule
```

```
// PnS–DE
//
//                                          n_in_bdelta
//                                               |
//                                             __|__
//                                 control__\U3 /       vdd
//                                     |        \ /         |
//                                 |  \O         O      |  \|
//                                 |   \         |      |   \
//       in_ps ––––––––+––––|U1 O––––––+––––|U0 O–––––  out_ps
//                     |      |   /        n1      |   /
//                     |      |  /                 |  /
//                   __|__
//          control__\U2 /
//                    \ /
//                     O
//                     |
//                     |
//                out_ldelta


module PnSDE (
in_ps ,
n_in_bdelta ,
control ,
out_ps ,
out_ldelta
);
//Input declaration
input in_ps;
input n_in_bdelta;
input control;
//Ouput declaration
output out_ps;
output out_ldelta;
//Port Data types
wire  in_ps;
wire  n_in_bdelta;
wire  control;
wire  ncontrol;
wire  out_ps;
wire  out_ldelta;
wire  n1;

Tristate         U1 (.in (in_ps), .out (n1), .en (ncontrol));
Tristate         U0 (.in (n1), .out (out_ps), .en (1'b1));
Tristate         U2 (.in (in_ps), .out (out_ldelta), .en (control));
Tristate         U3 (.in (n_in_bdelta), .out (n1), .en (control));
Inverter_control U4 (.in (control), .out (ncontrol));

endmodule
```

The file *ComplementaryDE.v* uses the definitions of the two previous files to elaborate the Complementary-DE. In addition, three parameters can be defined by the user (or script) to set the DE dimensions (MINIMUM_DELAY, RESIL_WINDOW_SIZE and PROC_-VAR_STAGES).

```
// Dependencies:
// > ./Cells.v
// > ./PSDE.v
// Generics for configuration:
//  MINIMUM_DELAY ==> Number of static buffers (minimum little delta)
//  RESIL_WINDOW_SIZE ==> Number of stages for Resilient Window (control
   --> one hot code)
//  PROC_VAR_STAGES ==> How many bits to adjust Process Variation (Number
    of MUXES)

`ifndef PROC_VAR_STAGES
        `define PROC_VAR_STAGES 4
`endif

`ifndef RESIL_WINDOW_SIZE
        `define RESIL_WINDOW_SIZE 16
`endif

`ifndef MINIMUM_DELAY
        `define MINIMUM_DELAY 0
`endif


module ComplementaryDE (
little_delta_in   ,
big_delta_in      ,
control            ,
selection          ,
little_delta_out  ,
big_delta_out
);
//Input declaration
input little_delta_in;
input big_delta_in;
input control;
input selection;
//Ouput declaration
output little_delta_out;
output big_delta_out;
//Port Data types
wire  little_delta_in;
wire  big_delta_in;
wire  nbig_delta_in;
wire  [(`RESIL_WINDOW_SIZE-1):0] control;
wire  [(`PROC_VAR_STAGES-1):0] selection;
wire  little_delta_out;
wire  big_delta_out;
wire  [(`MINIMUM_DELAY+`RESIL_WINDOW_SIZE+`PROC_VAR_STAGES):0] n;
wire  invert_mux;

assign n[(`MINIMUM_DELAY+`RESIL_WINDOW_SIZE+`PROC_VAR_STAGES)] =
   little_delta_in;

Inverter_DELTA U00(.out (nbig_delta_in), .in (big_delta_in));

genvar i;
genvar j;
generate
if(`PROC_VAR_STAGES%2==0)
begin : Odd_Mux_Stages
        for(i=0;i<`PROC_VAR_STAGES;i=i+1)
                begin : PROC_VAR_U
                Mux  U(.in1 (n[((2**i)+i)]), .in0 (n[(2**(i+1)+i)]), .s (
                    selection[i]), .out (n[((2**i)-1+i)]));
                if((i%2)==0)
                        begin
                        for(j=((2**i)+i);j<(2**(i+1)+i);j=j+1)
                        begin : Resilient_Window
```

```
                                        PnSDE U(.in_ps (n[(j+1)]), .
                                            n_in_bdelta (nbig_delta_in), .
                                            control (control[(j-(i+1))]),
                                            .out_ps (n[j]), .out_ldelta (
                                            little_delta_out));
                        end
                end
                else
                        begin
                        for(j=((2**i)+i);j<(2**(i+1)+i);j=j+1)
                        begin : Resilient_Window
                                        PSDE U(.in_ps (n[(j+1)]), .
                                            n_in_bdelta (nbig_delta_in), .
                                            control (control[(j-(i+1))]),
                                            .out_ps (n[j]), .out_ldelta (
                                            little_delta_out));
                        end
                end
        end
        assign big_delta_out = n[0];
end
else
begin : Even_Mux_Stages
        for(i=0;i<`PROC_VAR_STAGES;i=i+1)
                begin : PROC_VAR_U
                Mux  U(.in1 (n[((2**i)+i)]), .in0 (n[(2**(i+1)+i)]), .s (
                    selection[i]), .out (n[((2**i)-1+i)]));
                if((i%2)==0)
                        begin
                        for(j=((2**i)+i);j<(2**(i+1)+i);j=j+1)
                        begin : Resilient_Window
                                PSDE U(.in_ps (n[(j+1)]), .n_in_bdelta (
                                    nbig_delta_in), .control (control[(j-(
                                    i+1))]), .out_ps (n[j]), .out_ldelta (
                                    little_delta_out));
                        end
                end
                else
                        begin
                        for(j=((2**i)+i);j<(2**(i+1)+i);j=j+1)
                        begin : Resilient_Window
```

```
                            PnSDE U(.in_ps (n[(j+1)]), .n_in_bdelta (
                                nbig_delta_in), .control (control[(j-(
                                i+1))]), .out_ps (n[j]), .out_ldelta (
                                little_delta_out));
                        end
                end
        end
        Tristate U(.in (n[0]), .out (invert_mux), .en (1'b1));
        assign big_delta_out = invert_mux;
end
endgenerate

generate
        for(i=((2**`PROC_VAR_STAGES)+(`PROC_VAR_STAGES-1));i<(`
            RESIL_WINDOW_SIZE+`PROC_VAR_STAGES);i=i+1)
        begin : Resilient_Window_Complement
                PSDE U(.in_ps (n[(i+1)]), .n_in_bdelta (nbig_delta_in), .
                    control (control[i-(`PROC_VAR_STAGES)]), .out_ps (n[i
                    ]), .out_ldelta (little_delta_out));
        end
endgenerate

generate
for(i=0;i<`MINIMUM_DELAY;i=i+1)
        begin : Static_DE
                Tristate_buffer  U(.in (n[(`RESIL_WINDOW_SIZE+`
                    PROC_VAR_STAGES+i+1)]), .out (n[(`RESIL_WINDOW_SIZE+`
                    PROC_VAR_STAGES+i)]));
        end
endgenerate

endmodule
```

APPENDIX B – PIN PLACEMENT IN COMPLEMENTARY-DE SYNTHESIS

This Appendix presents the pin placement commands used to redistribute pins inside the complementary-DE IP design. It is important that the pin placement around the IP be designed to reduce crosstalk wire interference and to allow future placement over other circuits' power rows. The commands here are for the Innovus Cadence environment. An alternative version for the older Encounter Cadence environment is also available in the design database. Templates and scripts are all available in the svn of the design flow in the link:

https://corfu.pucrs.br/svn/delayline/tags/complementary-de.

The script used here is in the *synthesis/physical_innovus* folder of the design database.

```
set filler_size 0.8
set min_x [lindex [get_db current_design .bbox] {0 0}]
set max_x [lindex [get_db current_design .bbox] {0 2}]
set pin_min_x [expr $min_x+($filler_size/2+0.1)]
set pin_max_x [expr $max_x-($filler_size/2-0.1)]
set number_of_pins 0
foreach_in_collection x [get_ports] { incr number_of_pins}
set design_height [expr [lindex [get_db current_design .bbox] {0 3}] - \
[lindex [get_db current_design .bbox] {0 1}]]
set pins_space [expr ($design_height-2) / (($number_of_pins+1)/2)]
set Right_pins little_delta_out
set Left_pins little_delta_in
lappend Right_pins big_delta_in
lappend Right_pins big_delta_out
set pins_counter 1
set control_pins 0
foreach_in_collection x [get_ports control] {
        if {[expr $pins_counter<($number_of_pins/2)]} {
                lappend Left_pins control[$control_pins]
                incr control_pins
                incr pins_counter
        } else {
                lappend Right_pins control[$control_pins]
                incr control_pins
                incr pins_counter
        }
}
set selection_pins 0
```

```
foreach_in_collection x [get_ports selection] {
        if {[expr $pins_counter<($number_of_pins/2)]} {
                lappend Left_pins selection[$selection_pins]
                incr selection_pins
                incr pins_counter
        } else {
                lappend Right_pins selection[$selection_pins]
                incr selection_pins
                incr pins_counter
        }
}
for {set pins_counter 0} {$pins_counter < [llength $Left_pins]}
{incr pins_counter} {
edit_pin -side inside -assign [list $pin_min_x [expr ($pins_counter+1)*
($pins_space)]] -pin_width 0.1 -pin_depth 0.52 -layer 3 -fixed_pin 1
-pin [lindex $Left_pins $pins_counter]
}
for {set pins_counter 0} {$pins_counter < [llength $Right_pins]}
{incr pins_counter} {
edit_pin -side inside -assign [list $pin_max_x [expr ($pins_counter+1)*
($pins_space)]] -pin_width 0.1 -pin_depth 0.52 -layer 3 -fixed_pin 1
-pin [lindex $Right_pins $pins_counter]
}
gui_fit
```

APPENDIX C – LIST OF PUBLISHED ARTICLES

During the development of this Thesis, five scientific articles were authored or co-authored by the Author. From these articles, one was published on a renowned scientific journal and four in international conferences with a high impact on VLSI research. Among these publications, the Author collaborated with seven different co-authors from Brazil, United States and India, including three Professors, two Ph.D. students, one M.Sc. student and one undergraduate student.

1. Heck, G. and Heck, L. S. and Moreira, M. T. and Moraes, F. G. and Calazans, N. L. V.
   A new local clock generator for globally asynchronous locally synchronous MP-SoCs.
   In: Analog Integrated Circuits and Signal Processing, 89(3), pp. 631-640, 2016.

2. Heck, G.; Heck, L. S.; Singhvi, A.; Moreira, M. T.; Beerel, P. A.; Calazans, N. L. V.
   Analysis and Optimization of Programmable Delay Elements for 2-Phase Bundled-Data Circuits.
   In: 28th International Conference on VLSI Design (VLSID), 2015, Bangalore, India. *Qualified this work*

3. Heck, G.; Heck, L.; Moreira, M. T.; Moraes, F.; Calazans, N. L. V.
   A Digitally Controlled Oscillator for Fine-Grained Local Clock Generators in MP-SoCs.
   In: Latin American Symposium on Circuits & Systems (LASCAS), 2015, Montevideo, Uruguay.

4. Moreira, M. T.; Guazzelli, R. A.; Heck, G.; Calazans, N. L. V.
   Hardening QDI circuits against transient faults using delay-insensitive maxterm synthesis.
   In: 24th edition of the Great Lakes Symposium on VLSI (GLSVLSI), 2014, Houston, TX, USA.

5. Guazzelli, R. A. and Heck, G. and Moreira, M. T. and Calazans, N. L. V.
   Schmitt trigger on output inverters of NCL gates for soft error hardening: Is it enough?
   In: Latin American Test Workshop (LATW), 2014, Fortaleza, CE, Brazil.