Technology-Accurate Variability-Aware Performance Macromodels for On-Chip Communication Synthesis

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor–Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Ing.
Petru Bogdan Bacinschi
geboren am 24. Juli 1981
in Luduş, Rumänien

Referent:	Prof. Dr. Dr. h. c. mult. Manfred Glesner
Korreferentin:	Prof. Dr.-Ing. Anca Manolescu
Korreferent:	Prof. Dr.-Ing. Norbert Wehn
Tag der Einreichung:	2. Juli 2010
Tag der mündlichen Prüfung:	5. November 2010

D17

Darmstadt 2010

To my lovely wife,

Veronica

Acknowledgments

This dissertation is the outcome of my work as a teaching and research assistant at the Institute of Microelectronic Systems, Technische Universität Darmstadt. Many people have contributed in countless ways in making this work possible. I would like to sincerely thank my *Doktorvater*, Prof. Manfred Glesner, for his kind advice and guidance during my doctoral years and for involving me in various teaching activities and research projects funded by several companies and scientific foundations.

I also express my gratitude towards Prof. Anca Manolescu and Prof. Norbert Wehn, who kindly accepted to act as reviewers for this thesis. Their comments and observations have been very valuable for improving the quality of the work. Furthermore, I would like to thank Prof. Udo Schwalke, Prof. Volker Hinrichsen, and Prof. Gerd Balzer for acting as members of the examination committee.

This work could not have been accomplished in a pleasant way without a good atmosphere at the working place. For this, I would like to express my thanks to all colleagues at the institute with whom I had the pleasure of carrying out important research projects, producing reports and papers, sharing various teaching activities, and solving several stringent administrative issues. The friendly help and support of Hans-Peter Keil, Leandro Möller, Sebastian Pankalla, François Philipp, Faizal Samman, Christopher Spies, Pongyupinpanich Surapong, and Ping Zhao permitted me to concentrate on writing the final manuscript and preparing the exam. I would also like to thank my older and former colleagues Prof. Alberto García, Andre Guntoro, Heiko Hinkelmann, Prof. Klaus Hofmann, Prof. Thomas Hollstein, Prof. Leandro Indrusiak, Octavian Mitrea, Massoud Momeni, Tudor Murgan, Oana Mutihac, Oliver Soffke, and Prof. Peter Zipf, who shared, on various occasions, their experience regarding a multitude of issues like writing papers and project proposals, finalizing project reports, as well as scientific and less-scientific practical advices. Further, I am also greatly indebted to my colleagues Roland Brand and Andreas Schmidt for their continuous support in many technical issues. Many thanks also to our friendly secretaries, Silvia Hermann and Iselona Klenk.

Last but not least, I wish to greatly thank my lovely wife Veronica for her great love and support, and also my entire family for all their efforts and for the received education and opportunities.

Abstract

A major challenge in the design of multi-processor systems-on-chip (MPSoCs) is to provide an adequate on-chip communication architecture. Hereby, a series of parameters must be considered, including communication data size, speed, power consumption, and topology, to name only a few. Additionally, variable data flows, as well as increasing process and environmental parameter variations lead to undesired effects, such as reduced yield or increased leakage power levels. The main objective of this thesis is to provide a methodology for the parametrized joint optimization of delay and energy consumption during the communication architecture synthesis, by performing a statistical analysis and optimization of parametric yield under the influence of parameter variations. Moreover, in order to increase the accuracy of the proposed methodology, circuit-level models for the communication activities and technology-accurate models for the interconnection segments are developed.

In order to accurately specify statistical parameter distributions in the application profile and process parameter variations, this thesis develops a complete methodology for variability description and propagation across performance macromodel expressions. For this purpose, a generalized random variable model is developed, capable of representing non-standard estimated distributions using discretized pdfs with adjustable accuracy. Another important contribution represents the development of a propagation method for statistical distributions across the modeling expressions using analytic implementations of the most often used operators as well as the introduction of a fast generalized method for implementing statistical operators with a precision comparable to Monte Carlo at a very small fraction of the execution time. Based upon this methodology, statistical performance macromodels for delay and energy consumption are constructed.

Since the use of different signaling methods has a strong impact on communication performance, a further important contribution is the inclusion of signaling techniques in the communication synthesis in the form of circuit-level communication models. First, a technology-dependent statistical transistor model is derived, which supports variability descriptions for all process-dependent parameters and employs the previously-developed statistical operators to propagate the parameter distributions throughout the model expressions. Furthermore, pulsed current-mode and voltage-mode signaling circuits are analyzed and modeled using the statistical transistor model, equivalent circuit models, and analytic expressions of the current and voltage signals. Within this context, the im-

pact of voltage scaling and body biasing on the circuit performance are also analyzed. Afterwards, the circuit-level models are employed for modeling entire communication segments and the segment models are included within the system-level performance macromodels for the communication synthesis. The accuracy of communication segment models is further enhanced through a wide-bandwidth characterization method for arbitrary interconnect segments. The method relies on an initial set of parameter extractions, designed to reflect the particularities of a given manufacturing process, and applies a sequence of incremental extrapolations to construct the model of a specified segment. Accuracy evaluations show a performance close to industry-standard field simulators.

Finally, synthesis results in the context of delay-driven and energy-driven optimizations show the efficiency of pulsed current-mode signaling on long communication segments and the advantages of voltage-mode signaling on short links. In addition, it is shown that voltage scaling and body biasing can be integrated effectively in the communication synthesis to reduce energy consumption.

Kurzfassung

Eine bedeutende Herausforderung für den Entwurf von Multi-Prozessor-Systems-on-Chip (MPSoCs) ist die Erstellung einer geeigneten On-Chip-Kommunikations-Architektur. Dabei soll eine Reihe von Parametern berücksichtigt werden, wie z.B. Kommunikationsdatenmenge, Geschwindigkeit, Stromverbrauch und Topologie, um nur einige zu nennen. Darüber hinaus führen variable Datenflüsse sowie zunehmende Prozess- und Umgebungsparametervariationen zu unerwünschten Wirkungen, wie einer reduzierten Fertigungsausbeute oder einer erhöhten Verlustleistung. Das Hauptziel dieser Dissertation ist es, eine Methode für die parametrisierte gleichzeitige Optimierung von Verzögerung und Energieverbrauch im Rahmen der Kommunikationssynthese zu entwickeln, die sich durch die Durchführung einer statistischen Analyse und Optimierung der parametrischen Ausbeute unter dem Einfluss von Parametervariationen kennzeichnet. Darüber hinaus werden Schaltungsmodelle für die Kommunikation sowie technologiegenaue Modelle für die Verbindungssegmente entwickelt, um die Genauigkeit der vorgeschlagenen Methode zu erhöhen.

Um statistische Parameter-Distributionen in dem Anwendungsprofil sowie Prozessparametervariationen genau spezifizieren zu können, wird in dieser Dissertation eine integrierte Methode für die Beschreibung und Übertragung der Variabilität durch Modellgleichungen entwickelt. Zu diesem Zweck wird ein allgemeines Zufallsvariablenmodell entwickelt, das nicht-standardverteilte Distributionen mittels diskretisierter Dichtefunktionen mit einstellbarer Genauigkeit beschreiben kann. Weitere wichtige Beiträge stellen die Entwicklung einer Methode zur Übertragung statistischer Verteilungen durch Modellgleichungen mittels analytischer Implementierungen der am häufigsten verwendeten Operatoren sowie die Einführung einer allgemeinen Methode für die Umsetzung schneller statistischer Operatoren mit Monte-Carlo-ähnlicher Genauigkeit dar. Basierend auf dieser Methode werden statistische Makromodelle für die Verzögerung und den Energieverbrauch erstellt.

Da die Verwendung verschiedener Signalübertragungsmethoden einen wichtigen Einfluss auf die Kommunikationsleistung hat, stellt ein weiterer wichtiger Beitrag die Integration der Signalübertragungstechniken in der Kommunikationssynthese als Kommunikationsmodelle auf Schaltungsebene dar. Zunächst wird ein technologieabhängiges statistisches Transistormodell abgeleitet, das Variabilitätsbeschreibungen für alle Prozessparameter unterstützt und die zuvor entwickelten statistischen Operatoren verwen-

det. Darüber hinaus werden Signaltreiberschaltungen im gepulsten Strom-Modus und Spannung-Modus analysiert. Diese werden mit Hilfe des entworfenen statistischen Transistormodells sowie der Ersatzschaltungsmodelle und analytischer Ausdrücke der Strom- und Spannungssignale modelliert. In diesem Zusammenhang werden die Auswirkungen der Spannungsskalierung und des "Body Biasing" (Substratvorspannung) auf das Schaltungsverhalten analysiert. Anschließend werden die Schaltungsmodelle für die Modellierung gesamter Kommunikationssegmente eingesetzt und die Segmentmodelle werden innerhalb der Makromodelle für die Kommunikationsynthese auf Systemebene verwendet. Die Genauigkeit der Modelle für Kommunikationssegmente wird weiter durch eine breitbandige Charakterisierungsmethode für arbiträre Leiterbahnsegmente verbessert. Die Methode basiert auf einer Reihe von Parameterextraktionen, welche die Besonderheiten des spezifischen Herstellungsprozesses abbilden. Nachfolgend wird hierauf basierend und unter Durchführung inkrementeller Extrapolationen ein Modell für ein ausgewähltes Kommunikationssegment erstellt. Genauigkeitsanalysen zeigen, dass die so erzielte Modellgenauigkeit nahe an Ergebnissen liegt, die mit branchenüblichen Feldsimulatoren erreicht werden können.

Schließlich zeigen Syntheseergebnisse, die für Verzögerung oder Energieverbrauch optimiert sind, die Effizienz der gepulsten Strom-Modus-Signalübertragung auf langen Kommunikationssegmenten, sowie die Vorteile der Spannung-Modus-Signalübertragung für kurze Verbindungen. Darüber hinaus wird gezeigt, dass Spannungsskalierung und Body Bias wirksam in der Kommunikationssynthese eingesetzt werden können, um den Energieverbrauch zu senken.

Table of Contents

List of Abbreviations

ABB	Adaptive Body Biasing
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instruction-Set Processor
BSIM	Berkeley Short-channel IGFET Model
CAD	Computer-Aided Design
CD	Critical Dimension
CDF	Cumulative Distribution Function
CMOS	Complementary Metal Oxide Semiconductor
CMP	Chemical-Mechanical Polishing
CN	Communication Node
CPU	Central Processing Unit
CSF	Communication Speed Flexibility
CSP	Communicating Sequential Process
D2D	Die-to-Die (Parameter Variations)
DAG	Directed Acyclic Graph
DIBL	Drain Induced Barrier Lowering
DOF	Depth of Focus
DSM	Deep Sub-Micron
DSP	Digital Signal Processing
FBB	Forward Body Biasing
FCT	Floorplan Cluster Tree
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
FSM	Finite-State Machine
GPP	General Purpose Processor
HDL	Hardware Description Language
IGFET	Insulated-Gate Field-Effect Transistor
ITRS	International Technology Roadmap for Semiconductors
IP	Intellectual Property
LDD	Lightly Doped Drain
LER	Line Edge Roughness

MC	Monte Carlo
MOS	Metal Oxide Semiconductor
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
MPSoC	Multiprocessor System-on-Chip
NDF	Neighboring Density Factor
NMOS	N-Type MOS
NoC	Network-on-Chip
NRMSE	Normalized RMSE
ODE	Ordinary Differential Equation
OM	Order of Magnitude
PE	Processing Element
PCA	Principal Component Analysis
PCM	Pulsed Current Mode
pdf	Probability Density Function
PM	Performance Macromodel
PMN	PM Node
PMOS	P-Type MOS
PN	Processing Node
PSK	Phase-Shift Keying
PSM	Program-State Machine
PTM	Predictive Technology Model
PWL	Piece-Wise Linear
RBB	Reverse Body Biasing
RDF	Random Dopant Fluctuations
RMS	Root Mean Square
RMSE	RMS Error
RSF	Response Surface Function
RSM	Response Surface Methodology
RT	Resource Type
RTA	Rapid Thermal Annealing
RTL	Register Transfer Level
RV	Random Variable
SA	Simulated Annealing
SoC	System-on-Chip
SPICE	Simulation Program with Integrated Circuit Emphasis
STA	Static Timing Analysis
TG	Task Graph
UML	Unified Modeling Language
VHDL	Very-High-Speed Integrated Circuit Hardware Description Language
VM	Voltage Mode

List of Tables

List of Figures

Chapter 1

Introduction and Overview

Contents

1.1 Motivation

The steady increase in performance requirements for embedded systems coupled with the ability to integrate more transistors per unit area with every new technological node have lead to the concept of system-on-chip (SoC), an integrated version of the classical embedded system architecture. Furthermore, the current trend to maximize the execution parallelism of general purpose processors by increasing the number of integrated processing cores has been adopted also by the SoC architectures. Consequently, heterogeneous multi-processor systems-on-chip (MPSoCs) are currently the architecture of choice for implementing complex consumer applications, such as high definition television receivers, mobile communication platforms, and video game consoles, and their usage is thus increasing. While many-core architectures are advertising significant performance boosts for parallel data-heavy applications, particularly in the case of heterogeneous implementations the inter-core communication is likely to become a bottleneck and lead to the saturation of performance increase with the number of processing elements. This way, the design of an adequate communication architecture for many cores and for a variable number of running applications is becoming one of the paramount design concerns for MPSoCs. Hereby, a series of constraints must be considered, such as communication needs, minimum performance level, power budget, area, required yield, to name only a few.

Furthermore, the workload on each communication segment may have significant time fluctuations, caused by multiple applications possibly sharing the same architec-

ture, resulting into variable data flows which must be transferred at the given parameters. Apart from data flow variations, process as well as environmental parameter deviations including temperature changes (hot-spots) and supply voltage variations may no longer be neglected, since they increasingly lead to undesirable effects, such as reduced yield and higher power dissipation. Particularly the increasing level of intra-die variations [20, 94] exhibits a stronger impact on the total circuit delay and leakage variations [82] with every new technology node.

The main objective of this thesis is to provide a methodology for the parametrized joint optimization of delay and energy consumption at the system level during the communication architecture synthesis, by performing a statistical analysis and optimization of the parametric yield under the influence of parameter variations. The automated synthesis of communication architectures has received recently a significant attention from the design community [156], with the focus on both performance and power optimization, however the inclusion of parameter variations, a rigorous statistical analysis using arbitrary non-normal variability models together with algebraic operations on random variables, the optimization of parametric yield, and technology accuracy through the use of circuit-level models represent significant novel approaches.

1.2 Research Objectives

The goal of the present thesis is to provide an integrated methodology for the modeling and optimization of on-chip communication synthesis, with an emphasis on parameter variability and technology accuracy. For this purpose, a set of system-level statistical delay and energy macromodels are developed, which employ accurate circuit-level models for the communication structures. The developed macromodels are then employed to explore the synthesis and optimization of on-chip communication architectures.

For the efficient characterization of application requirements, a profiling interface is defined which allows the specification of application-relevant data, such as processing tasks and communication loads, and of architecture and technology parameters, such as MPSoC resources and technology parameters. Moreover, in order to efficiently characterize non-Gaussian parameter variations, a random variable model with adjustable accuracy is developed, which relies on the discrete representation of probability density functions. Together with this model, a set of statistical operators are developed, including an analytic implementation of a statistical product operator and fast numerical implementations with adjustable precision for any other statistical algebraic operation. Upon this statistical method for the propagation of discrete pdf representations across algebraic expressions, a set of variability-aware macromodels for delay and energy consumption are developed. The performance macromodel structures embed statistical operation nodes which store locally a discrete pdf representation of the computed result. Next, the developed macromodels are employed to optimize the mapping and scheduling of processing

tasks on the MPSoC resources with respect to a desired parametric yield extracted from the performance distributions using quantile functions.

An accurate modeling of the communication activities is achieved by developing circuit-level models for the on-chip communication links. First, a technology-accurate statistical transistor model is developed using BSIM4 equations, CMOS process parameters, and the statistical methodology developed previously. This current-source transistor model is then employed to develop circuit-level models for pulsed current-mode and voltage-mode signaling circuits. Since the choice of different signaling methods as well as voltage scaling and body biasing have a significant influence on the on-chip communication performance, these methods are applied to the developed circuit models and analyzed. Further, the circuit-level models are employed in the modeling of on-chip communication segments and the corresponding communication activities. It is important to note that this thesis does not focus on a particular communication architecture, such as hierarchical on-chip buses or networks-on-chip, but rather uses the concept of communication segment to represent an on-chip communication link.

For accurate representations of on-chip interconnection segments and for validating the synthesized architecture, a computationally-efficient wide-bandwidth characterization method is developed. The method defines a set of initial parameter extractions for characterizing the CMOS manufacturing process, followed by on-demand multistep extrapolations for modeling a given interconnection segment with specified wire length, wire widths, spacings, metal layer, and neighboring routing information.

1.3 Thesis Outline

This thesis is organized in three main parts. First, an introductory part presents the motivation, problem formulation, fundamentals, and current challenges in on-chip communication synthesis. After that, the core of the thesis contains the main contributions in the areas of variability-aware performance macromodels, circuit-level modeling of communication structures, and technology-accurate characterization of interconnection segments. At the end, the thesis summarizes the proposed methodology in an application context and presents several concluding remarks.

Part I Chapter 2 presents the most important aspects which must be considered in the design of on-chip communication architectures. Within this context, the concepts of delay and power macromodels are detailed and several modeling approaches are discussed. In addition, the importance of statistical modeling combined with process accuracy for performance estimations of state-of-the-art silicon implementations is emphasized. Several statistical methods to analyze parameter variations are examined and their drawbacks are indicated. Moreover, the shortcomings of several modeling approximations and of the underlying transistor-level models are

evidenced. Finally, additional resources at the circuit level which can be applied in the optimization of communication architectures are illustrated.

Part II Chapters 3, 4, and 5 represent the main contributions of this work. Starting from the need to accurately specify statistical parameter distributions in the application profile and for process parameter variations, chapter 3 develops a complete methodology for the variability propagation across performance macromodel expressions. For this purpose, a generalized random variable model is developed, capable of representing non-standard estimated distributions using discretized pdfs with adjustable accuracy. Another important contribution is the development of a propagation method for statistical distributions across the modeling expressions using analytic implementations of the most often used operators and introducing a fast generalized method for implementing statistical operators with a precision comparable to Monte Carlo at a very small fraction of the execution time. Based upon this methodology, statistical performance macromodels for delay and energy consumption are constructed. Since the use of different signaling methods has a strong impact on communication performance, chapter 4 brings an important contribution to the inclusion of signaling techniques in communication synthesis frameworks in the form of circuit-level communication models. First, a technology-dependent statistical transistor model is derived, which supports variability descriptions for all process-dependent parameters and employs the statistical operators developed in the previous chapter to propagate the parameter distributions throughout the model expressions. Furthermore, pulsed current-mode and voltage-mode signaling circuits are analyzed and modeled using the statistical transistor model, which is dependent on process and environmental variations. Within this context, the impact of voltage scaling and body biasing on the circuit performance are also analyzed. Afterwards, the circuit-level models are employed for modeling entire communication segments and the segment models are included into the system-level performance macromodels employed in the communication synthesis. The accuracy of communication segment models is further enhanced in chapter 5, which introduces a computationally-efficient wide-bandwidth characterization method for arbitrary interconnect segments. The method relies on an initial set of parameter extractions, designed to reflect the particularities of a given manufacturing process, and applies a sequence of incremental extrapolations to obtain the n-port model of a specified segment. Accuracy evaluations show a performance close to industry-standard field simulators.

Part III The results of applying the developed methodology in the context of a practical example are analyzed in chapter 6. The choice of communication segments, signaling methods, supply voltage, and body bias are presented and discussed for optimization scenarios oriented on delay or energy minimization. The accuracy achieved by the modeling framework is again investigated for the synthesized communication segments. Finally, chapter 7 summarizes the thesis and identifies possible directions

for future enhancements.

Chapter 2

Fundamentals and Challenges of Accurate Communication Synthesis

Contents

Embedded systems represent an increasingly ubiquitous presence in our lives since almost two decades. Recently, complex consumer applications, such as high definition television sets, video games, and state-of-the-art video encoding/decoding systems, are mostly integrated on large heterogeneous multiprocessor systems-on-chip (MPSoCs) [144]. Due to the inherent complexity of MPSoC architectures and of current manufacturing processes, the design of such systems represents a particularly challenging task, from the perspectives of chip-level optimization and on-chip communication design. In essence, embedded system design involves an accurate functional and architectural specification, followed by the optimized mapping of the application on the target architecture. Within this context, it is important to notice the lack of a complete *de facto* automated design methodology or tool, that assists the designers during the complete design of MPSoCs, from the initial specifications to the tape-out submission of the chip layout, and which is still accurate for the current manufacturing technologies. First observed by Gajski in [64] for embedded system designs, the aforementioned remark is still valid today, mostly because the architectural and manufacturing challenges scaled up with the advances in the research and development of CAD tools. Nevertheless, a substantial number of approaches and methods have been developed to tackle different important aspects in the design process. In this chapter we enumerate the most relevant methodologies for the on-chip communication synthesis and point out their key challenges. From this perspective, the work described in this thesis fits into the global set of design methodologies and addresses several of the paramount challenges, such as parameter variability and technology accuracy.

Sec. 2.1 discusses the abstraction of application behavior, architectural description, and performance estimation, required for design space explorations. Further, Sec. 2.2 presents the concepts of delay and power macromodels, together with the importance of statistical modeling combined with process accuracy. Sec. 2.3 enumerates several scheduling techniques and points out the importance of scheduling decisions on the overall latency and energy consumption. The important challenge posed by parameter variations is discussed in Sec. 2.4, where several approaches to the modeling of variability are analyzed. Moreover, the importance of technology accuracy in yield estimations and the underlying transistor models is evidenced in Sec. 2.5. Finally, Sec. 2.6 discusses additional optimization resources for communication synthesis available at the circuit level.

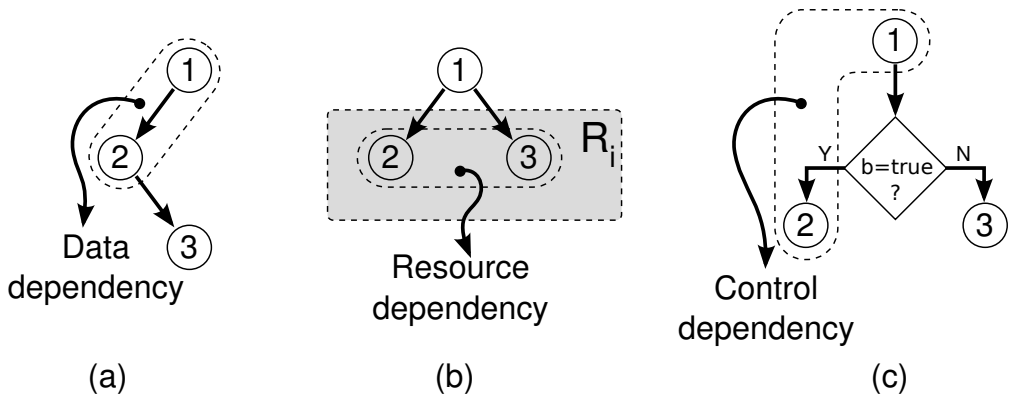

Fig. 2.1: Task dependencies represented as data flow graphs.

2.1 Application Profile and Design Space Exploration

The overall system design begins with a functional specification, followed by mapping the individual tasks on the target physical architecture. This step includes a behavioral specification of the running application and an architectural description of the available IP resources. In addition, the design constraints and the parametric description of task-resource mappings are extracted and specified. The following step consists of creating the necessary performance models for the required performance metrics and at the desired abstraction levels. Finally, the best communication synthesis is identified through exploration of various implementation alternatives and estimation of the corresponding performance values.

2.1.1 Behavioral Specification

Describing the desired system functionality means usually creating a behavioral model of the system, using a high-level description language, such as Matlab/Simulink, UML, Verilog, VHDL, or SystemC, to name only a few. This coarse functional description can be already validated through simulation or formal verification methods. It is to be noted that this initial system description is usually architecture-independent, therefore it can be performed before gathering any knowledge concerning the target resources. Several models are available for describing system functionality [62], such as finite-state machine (FSM), communicating sequential processes (CSP) [77], program-state machine (PSM) [63], Petri nets, flowcharts, UML models etc. Throughout this thesis, the preferred data structure is a data flow graph, or task graph for representing the system behavior. Particularly, we employ an extended version of the task graph, to represent concurrencies of the allocated resources, communication activity, and the different types of task dependencies. Fig. 2.1(a) shows for instance a simple data flow dependency between the tasks. In Fig. 2.1(b), although independent from a data transfer viewpoint, the two tasks are constrained to run sequentially on the resource R_i. A control dependency can be seen as a particular case

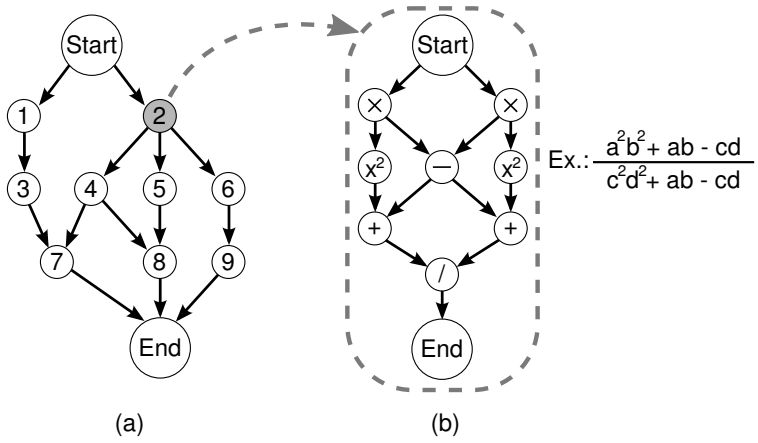

Fig. 2.2: Task graph (a) and refined processing node representation at the operation level (b).

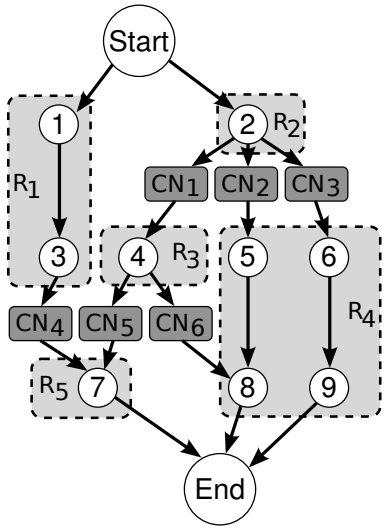

Fig. 2.3: Extended task graph representation showing IP resources R_i and the inter-resource communication nodes CN_i.

of a data flow dependency with an additional control condition (i.e. a control variable check), as shown in Fig. 2.1(c).

Depending on the selection of a task's granularity level, the processing nodes in the task graph can represent either entire jobs, processes, individual statements, or operations. In this work, a coarser representation for the processing tasks is chosen, which simplifies the partitioning and enables a faster design space exploration. Since the main focus of the thesis is on the communication synthesis, a more fine-grained representation of the inter-resource channels is provided, including driving circuits and interconnect segments. Given these considerations, the hierarchical representation of the system functionality can be captured in a task dependency graph, as shown in Fig. 2.2(a). Different granularities of the processing node representations can be adopted, such as more detailed operation-level dependencies (illustrated in Fig. 2.2(b)) [156].

Computational tasks in the context of a task graph (TG) are denoted in this thesis as

processing nodes (PNs), while communication operations are denoted rather as communication nodes (CNs). Hereby, a communication operation represents an inter-resource communication, where tasks (PNs) assigned to different IP resources need to communicate and require therefore an inter-resource communication channel. Based on these considerations, an extended task graph includes both the PNs assigned to the available resources and the required inter-resource communication nodes. Such an extended TG segment is shown in Fig. 2.3.

2.1.2 Architectural Description and Design Constraints

The description of the target subsystem comprises the enumeration of available IP resources and their characteristics. Such IP blocks would include e.g. application-specific integrated circuits (ASICs), application-specific instruction-set processors (ASIPs), parallel processors, digital signal processors (DSPs), microcontrollers, microprocessors, general-purpose programmable microprocessors, larger mega-cores such as microprocessor+ASIC combinations, and other specialized pre-designed logic blocks such as memories, arbiters, multipliers, FFT units, interfaces etc. Next, a selected set of such physical resources is allocated for the system implementation and represents the target architecture. This resource set is then described in terms of the performance constraints of each resource (e.g. timing information, dynamic and leakage power dissipation values etc.) and other implementation details, such as the block area in a given technology.

It is to be mentioned that this work focuses on the communication synthesis, therefore IP-level implementations for task processing blocks, such as the synthesis of a behavioral-level description into an ASIC hardware implementation, or the automatic software generation for a given microprocessor are beyond our scope. Hence, for the scope of communication synthesis, a limited description of the IP resources is employed. Particularly interesting details include power, delay (execution time), and communication load values. For these purpose, performance values related to the execution of tasks are described by parametrized functions, either specified (by the IP provider) or extracted within a profiling procedure.

In order to extract the parametric description for the target architecture, the behavioral model of the application is simulated with language-specific tools and using dynamic profiling tools. This procedure is used to determine branch probabilities, possible execution paths, number of calls to specific operations, and the execution time on the target architecture (e.g. estimated in cycles). It has been pointed out that very accurate estimations can be obtained in this profiling step if the instruction set and a corresponding compiler for the target processor are available [70]. For hardware logic circuits, this parametric description can be obtained from a coarse synthesis of the blocks, estimation of the number of logic gates, and by employing the timing and power metrics for the envisaged technology from logic cell libraries. It is to be noted that particularly the extraction of communication loads between processing tasks is of utmost importance for

the communication synthesis.

Typically, the result of the profiling analysis consists of minimum, maximum, and average estimations of performance metrics. Alternatively, multiple samples of these estimations can be collected and used to build discretized probability density functions and obtain a more insightful characterization. Another option is to approximate the profiling results with given standard distribution types. The latter two description methods are employed in this work for specifying application profiles.

After a thorough insight concerning the achievable performance metrics of the target architecture is obtained, the design constraints for the communication synthesis can be specified. Factors such as overall die area and layout design rules are dictated by cost restrictions and the chosen technology. On the opposite, delay and power budgets are derived from the application requirements. Chip area constraints are mainly used during floorplanning and routing, but also to determine spatial correlations of process parameters across the die. In contrast, the total delay and power budgets are relevant for almost every of the following design steps, including task-resource mapping, scheduling, communication (signaling) resource allocation, and circuit-level voltage optimizations.

2.1.3 Performance Model Creation

Fast and accurate evaluations of various performance metrics for a given system configuration are necessary to find a design solution which satisfies all the constraints and which is optimized according to a given objective. To evaluate the performance metrics we use performance models, which, in essence, are parametrized expressions built to estimate a given metric. A continuous trade-off between speed and accuracy dictates the development and improvement of performance models. On the one hand, fast estimations are critical to keep the design space exploration and optimization feasible. On the other hand, technology accuracy is becoming extremely important with state-of-the-art manufacturing processes where parameter variations exhibit a significant influence on the yield [20]. In the particular case of communication synthesis, accurate models for the communication segments are required. For efficiently estimating the overall system performance only coarse estimations are necessary for the processing tasks while they run on the allocated resources. Such estimations can be represented in the form of statistical descriptions of the execution times and power dissipation levels, extracted from simulation and profiling.

A performance model represents a set of data structures and expressions which symbolically describe a given performance metric [156]. Depending on the particular metric (delay, dynamic power, leakage etc.) and modeling complexity, several performance models for system-level estimations have been proposed. A typical deterministic performance model for delay estimations is shown in Fig. 2.4 in the form of a maximum operation followed by a sum. Here, we denote with T_i^s the earliest starting time allowed for processing node i, T_i^e is its ending time, and T_i^x is its execution time, always specified

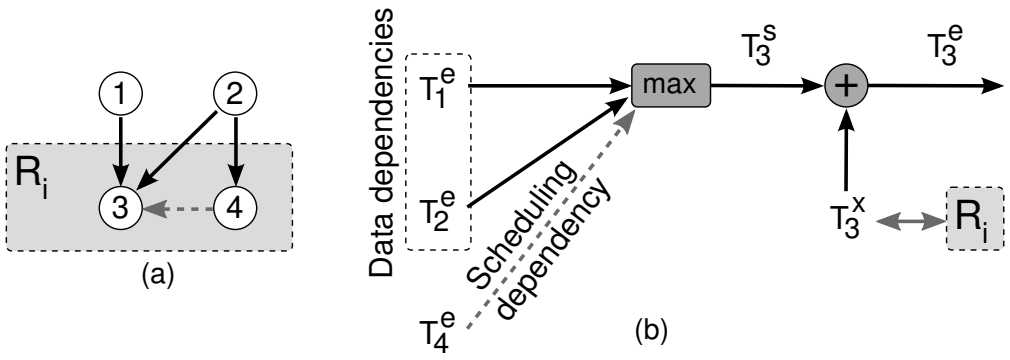

Fig. 2.4: Section from a task graph with four processing nodes (a) and the resulting deterministic delay model for processing node 3 (b).

in the context of given resource on which the node is running. The maximum operation expresses the condition for the earliest starting time, which depends on the finishing of all previous PNs which have data or scheduling dependencies with the current node. Finally, the sum operation adds the execution time to the starting time to determine the earliest end time of the node. A more detailed overview on performance modeling approaches is offered in Sec. 2.2. The main contributions of this thesis in the field of performance modeling are described in detail in chapters 3 and 4.

2.1.4 Estimation and Optimization

Exploration in the implementation space implies the evaluation of possible design alternatives, given by various configurations of the set of interconnected resources, each of them implementing sections of the behavioral specification. At every step in the design space exploration, an estimation of the design quality is performed, using performance models and applying a cost function on the estimated performance metrics. The acceptability of a particular design depends on the given constraints and on the particular figure of merit chosen as objective function to be optimized. Hereby, the exploration speed and efficiency are directly determined by the granularity and accuracy of the employed performance models. Here, the involved computational effort and the resulting total time for the entire design space exploration may be substantial with respect to the other design steps. Note that, the choice and implementation of the exploration and optimization algorithm has also a strong influence on the overall effort.

At this point in the design process, the physical resources are allocated for the different tasks. Usually, variables are stored into memory blocks, behaviors are implemented by processors, and the communication tasks are attributed to inter-resource channel segments. This partitioning step is characterized by the granularity of the structural objects. This way, we can apply the resource mapping at gate level, block level, core level etc. Since the focus lies on the communication, we apply the mapping of processing tasks at the coarser core and block levels. A fine-grained circuit-level representation is employed

for the communication structures, considering also the interconnect-related delay and power dissipation in the equivalent circuits.

Further, the performance metrics to be considered during the optimization must be defined. Examples include the overall execution time (application delay, or latency), dynamic power dissipation, and, particularly important for recent technologies, leakage power. The next step is to combine the considered metrics into a cost (or objective) function which best represents the figure of merit for optimization. Finally, an optimization algorithm is applied during the exploration. Examples include mixed-integer linear programming [132], greedy priority-driven clustering [51], list scheduling [57], iterative-improvement methods, such as simulated annealing and tabu search [156, 128], genetic algorithms [25], and custom heuristic methods [59]. Nonetheless, the largest challenge for the design optimization step remains the characterization of the optimum solution and the certainty of achieving a global optimum.

2.2 Performance Modeling

The communication synthesis paradigm states that a communication architecture must be found which is optimal for all the applications running on the designed System-on-Chip [156]. From this point of view, a series of difficulties must be considered. First, the communication architecture is unique and must be adequate for all intended applications. This implies that all the possible data flow variations must be taken into account during the optimization. Second, important information about the routing path, segment length, and interconnect parasitics is not available at early design stages. Hence, there is a substantial need for new algorithms and modeling approaches which are able to efficiently predict the segment length and accurately estimate line parasitics. Further, to obtain an architecture which satisfies all performance constraints for the given applications, accurate performance models for the relevant metrics (e.g. latency, power) are required.

2.2.1 Performance Macromodel Concept

A performance macromodel (PM) represents a symbolical description of the system which allows the estimation of performance attributes, such as delay, dynamic power, or leakage, considering a particular system configuration during design space explorations. The term macromodel indicates an overall system-wide model, resulted from the composition of several smaller resource-level or task-level models. Typical representations of performance macromodels include analytical model expressions, numerical equations, and graphical representations implemented as linked data structures and operations.

Highly-flexible performance models for latency and power, which can be easily extended and refined for new requirements, have been proposed in [156, 56, 155]. Here, latency models are developed as symbolic representations of the timing values for the

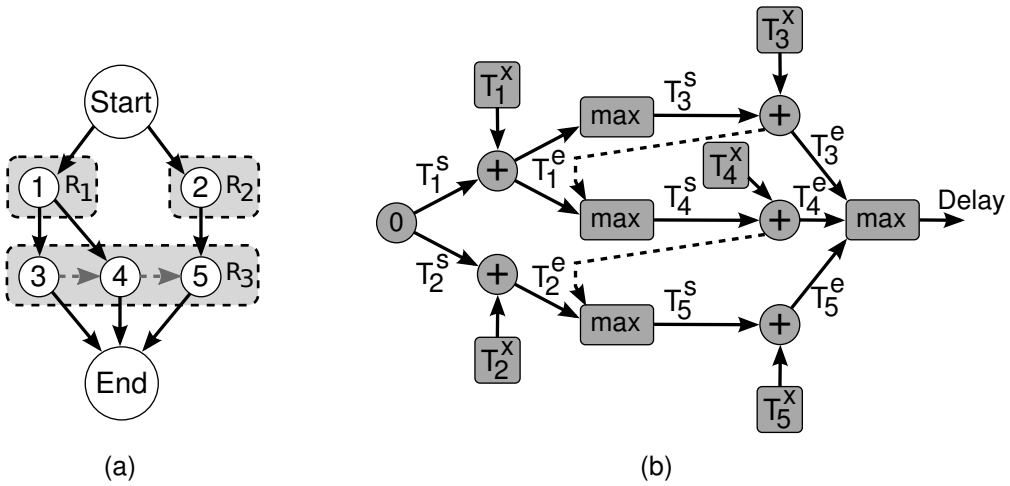

Fig. 2.5: Task graph example (a) and the attached delay PM (b).

start, execution, and end times of processing and communication nodes. Within this representation, communication nodes consist of alternating sequences of data packets and synchronization nodes for handshaking.

A graph representation of a PM for delay, directly derived from a task graph as proposed in [156] is shown in Fig. 2.5. The start node in the delay PM is set to the deterministic value of 0, which represents the initial timing value for the delay computation. Next, the main part of the macromodel consists of nodes representing symbolic variables and operations linked by directed arcs. Finally, the additional dashed links represent scheduling dependencies (in the example from Fig. 2.5 it has been assumed that PNs 3, 4, and 5 are scheduled on resource R_3 in this order).

The numeric estimation of the modeled performance attribute occurs by evaluating all the operational nodes. Further model refinements and extensions could add to this basic structure also other operations, such as multiplication, minimum, division, square root etc. Typically, operation nodes represent either fundamental computations in the macromodel, such as the sum of dissipated power by multiple resource units, or performance constraints, such as wait conditions for the end times of all predecessors.

Additional structures may be added to the macromodel as a result of design decisions during exploration, such as adding a new communication segment which inserts an additional latency between two processing nodes. Changing task-resource mappings or altering the scheduling sequence has also an impact on the PM structure. It is important to note, that the performance macromodel definition is very general and can be adapted for many performance metrics. In addition, model flexibility is an important requirement, to allow for extensions, refinements, and adding new relationships between the modeled attribute and design changes.

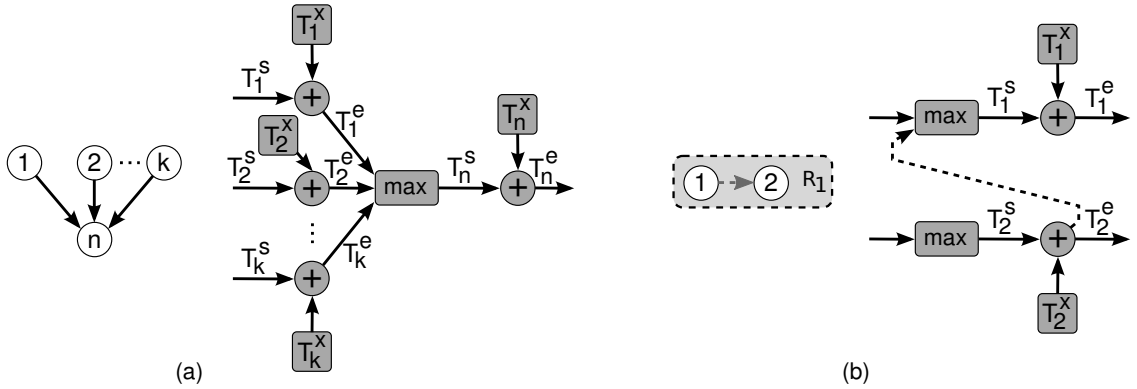

Fig. 2.6: Modeling of data (a) and scheduling (b) dependencies (after [156]).

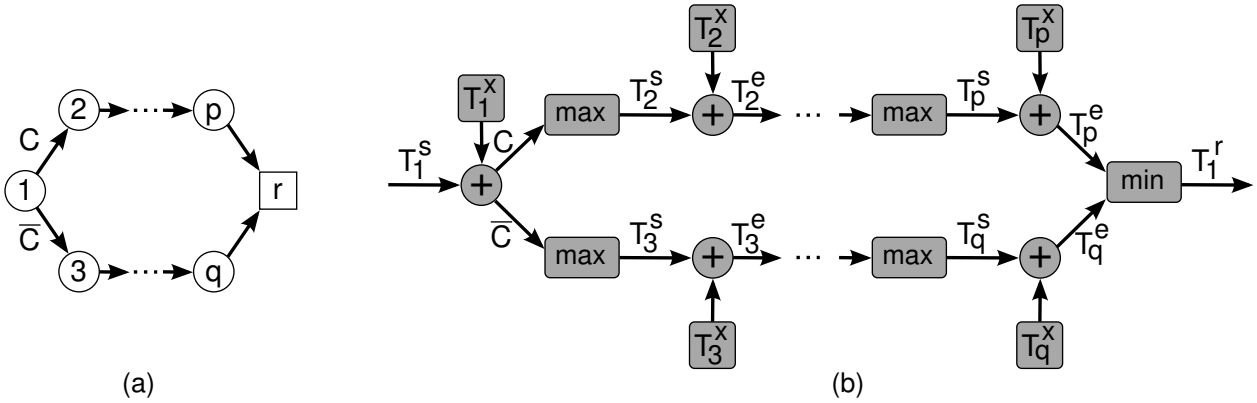

Fig. 2.7: Control dependencies in the task graph (a) and in the delay macromodel (b) (after [56]).

2.2.2 Delay Macromodels

A delay macromodel example has been shown in Fig. 2.5, whereas the general rules for expressing data and scheduling dependencies are depicted in Fig. 2.6(a) and (b). As mentioned before, data dependencies add PM links between the maximum node of a given PN and the end times of all its predecessors, whereas scheduling dependencies are represented by additional dashed links between otherwise independent tasks which are assigned to the same resource.

Control dependencies can also be included in the macromodel [56], where the flow of processing nodes is influenced by conditional branches. Such an example is illustrated in Fig. 2.7(a), where after the execution of PN 1 a boolean condition C is tested. If condition C is found true, then PNs $2 \ldots p$ are executed. Otherwise, the execution flow is directed towards PNs $3 \ldots q$. The end of the conditional branch, where the possible execution flows rejoin, is symbolically marked by inserting the artificial node r. The corresponding representation of the control dependency in the delay macromodel is shown in Fig. 2.7(b). If condition C is true, the upper branch of the macromodel is evaluated in the usual way, while in the lower branch, the link marked by the false $\overline{C}$ condition propagates the value *infinite*. This way, at the rejoining node r, a minimum operation ensures that the false

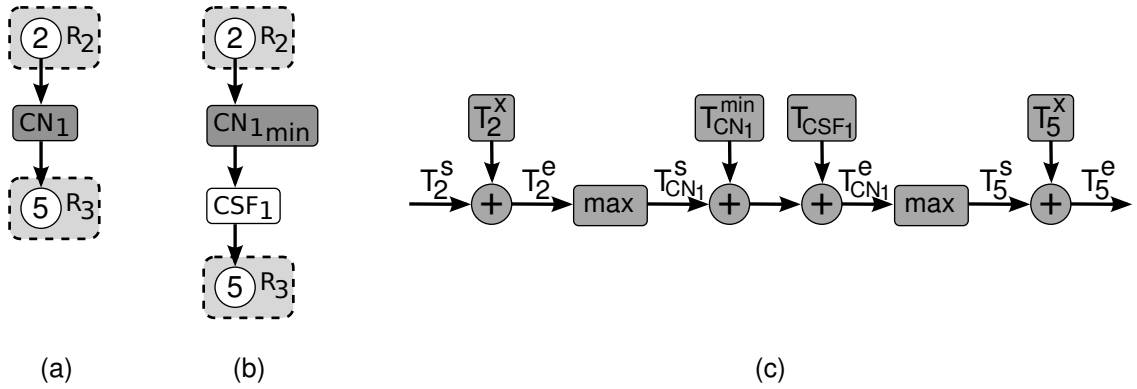

Fig. 2.8: Insertion of communication speed flexibility nodes in the task graph (a,b) and the corresponding delay macromodel structure (c) (after [155]).

condition branch is discarded. Reversely, if C is false, then the upper branch propagates the infinite value and the lower $\overline{C}$ branch is executed normally.

Given the above formulations, the end time of a given processing node i is evaluated as:

$$T_i^e = \max_{\substack{p=j\ldots k \\ s=l\ldots m}} \left(T_p^e, T_s^e\right) + T_i^x \tag{2.1}$$

where $p = j \ldots k$ iterates all the predecessors of PN i in the task graph (data dependencies), and $s = k \ldots l$ iterates all the scheduling dependencies. If PN i is followed by a conditional branch, the output T_i^e is also marked by the corresponding condition, and the computation becomes:

$$T_i^e = \begin{cases} \max\limits_{\substack{p=j\ldots k \\ s=l\ldots m}} \left(T_p^e, T_s^e\right) + T_i^x & , C_i = true \\ \infty & , C_i = false \end{cases} \tag{2.2}$$

according to the value of the branch condition C_i.

A method for estimating the execution time of communication tasks has been proposed in [155] in the form of communication speed flexibility (CSF). The concept is illustrated in Fig. 2.8 for a given communication node CN_1. In Fig. 2.8(b), the CN is replaced with a minimum CN ($CN_{1_{min}}$) followed by a CSF node. First, the minimum CN represents the shortest delay which can be achieved for the given communication load, in the target technology, with a minimum-length communication segment. Hence, the minimum CN introduces the absolute minimum achievable communication latency, independent of the floorplanning and routing information which is not available before synthesizing the complete communication architecture. Next, the CSF node inserts a delay equal to the maximum tolerable latency on the respective segment, which does not violate the total system delay constraint. This value can be evaluated and updated during the optimization, for each system configuration, by comparing the total delay with the delay constraint and computing the allowed slacks on each communication segment. The added minimum CN and CSF delays are inserted in the delay PM as shown in Fig. 2.8(c). Note

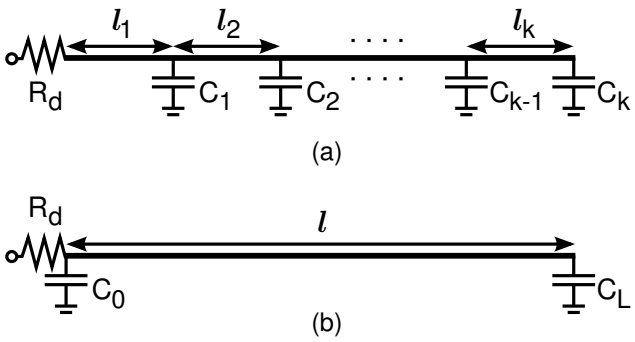

Fig. 2.9: Transformation of a multiple-pin net (a) into a two-pin net (b) (after [49]).

that, the sum of the minimum CN delay and the CSF represents the maximum communication delay tolerable on a given segment. As explained in [155], this maximum tolerable delays become constraints for the subsequent communication synthesis step. A bus segment which exceeds the maximum tolerable delay will also violate the overall system latency constraint.

An alternative approach in the form of a delay estimation engine has been employed in [130] for bus delay modeling. Within this framework, the bus wire lengths are computed after a simulated annealing-based floorplanning using the half-perimeter of the minimum bounding box which encloses the bus connections [32]. Given the estimated length l, wire delays are computed according to [49] as:

$$T = R_d C_0 + \left(\frac{\alpha_1 l}{W^2(\alpha_2 l)} + 2\frac{\alpha_1 l}{W(\alpha_2 l)} + R_d c_f + \sqrt{R_d r c_a c_f l} \right) \cdot l \tag{2.3}$$

where $\alpha_1 = \frac{rc_a}{4}$, $\alpha_2 = \frac{1}{2}\sqrt{\frac{rc_a}{R_d C_L}}$, $W(x)$ is Euler's Lambert function [29] defined as $W(x) = \{w | we^w = x\}$, R_d is the driver's on resistance, r is the sheet resistance ($\Omega/\square$), c_a is the unit area capacitance (fF/μm^2), and c_f is the unit effective-fringing capacitance (fF/μm), whereas C_0 and C_L are lumped capacitances computed for equivalence in terms of the Elmore delay [60, 136, 49] as shown in Fig. 2.9 and given by:

$$C_L = \sum_{j=1}^{k} \frac{\sum_{i=1}^{j} l_i}{l} \cdot C_j \tag{2.4}$$

$$C_0 = \sum_{j=1}^{k} C_j - C_L \tag{2.5}$$

Note that, the delay estimated by (2.3) is assuming an optimal wire sizing as indicated in [49] with an average wire width of:

$$w = \sqrt{\frac{r(c_f l + 2C_L)}{2R_d c_a}} \cdot l \tag{2.6}$$

An alternative approach to performance modeling has been recently proposed in [14] where commercial and academic standalone CAD tools are embedded into an open framework and employed for performance estimations. Such tools typically embed complete

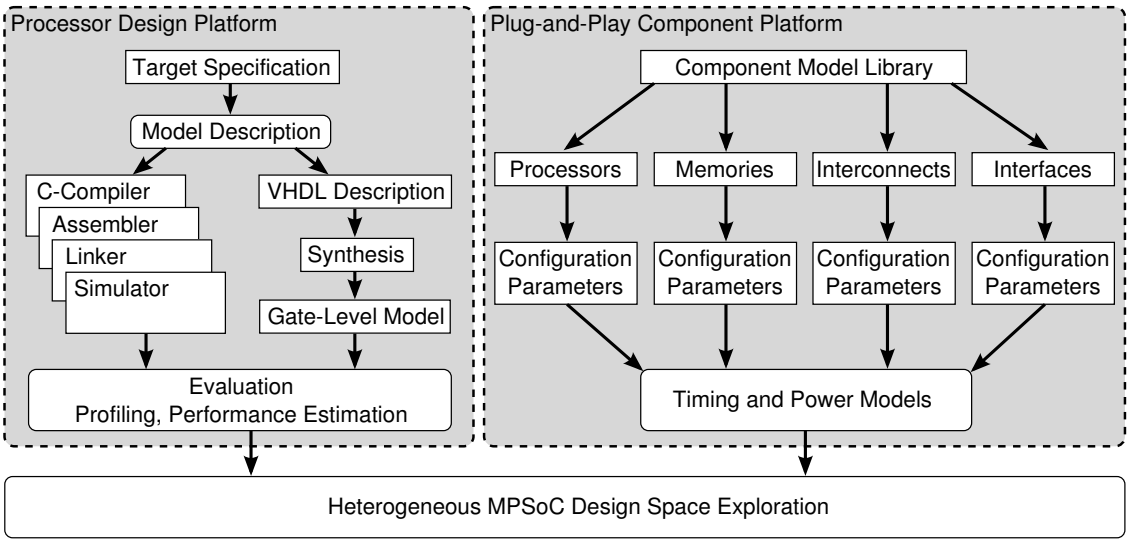

Fig. 2.10: Open framework with embedded CAD tools for performance modeling and design exploration, as proposed in [14].

design flows for processors, with high-level modeling platforms, instruction-set simulators, C-compilers, assemblers, and linkers. Moreover, academic tools such as MPARM [102], include collections of component models, like processors, interconnects, memories, and dedicated interfaces, together with simulation engines. After selecting the required components from a library, further configuration options are available for each element to allow a better adaptation to the application needs. Timing and power models are also available for the included components, hence the performance metrics can be directly evaluated. Finally, a flexible open framework based on the SystemC language integrates the two platforms and allows the MPSoC design using either plug-and-play library components, or custom-designed processing and logic blocks, as summarized in Fig. 2.10.

2.2.3 Macromodels for Power Estimation

Meeting the power constraint for the entire system is a challenging task, particularly since it can be strongly influenced at every design step, including partitioning and mapping, scheduling, and the communication synthesis. Accurate estimations of the power dissipation are possible at the RTL level for logic blocks and through circuit-level simulations including parasitics for the communication segments. Nevertheless, since early design decisions, such as mapping a task to a particular resource, or sharing a narrower communication segment at the cost of added latency, have a strong influence on the total power consumption, we need to develop and integrate power estimation models also at the higher levels of abstraction.

Early approaches to high-level power estimation rely on simple observations, which are usually proven only after the optimization step, through subsequent simulations or measurements. In [146] a high-level power estimation for processors is used to optimize

the scheduling of running tasks. The optimization relies on bringing the processor into a power-down mode, were all parts except the clock and timer circuits are turned off, and applying a dynamic frequency and voltage scaling while the processor is running. The power estimation relies on the proportionality of the dynamic power consumption with the frequency and with the square of the supply voltage.

The first approach to optimize power consumption during the hardware-software co-synthesis of embedded systems has been published in [50] and includes a power estimation step for CPUs, ASICs, FPGAs, as well as for communication activities. System description includes an average-power vector $\xi(t_i) = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{in}\}$ for each task t_i, where each element ξ_{ij} represents the average dissipated power of task t_i if assigned to processing element j. Here, the average power dissipation is estimated at nominal supply voltage and assuming an average data stream. A similar vector is defined for the peak power, $\kappa(t_i) = \{\kappa_{i1}, \kappa_{i2}, \ldots, \kappa_{in}\}$, where each element κ_{ij} designates the peak power dissipation of t_i if assigned to processing element j. Hereby, the operating conditions for peak power dissipation are assuming the highest supply voltage level and the highest data stream. Idle power levels are also considered, for processing elements (PEs), ASICs, FPGAs, and communication links, estimated for the case in which no task is being executed on the respective resource. It is further assumed that the average and peak idle power consumption levels are specified for each processor. For ASICs and other logic elements, the gate-level power is specified, while for FPGAs, the average and peak idle power levels are also given. Similarly, each communication link is described by the average and peak idle power levels. During the automated system design, processing tasks are organized in clusters, which are then assigned to existing resources. To optimize power, the energy levels of each task are considered during clustering, instead of the task priorities. An example of such a energy-oriented clustering is shown in Fig. 2.11, where the two clusters C_1 and C_2 are formed along the higher-energy paths. The values in brackets represent energy levels, while the numbers in bold type indicate the task priorities. e_1 to e_6 are the inter-task communication edges. Although the main optimization focus lies on power minimization, energy consumption levels are employed to take into account both active-mode and idle power consumptions for resources and communication links in a unified manner.

For a given task t_i (or edge e_i), the average energy consumption is given by:

$$E_i = T^x_{wc_{ij}} \cdot \xi_{ij} \tag{2.7}$$

where $T^x_{wc_i}$ is the worst-case execution time and ξ_{ij} is the average power dissipation, both in conjunction with a given resource j. Here, the use of worst-case execution times is chosen if the real-time constraint precedes the total power constraint, otherwise the normal operating conditions are employed. Furthermore, if task t_i has no child dependencies, its average energy is given by (2.7). Otherwise, the average energy level is computed from every child edge $e(t_i, t_c)$ as:

$$E^t_i = \max_{t_c} \left(E_i + E_{e(i,c)} + E_c \right) \tag{2.8}$$

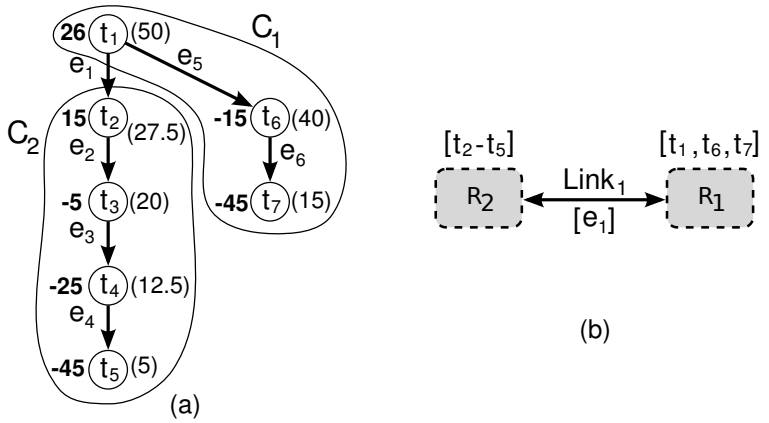

Fig. 2.11: Power-optimized clustering of processing tasks (a) and the corresponding resource mappings and communication link (after [50]).

where E_i is the average energy of task t_i, E_c is the average energy of t_c, and $E_{e(i,c)}$ is the average energy of edge $e\left(t_i, t_c\right)$, all estimated by (2.7).

It is important to note that energy levels must be updated after each clustering. Particularly the communication edges have a different cost (execution time and power dissipation) if they connect two different resources as compared to the case in which they connect tasks assigned to the same processing unit. In this way, the clustering configuration is optimized until it minimizes the average energy level. During the allocation of resources, the peak power dissipation is estimated and checked against the maximum constraint. For processors and communication links, the average and peak power dissipations are estimated according to the processing tasks and communication edges which are running on them. If we denote with $\Re^{\xi}$ the average energy consumption of the resource j and with θ^{ξ} the average idle power dissipation [50], then for each processing resource P and communication link L:

$$\Re^{\xi}\left(P\right) = \left[\sum_{t_i \in T} \xi_{ip} \cdot T_{ip}^{x} \cdot n_i\right] + \left[\theta^{\xi}\left(P\right) \cdot \Psi\left(P\right)\right] \tag{2.9}$$

$$\Re^{\xi}\left(L\right) = \left[\sum_{e_j \in E} \xi_{jl} \cdot T_{jl}^{x} \cdot n_j\right] + \left[\theta^{\xi}\left(L\right) \cdot \Psi\left(L\right)\right] \tag{2.10}$$

where $t_i \in T$ represent the processing tasks running on P, $e_j \in E$ the communication tasks assigned to L, n_i (n_j) is the number of times that t_i (e_j) is running across the total system time period, and Ψ represents the idle time of a given resource or communication link in the system period.

In contrast to processing units, tasks assigned to FPGAs or ASICs can also run in parallel if designed appropriately. As a consequence, the peak power dissipation is computed as the sum of the peak power levels of the tasks running in parallel, followed by a maximum of the tasks (or groups of parallel tasks) which run in series. Finally, the total system power dissipation can be found by dividing the total estimated energy consumption for

all resources and communication links by the total system latency, which is estimated using the delay macromodel.

A performance macromodel for power estimation during the hardware-software co-synthesis of low-power embedded systems has been proposed in [56] and represents the power dissipation at a task and resource level using a linked tree structure with operational nodes. This approach considers in particular the resource mapping and task scheduling decisions, whereas the inter-resource communication aspects are not captured. The power consumption is computed as the sum of the power dissipated by each processing element during task execution and during idle time. A special attention is payed to the option of temporarily shutting down a resource when it's not used by any processing task. An example for a small TG with five tasks is depicted in Fig. 2.12. P_i^a represents the average active-mode power consumption for each resource i and is multiplied by the execution time of each task (computed as the difference $T_j^e - T_j^s$). The total active-mode power consumption is estimated by adding the contributions of each resource and of all assigned tasks. In addition, for each resource, the idle time is computed, as the difference between the starting time of a task T_j^s and the end time of the previous task in the scheduling list T_k^e. In the example from Fig. 2.12, the idle times of resource R_3 are between tasks 3 and 4, and between tasks 4 and 5, thus given by the differences $T_4^s - T_3^e$ and $T_5^s - T_4^e$, respectively. This idle time is then multiplied by the idle-mode power consumption, which for resource R_3 is indicated by P_3^i. If R_3 is shut down between tasks 3 and 4, then this decision is indicated by the symbolic variable S_{34} which is set to "1", otherwise it will be set to "0" indicating that the resource is kept on. Similarly, if R_3 is shut down and restarted between tasks 4 and 5, then S_{45} will be set to "1", otherwise set to "0". It is to be noticed that shutting down and restarting a resource requires an additional power, which is represented by P_i^{stop} and P_i^{start}, respectively. By multiplying the sum of these power amounts with the value of the flags S_{jk}, their contribution is automatically added or ignored to the total power consumption. In a similar way, the idle-mode power amounts are multiplied with the negated shutdown flags, $\overline{S_{jk}}$, to enable their contribution when the respective resource is not turned off. Although not depicted in Fig. 2.12(b), similar idle-mode estimations must be implemented also for the resources R_1 and R_2. For instance, the idle time of R_2 in this small example is given by the difference between the system end time (as given by the output of the delay PM) and the end time of task 2, T_2^e. In this time frame, R_2 can be kept on, where the idle-mode contribution P_2^i multiplied with the idle time will be added, or turned off, where P_2^{stop} must be added. Finally, it is important to notice, that also in this approach, both active-mode and idle-mode power contributions are multiplied by time durations (execution time, respectively idle time sequences). This observation leads to the remark that the estimated metric represents an energy quantity, rather than power. It is also important to remember at this point, that for battery-powered embedded systems, the total energy consumption is of a more critical concern than the heat generated by instant power dissipation. Therefore, estimating energy consumption versus power dissipation appears to be a more appropriate decision in this case. Note also, that the power dissipation can be easily obtained by dividing the

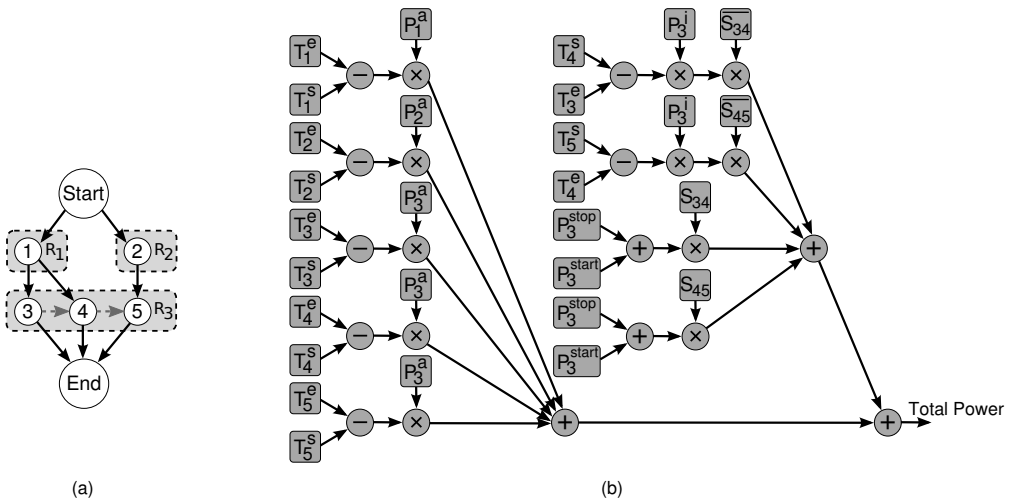

Fig. 2.12: Task graph example (a) and the derived power PM (b) (after [56]).

total energy consumption over a system period by the time obtained from the delay PM.

Given these observations, we can conclude that the total system energy considering shutting down and restarting resources can be estimated as the sum:

$$E_{total} = E_{active} + E_{idle} + E_{restart} \tag{2.11}$$

The active-mode energy contribution is given by:

$$E_{active} = \sum_{R_k \in \{R\}} P_k^a \sum_{PN_j \leftrightarrow R_k} \left(T_j^e - T_j^s\right) \tag{2.12}$$

where $\{R\}$ is the set of resources and $PN_j \leftrightarrow R_k$ denotes the set of PNs assigned to resource R_k. The idle-mode energy contribution is determined by:

$$E_{idle} = \sum_{R_k \in \{R\}} P_k^i \sum_{PN_j \leftrightarrow R_k} \left(T_j^s - T_{j-1}^e\right) \cdot \overline{S_{j-1,j}} \tag{2.13}$$

and the energy consumption for shutting down and restarting the resources can be estimated as:

$$E_{restart} = \sum_{R_k \in \{R\}} \left(P_k^{stop} \cdot T_k^{stop} + P_k^{start} \cdot T_k^{start}\right) \sum_{PN_j \leftrightarrow R_k} S_{j-1,j} \tag{2.14}$$

where T_k^{stop} and T_k^{start} are the time duration required for shutting down and restarting resource R_k and have been added to the model to provide a unified expression for the total energy consumption. It is to be mentioned that the implications concerning the additional latency caused by shutting down and restarting a resource must also be taken into account in the delay PM.

The total idle time of a resource can also be estimated by computing the utilization rate [75]:

$$u_{R_k} = \frac{N_{active}^{R_k}}{N_{total}} \tag{2.15}$$

where $N_{active}^{R_k}$ represents the number of clock cycles in which R_k is active and N_{total} is the total number of cycles for the execution of the complete application. This way, an alternative expression for estimating the idle energy can be written in terms of utilization rates as:

$$E_{idle} = \sum_{R_k \in \{R\}} (1 - u_{R_k}) \cdot P_k^i \cdot T_{total} \tag{2.16}$$

where T_{total} is the total execution time of the application, as given by the delay PM.

As it can be seen, several timing values are required for estimating the power, respectively the energy consumption. Thus, a tight connection must exist between the delay PM, which estimates all the start and end times of the tasks, and the power PM. This connection can be implemented by linking the outputs of several operational nodes from the delay PM which compute e.g. T_j^s and T_j^e to the inputs of operational nodes (mostly subtraction nodes, see Fig. 2.12(b)) from the power PM. Thus, there is a substantial overhead to update the PMs and the inter-PM connection links when iterating through design changes during the system optimization. This aspect is discussed in more detail in chapters 3 and 4. It is also important to add that the inclusion of inter-resource communication activity in the system architecture must be also included in the power estimation. The inclusion of communication links in power PMs is discussed in chapter 4.

2.2.4 Statistical and Process-Accurate Modeling

Increasing variations in process and environmental parameters [20] have brought the need for new approaches to performance modeling, which are both accurate, to include the parasitic effects of very deep sub-micron processes, and statistical in nature, to model the influence of variations. The first efforts towards statistical modeling have been focused on the evolution of static timing analysis (STA) towards statistical STA [24].

One of the first concepts for a statistical performance model for delay estimations has been presented in [142] and derives a method for computing the delay of interconnect lines with two drivers. This method employs an equivalent circuit model which includes the driver output impedance and capacitance, load capacitance at the receiver, and an RC model for the interconnect line. Considering process variations in the interconnect wire width, both resistance and capacitance values are expressed in terms of their dependence on this parameter, as:

$$R = R_{nom} \frac{w}{w + \Delta w} \tag{2.17}$$

$$C = C_{nom} \left(1 + \frac{w}{w - \Delta w}\right) \tag{2.18}$$

where w is the line width and Δw is the corresponding variation. For the interconnect width parameter, a Gaussian distribution is assumed with zero mean and 10% standard deviation. The Elmore delay model is employed to compute the delay at the end of the

line as:

$$d = R_s\left(2C_l + C_1 + C_2\right) + R_1\left(\frac{C_1}{2} + C_2 + C_l\right) + R_2\left(\frac{C_2}{2} + C_l\right) \tag{2.19}$$

where R_s is the output resistance of the driver, C_l are the load capacitances at the input and output of the line, and the interconnect runs over two metal layers, each segment being characterized by R_1, C_1, and R_2, C_2, respectively. In this approach, a simple sensitivity dependence of the delay with respect to a given parameter variation is extracted, in the form of the first derivative. For instance, if only interconnect wire width variations are considered as in [142], the delay expression including the contributions of the parameter variations can be written as:

$$d = d_{nom} + \frac{\partial d}{\partial w_1}\Delta w_1 + \frac{\partial d}{\partial w_2}\Delta w_2 \tag{2.20}$$

where w_1 and w_2 are the wire widths of the line on the two metal layers. The sensitivities can be further analytically extracted from (2.19), as shown here for the dependence on w_1:

$$\frac{\partial d}{\partial w_1} = \left(R_s + \frac{R_1}{2}\right)\frac{\partial C_1}{\partial w_1} + \left(\frac{C_1}{2} + C_2 + C_l\right)\frac{\partial R_1}{\partial w_1} \tag{2.21}$$

Such a simplified sensitivity analysis keeps only the first derivatives of the delay with respect to the variable process parameters. As a result, only the first derivatives are propagated across the model, not the full distributions of the parameter variations. The modeling capability is therefore limited to a first-order approximation, as shown in this expression of the arrival time expanded for the process parameters $p_1, p_2, \ldots, p_N$:

$$A\left(p_1, p_2, \ldots, p_N\right) \approx A\left(p_1^{nom}, p_2^{nom}, \ldots, p_N^{nom}\right) + \sum_{i=1}^{N} a_i \Delta p_{i,a} \tag{2.22}$$

where a_i are the first-order sensitivities (derivatives) of the delay with respect to p_i, and $\Delta p_{i,a}$ is the deviation of process parameter p_i for the net A where the delay is estimated. The model complexity is limited by the extraction of first-order derivatives of the performance metric with respect to every parameter variation considered. In addition, analytic expressions of the sensitivities can be extracted only if the base model includes all the process and environment parameters which are to be taken into account.

Another early approach towards statistical delay modeling has been published in [121], proposing a simplified analytical model for estimating the influence of gate length (L_{gate}) variations to the variance of the overall delay. Within this approach, the MOSFET gate length is represented in terms of the distinct contributions to the variation:

$$L = L_{nom} + L_{prox} + L_{spat} + \varepsilon \tag{2.23}$$

where L_{prox} represents the proximity-dependent variation, L_{spat} is the spatial variation, and ε is the random residual variation. Typically, proximity-dependent sources of variations are treated as systematic, since the frequency and placement of particular gates in

the layout is known. As a result, L_{prox} is modeled by a discrete random variable, while L_{spat} and ε are represented by Gaussian distributions, such as $L_{spat} \sim N\left(0, \sigma^2_{spat}\right)$ and $\varepsilon \sim N\left(0, \sigma^2\right)$. The delay of a gate is estimated using the compact gate delay model [39] and is given by:

$$d = \frac{C_L V_{dd}}{n}\left(I_{dn}^{-1} + I_{dp}^{-1}\right) \tag{2.24}$$

where C_L is the load capacitance, $n = 3.7$, and I_{dn}, I_{dp} are the drain current of the NMOS and PMOS transistors. Further, the drain saturation current is approximated using the following empirical relationship [39]:

$$I_{dsat} \sim L_{eff}^{-0.5} T_{ox}^{-0.8}\left(V_{dd} - V_{th}\right) \tag{2.25}$$

where L_{eff} is the effective gate length, T_{ox} is the oxide thickness, and V_{th} is the threshold voltage. Assuming that $L_{eff} \approx L$ and that $C_L \approx L \cdot W \cdot C_{ox}$ (where W is the channel width and C_{ox} is the oxide capacitance), the dependence of the gate delay on the gate length considering (2.24) is of the form $d \sim L^{1.5}$, or $d = k \cdot L^{1.5}$, where k is a process-dependent constant. The same analysis can be further extended for a path of m gates, where the path delay becomes:

$$D = \sum_{i=1}^{m} d_i = k \sum_{i=1}^{m} L_i^{0.5} L_{i+1} \tag{2.26}$$

where L_i and L_{i+1} are the gate length of the driver and of the load, respectively, for every two successive gates in the path. The analysis then finds the variance of the path delay, which can be computed by propagating the variances of the individual contributions from (2.23) through the delay expression from (2.26). Finally, the standard deviation of the path delay is expressed as:

$$\sigma_D = \sqrt{\mathrm{Var}\left\{D\right\}} = \frac{1.5 D_{nom}}{L_{nom}}\left(\frac{\sigma^2_{L_{prox}} + 0.28\sigma^2}{m} + \sigma^2_{L_{spat}} \cdot \sum_{l=1}^{k} \gamma_l^2\right)^{1/2} \tag{2.27}$$

where k is the number of the spatial partitions crossed by the given path (inside a spatial partition the spatial correlation is considered to be equal to 1), and $\gamma_l = \frac{n_l}{m}$ is the ratio of the number of gates n_l in a given partition l to the number of gates in the path (m).

This method shows the derivation of an analytic expression of the standard deviation of the overall path delay, starting from a simple analytic model of the gate delay. It is to be noticed that only gate length variations are captured in this approach. Additional parameter variations would require a new model derivation, starting from the gate delay, while the complexity of propagating all the variances would eventually become prohibitive. It is also important to add, that only the standard deviation is estimated, i.e. a single parameter is employed to characterize the distribution of the delay.

A more refined approach published in [34] includes the effects of process parameters such as isolation oxide strain, transistor orientation, and etch loading. In addition to process variations, also environmental parameter variations are considered, including supply voltage and temperature. The model derivation relies on the alpha-power law [139]

delay model for estimating the gate delay as [58]:

$$T_d = \frac{C_L \cdot V_{dd}}{I} = \frac{K \cdot V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{2.28}$$

where C_L is the load capacitance, $K = \frac{C_L}{\mu C_{ox}(W/L)}$, and α is a velocity saturation index. Further, a unified analytic expression for the drive current is obtained, in the form of:

$$I \propto \frac{\left\{\ln\left[1+\exp\left(\frac{V_{dd}-V_{th}}{2S}\right)\right]\right\}^2}{\left\{1+\ln\left[1+\exp\left(\frac{V_{dd}-V_{th}}{E_{sat}L}\right)\right]\right\}} \tag{2.29}$$

where S is the subthreshold swing and E_{sat} is the critical electrical field inducing the carrier velocity saturation [167]. Based on this formulation for the drive current, an expression for the gate delay is derived as:

$$T_d = \frac{K_L \cdot V_{dd} \cdot \left\{1+\ln\left[1+\exp\left(\frac{V_{dd}-V_{th}}{E_{sat}L}\right)\right]\right\}}{\left\{\ln\left[1+\exp\left(\frac{V_{dd}-V_{th}}{2S}\right)\right]\right\}} \tag{2.30}$$

where K_L represents a loading parameter and is modeled by a polynomial function:

$$K_L = \left(k_0 + k_1 \cdot L \cdot C_L + k_2 \cdot L^{a_k}\right)/W \tag{2.31}$$

The coefficients k_0, k_1, k_2, and a_k are extracted from simulations and are process-dependent. Based on this delay formulation, a canonical model (linear around the nominal value) is developed for the delay variability. For process parameter variations, statistical Gaussian distributions are assumed, whereas other environment parameters (i.e. temperature, voltage) are described using corner models. The delay variability is extracted in the form of a coefficient of variation, as the ratio of standard deviation to the mean (or nominal) value, and is expressed as:

$$\frac{\sigma_{T_d}}{T_d} = \sqrt{\left(\frac{\partial \ln T_d}{\partial L}\right)^2 \cdot \sigma_L^2 + \left(\frac{\partial \ln T_d}{\partial V_{th}}\right)^2 \cdot \sigma_{V_{th}}^2} \tag{2.32}$$

Again, this modeling approach extracts a single parameter to characterize the delay variability, namely the coefficient of variation, hence a more detailed characterization of the delay distribution is not offered. In addition, the only process and environment parameters which are captured are the ones present in the nominal gate delay expression from (2.30). Thus, for the inclusion of other parameter variations, a new delay model must be developed.

One of the most recently-published modeling methods in this field [149] derives analytic expressions for computing the delay and leakage of a digital gate considering inter-die and intra-die process parameter variations and their spatial correlations. In this approach, only variations in the gate length and the zero-bias threshold voltage V_{th0} are

considered. The analysis relies on the same sensitivity-based modeling of performance metrics with respect to parameter variations. Considering e.g. p different process parameters, with their respective variations ΔP, the sensitivity-based expressions for delay and leakage are given by:

$$Delay = d_{nom} + \sum_{i=1}^{p} \alpha_i \left(\Delta P_i\right) \tag{2.33}$$

$$Leakage = \exp\left(V_{nom} + \sum_{i=1}^{p} \beta_i \left(\Delta P_i\right)\right) \tag{2.34}$$

where d_{nom} is the delay nominal value, $\exp\left(V_{nom}\right)$ is the nominal value of the leakage power, α_i is the delay sensitivity with respect to process parameter P_i, and β_i are the corresponding sensitivities of the logarithm of leakage. These sensitivities must be extracted during circuit simulations while varying the corresponding process parameters. Spatial correlations between the parameters are captured using a 2-D grid across the die area and a principal component analysis (PCA) method is applied for enabling their representation using de-correlated standard normal distributions.

This method can be extended with additional process parameters, however at the expense of running additional simulations to extract the sensitivities. A big challenge here is the inter-parameter correlation. If only one parameter P_i is varied at a time and the corresponding sensitivity α_i (or β_i) is extracted, this sensitivity might also embed a significant dependence of the delay (or leakage) on another parameter P_j, assuming that P_j is correlated with P_i. Later, when P_j is varied, another sensitivity α_j (or β_j) is obtained. Since both α_i and α_j embed a common amount of the dependence on P_j, errors are introduced when summing up $\alpha_i \Delta P_i + \alpha_j \Delta P_j$. Such linear sensitivity models give accurate results only under the assumption of independence between the process parameters. Furthermore, this method stores the extracted sensitivities as 2-D lookup tables, indexed by the input rise time and the output capacitance. Nevertheless, besides the dependencies of the sensitivities on the input slew and load capacitance, there are many other possible dependencies, including dependencies on voltage levels, body bias, temperature-related dependencies, which are not captured by default. For any additional dependency a new extraction of the sensitivities and a corresponding storage form is required. Finally, at given circuit conditions, a lookup must be performed for all these parameters, to obtain the matching sensitivity, and if not found, possible interpolations must be considered, which would add further questions regarding interpolation method, induced errors etc.

To summarize, several research efforts have focused on performance modeling methods considering parameter variations, with significant results. One of the main limitations to their applicability consists of the difficulty to extend them to include additional process parameters. This is either not possible without reformulating the underlying delay model [142,121,34], or, at best, implying a time-consuming extraction of new sensitivities [149]. Here it should be also mentioned that numeric extractions of sensitivities from circuit simulations are always possible, nevertheless, as pointed out before, this approach

rises important questions concerning inter-parameter correlations.

Besides extension-related restrictions, the modeling capability of these methods is mostly limited. On the one hand, methods like [121, 34] employ a characterization of statistical distributions using a single parameter, such as the variance, standard deviation, or the variation coefficient. On the other hand, sensitivity-based approaches such as [142, 149] allow only a simple linear modeling of the dependence on parameter variations. The propagation of variation distributions from the parameter level up to the performance metric is therefore limited by this simplified representation. Furthermore, the extraction and storage of sensitivities has a limited ability to capture all the influence of external varying factors, such as changes in voltages and temperature.

A practical method for including all state-of-the-art process parameters in the model and for propagating their entire distributions across the analytic expressions has not been published yet. The main challenges for such an approach are mainly the model complexity (to include all the parameters) and the development of an extensive set of practical statistical operators which operate on distributions and propagate a complete representation of the variation, such as e.g. the probability density function. This thesis offers an important contribution in this area by providing an extensive modeling approach relying on the BSIM transistor model and the required statistical operators for propagating the entire distributions across the performance macromodels. The inter-parameter correlation issue is solved by employing a detailed analytic model at the transistor level, which embeds all process parameters, without recurring to sensitivity analyses. The spatial correlation of process parameters and temperature is also considered by employing a PCA-based de-correlation method.

Technology accuracy represents a key factor in the overall model precision, particularly for deep sub-micron processes. In the methods presented so far, the accuracy is limited by the simplified current, delay, and leakage models. Relatively-recent modeling approaches [34] employ simplified empirical transistor models developed two decades ago [139]. This thesis presents two modeling methods which are focused on technology accuracy at the circuit level (chapter 4) and at the interconnect level (chapter 5).

2.3 Resource Scheduling

Scheduling of processing and communication tasks on the assigned resources (processing elements and communication segments) determines their relative execution order, hence the start and finish time for each task. It has been shown in Sec. 2.1.3 and 2.2 that scheduling dependencies play an important role in performance model creation.

While assigning a scheduling sequence, it is important to consider all data and control dependencies between the tasks. In addition, the scheduling algorithm must ensure that no task is left waiting in the execution queue, i.e. that no task is “starving”. Usually a performance metric is also optimized during the scheduling method, such as minimum

execution time, or lowest power consumption. Since both scheduling and resource mapping are known to be NP-complete problems [66], heuristic approaches which do not guarantee optimality are often employed.

In this section a few representative scheduling methods are presented in the context of system-level optimization and communication architecture synthesis. Additionally, their implications in the performance model structure and the overall design of an optimized communication are shown.

2.3.1 Preemptive Methods

Within preemptive scheduling algorithms, the scheduler interrupts running tasks during their execution and switched the context to other tasks, typically on a priority-driven basis. Interrupted tasks continue their execution eventually, after restoring their state from a saved context. Preemptive scheduling is not always preferred in embedded systems, since it involves a substantial overhead: active scheduler, context switching, state saving etc. Nonetheless, depending on the particular application and task dependencies, it can improve the execution efficiency. For instance, if tasks are waiting frequently for external events, during so-called "busy-wait" states, preemption would allow other tasks to execute, thus leading to an overall shorter system latency.

A preemptive, static scheduling algorithm has been proposed in [50,51] which assigns priority levels to tasks based on a dynamic finishing time (T_i^e) estimation for delay minimization. Starting from the observation that a task can be reused by multiple functions, an optimized resource mapping can be performed. Furthermore, depending on the degree of reusage of a task, its preemption suitability is determined. To take into account the scheduler overhead during task preemption, such as state saving and context switches, an preemption overhead parameter is defined and employed for delay and power estimations. The finish times of each task and of the overall system are finally compared with the specified deadlines to validate scheduling decisions.

Processing and communication tasks are characterized by execution priority levels and are first sorted in the decreasing priority order. Hereby, tasks with equal priorities are sorted in the increasing order of their execution times. Since communication links can provide more than one segment, both sequential and concurrent communication activities are considered. The preemption decision for a task t_i with priority ϕ_i which runs on a given resource r in favor of another task t_j with priority ϕ_j is given by:

$$t_i \leftarrow t_j \quad \text{if} \quad \begin{cases} \phi_j > \phi_i \\ \text{or :} \\ T_i^b + \eta_r + T_{jr}^x \leq \mu\left(t_i\right) \end{cases} \tag{2.35}$$

where T_i^b is the best-case finish time of task t_i, T_{jr}^x is the execution time of t_j on resource r, and $\mu\left(t_i\right)$ is the required deadline for t_i. The parameter η_r is the preemption overhead

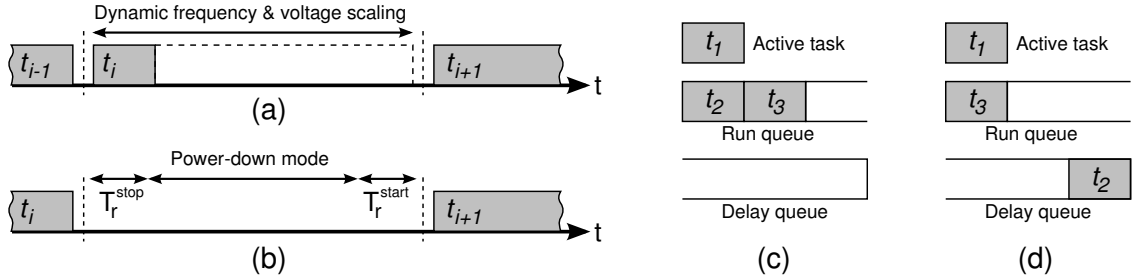

Fig. 2.13: Low power preemptive scheduling with fixed priority (after [146]).

of resource r, which includes context switching and state saving. The second branch in (2.35) indicates that the preemption occurs only if t_i is a sink task which will not miss its deadline. In static preemptive scheduling approaches, after a task is scheduled on a resource, the exact start and end times T_i^s and T_i^e can be precisely evaluated considering the preemption slots. Thus, the accuracy of system delay estimations depends on the modeling and integration of task scheduling.

A fixed priority preemptive scheduling for optimizing power efficiency has been developed in [146], which achieves power reduction by utilizing the slack times between the running tasks. These idle timing intervals are inherently caused by inter-task dependencies, such as in the case of tasks waiting for the input from other resources, but are also generated by variations in the actual execution times of tasks due to real-time conditions. A run-time mechanism identifies and utilizes these slacks for power-reduction mechanisms, such as switching the resource to a power-down mode, or applying a frequency and voltage scaling.

In real-time systems, the execution time of several tasks may vary due to several dynamic conditions, such as different resource allocation, computational load of a processing core, voltage drops etc. The worst-case execution time $T_{wc_i}^x$ can be estimated through profiling, or static analysis, as explained in [98]. Starting from the observation that the ratio of the best-case to the worst-case execution time can vary up to as much as one order of magnitude in typical applications [61], and that several idle time slots are present in systems with fixed priority scheduling, a substantial power reduction can be achieved by dynamically scaling the speed or shutting down the resource if the idle slot is sufficiently large. Thus, during the scheduling process, if only one task is eligible and its actual execution time T_i^x is smaller than the available slot S_i, a frequency and voltage scaling is applied according to the timing difference $S_i - T_i^x$, as shown in Fig. 2.13(a). If no task is scheduled in the time slot, then the resource is switched into a power-down mode, whereas the additional latencies required for stopping and restarting the resource must also be considered, as illustrated in Fig 2.13(b). For this purpose, a conventional scheduler with fixed priority is modified to support these mechanisms.

Fixed-priority scheduling algorithms ensure that each task meets its deadline constraints and are simple to implement. Typically, higher priorities are assigned for tasks with shorter execution times or higher execution frequencies. Lower-priority tasks are

preempted when higher-priority tasks request the execution. The scheduling mechanism proposed in [146] maintains two task queues. A run queue contains tasks waiting for execution, ordered by priority, while a delay queue maintains the tasks already finished, which are waiting to start again in the next system period. The delay queue is periodically searched, and high-priority tasks are moved into the run queue. Fig. 2.13(c) shows task t_1 being executed, while the lower-priority tasks t_2 and t_3 wait for their turn in the run queue. In Fig. 2.13(d) task t_1 requests again execution and task t_3, which was running at the moment, is preempted and moved back into the run queue. If the run queue is empty and no task is active, then the resource is powered down. Otherwise, if an active task is running, then speed scaling is applied.

2.3.2 Non-Preemptive Methods

Non-preemptive scheduling methods are usually preferred in embedded system design, because of their low overhead. Very simple methods, such as first-match first-serve, which select the first compatible task from the execution queue, are often employed on heterogeneous MPSoC platforms [144]. A basic extension to this method is to select the task which benefits the most from the instruction set or the hardware architecture of the respective core. This decision may be detrimental for very simple tasks, which will be constantly pushed back in the queue. Adding a priority level to each task would prevent starving conditions.

A static, non-preemptive deterministic scheduling mechanism has been presented in [59], in combination with a communication optimization algorithm for minimizing the worst-case system latency. Here, the particularities of the underlying architecture and the parameters of the communication protocol are considered for the generation of static scheduling tables for both data processing and communication tasks. Both data and control dependencies are taken into consideration, therefore the actual execution path in the task graph is not predictable at the design time. Fig. 2.14 shows an example where processing nodes PN_2, PN_{11}, and PN_{12} have control dependencies with their successors. The values of the branch conditions C, D, and K are determined only at run-time. Nevertheless, at a certain time moment in the execution, all the upstream conditions are known. Thus, the scheduling algorithm must take the best decision possible considering the known conditions. For this reason, the optimized static scheduling is determined through a heuristic algorithm, which minimizes the worst-case delay for all possible combinations of the branch conditions.

The scheduling table has one row for each processing and communication node (PN_i and CN_i, respectively), and contains the start times T_i^s for the different values of the branch conditions. The resulting scheduling table for the example in Fig. 2.14 is displayed in Tab. 2.1. The entry for a node in a given column corresponds to its scheduling time for the case in which the condition at the top of the column is true. It can be noted, that the start times for PN_1, PN_2, PN_{11}, and CN_1 do not depend on any branch condition, hence

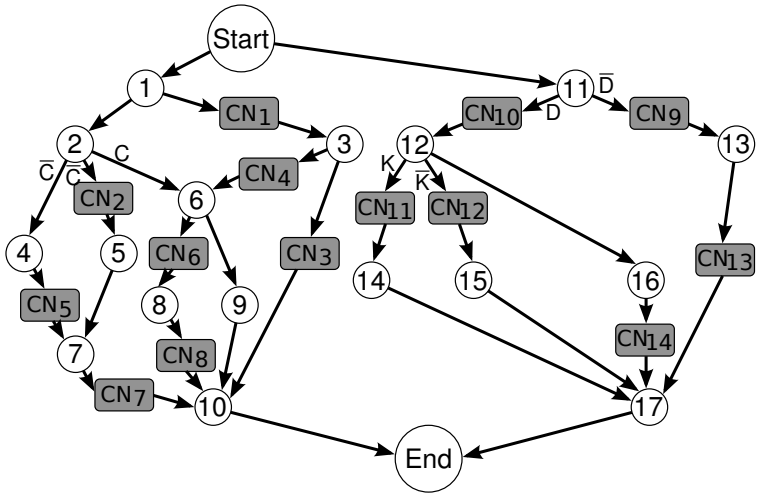

Fig. 2.14: Extended task graph example for static scheduling (after [59]).

they have unconditional deterministic start times (given in the first column).

The worst-case delay is determined by examining the scheduling rows for the terminal tasks. If the terminal tasks are PN_{t1}, PN_{t2},...,PN_{tn}, then the worst-case delay is given by:

$$T_{wc} = \max_{i=1...n} \left\{ \max_{j=1...N_c} \left\{ T^s_{ti,j} \right\} + T^x_{ti} \right\} \tag{2.36}$$

where N_c is the number of possible conditions in the schedule table, and $T^s_{ti,j}$ is the scheduled start time of PN_{ti} in the j-th column. Equation (2.36) represents also the cost function to be optimized by the heuristic scheduling algorithm. In the example from Fig. 2.14 the terminal tasks are PN_{10} and PN_{17}, thus, according to Tab. 2.1, (2.36) becomes:

$$T_{wc} = \max \left\{ 35 + T^x_{10}, 30 + T^x_{17} \right\} \tag{2.37}$$

Another non-preemptive list-scheduling algorithm with fixed priority has been published in [57] for tasks interacting through both data and control dependencies. In general, list-scheduling approaches select the task with the highest priority in terms of the optimization goal (e.g. shortest global execution time). To avoid execution conflicts, a single priority is usually assigned to each task and an individual scheduling table is built for every processing unit. Due to the fact that some nodes interact through control dependencies, an execution trace is identified for each combination of the branch conditions. Consequently, each conditional trace is scheduled independently as an alternative execution path. Within this process, the scheduling algorithm, which relies on heuristics to find close-to-optimal schedules, selects the nodes according to the worst-case combination of the branch conditions. This operation results into scheduling sequences determined by the most time-constrained traces. To cope with the complexity given by considering multiple traces, a set of priorities is assigned to each node, with a priority for each execution trace to which it belongs.

Typically, the priority function includes information regarding the relative position of a node in the critical path. If a node has many successors along the critical path, its

	$true$	D	$D \wedge C$	$D \wedge C \wedge \overline{K}$	$D \wedge C \wedge K$	$D \wedge \overline{C}$	$D \wedge \overline{C} \wedge \overline{K}$	$D \wedge \overline{C} \wedge K$	$\overline{D}$	$\overline{D} \wedge C$	$\overline{D} \wedge \overline{C}$
PN_1	0										
PN_2	3										
PN_3		6							6		
PN_4						7					7
PN_5							18	18			18
PN_6				21	20					20	
PN_7							21	21			21
PN_8				29	28					28	
PN_9				26	25					25	
PN_{10}				35	34		27	26		34	26
PN_{11}	0										
PN_{12}			9			9					
PN_{13}										10	13
PN_{14}					18			24			
PN_{15}				19			24				
PN_{16}				15	15		15	15			
PN_{17}				25	24		30	26		24	24
CN_1	3										
CN_2						9					8
CN_3				21	20		21	20		20	20
CN_4				19	18					18	
CN_5						12					13
CN_6				26	25					25	
CN_7							25	24			24
CN_8				32	32					32	
CN_9										8	11
CN_{10}			8			8					
CN_{11}					16			16			
CN_{12}				16			16				
CN_{13}										22	22
CN_{14}				23	22		23	22			

Tab. 2.1: Scheduling table for the example in Fig.2.14 [59].

execution must finish as early as possible, hence it receives a high priority. This approach leads to critical-path-driven schedulings, as shown for a four-task example in Fig. 2.15(b). However, tasks running on software processors and communication segments must be serialized and including this information in the scheduling algorithm could lead to better scheduling results, as shown in Fig. 2.15(c).

A general limitation of the existing scheduling approaches is the lack of support for variability in the execution times. All the operations applied for scheduling decisions are

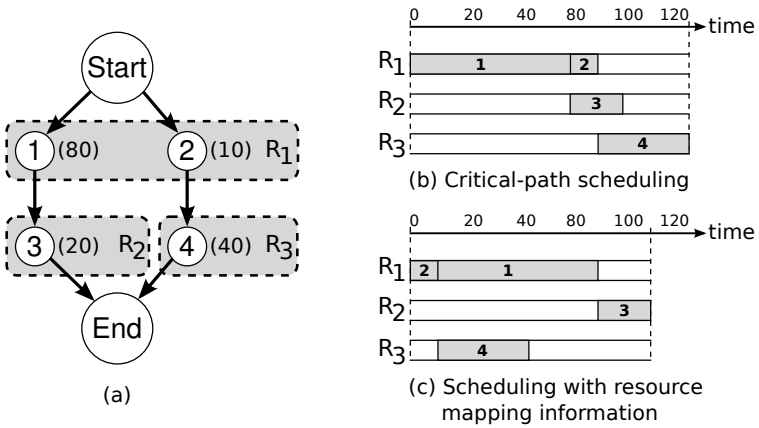

Fig. 2.15: Scheduling of four tasks (a) considering only critical-path information (b) and after including the resource mapping (c) (after [57]).

Tech. Node	90 nm	65 nm	45 nm	32 nm	22 nm
L_{eff} (3σ nm)	4.7	2.6	1.9	1.3	0.9
V_{Tn} ($3\sigma_{D2D}$)	12%	12%	12%	12%	12%
V_{Tp} ($3\sigma_{D2D}$)	6%	6%	6%	6%	6%
T_{ox} (3σ)	4%	4%	4%	4%	4%

Tab. 2.2: Predicted three-sigma variations of device parameters across several technology nodes.

performed on constant numbers and the simple algebraic operators are not suitable for statistical evaluations. Chapter 3 presents a statistical methodology which operates on timing quantities described by statistical distributions.

2.4 Parameter Variations and Statistical Analysis

As CMOS technologies continue to scale in the deep-submicron regime, an increasingly higher level of systematic and random variations in process, supply voltage, and temperature continuously affects the performance of integrated circuits [27, 20, 33]. Process-induced fluctuations result into significant variations of various device parameters, such as L_{eff}, T_{ox}, and V_T [118]. The sources of such variations are either environmental or physical factors. Tab. 2.2 shows predicted values from literature for variations in L_{eff} [80, 133, 34, 173, 95, 33], V_T [133, 34, 7, 20], and T_{ox} [133, 20]. It is to be mentioned that die-to-die (D2D) V_T variations differ between NMOS and PMOS devices as shown in [83] and intra-die variations (not shown in Tab. 2.2) are inversely-proportional to the square root of the channel area [20]:

$$\sigma_{V_T}^{ID} = 3.19 \cdot 10^{-8} \left(\frac{T_{ox} N_A^{0.4}}{\sqrt{L_{eff} W_{eff}}} \, [\mathrm{V}] \right) \tag{2.38}$$

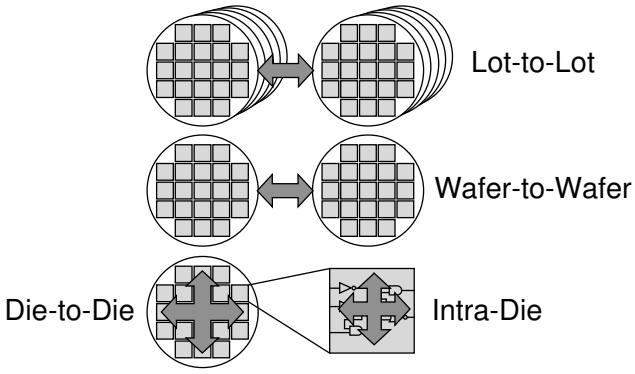

Fig. 2.16: Classification of parameter variations.

where N_A is the average channel doping. The intra-die V_T variations include also the effect of uncorrelated channel doping fluctuations.

Given this trend in device scaling for achieving an enhanced performance and cost reductions, the employed materials and manufacturing processes have come very close to their reliability limits [108]. Thus, an important step towards improving the manufacturing yield is to understand the sources of parameter variations [141] and to develop a statistical methodology for accurate performance modeling.

The inclusion of process parameter variations in modeling methodologies has started with the use of process corners. Hereby, an analysis is performed multiple times, for every process condition. If a sufficiently-high number of process conditions is considered, the effects of process variations on a particular performance metric can be estimated. This approach is however limited by several important aspects. First, this method has been developed to model die-to-die parameter shifts, whereas intra-die variations can not be accurately captured [24, 7]. Second, the number of process parameters which show significant variations has increased substantially with the new process generations [20], and the corresponding number of process corner files required to capture the entire parameter set has reached a number which prohibits the direct analysis using standard Monte Carlo approaches [143]. Effective modeling approaches for including various parameter variations have been developed lately in the field of statistical timing analysis [24]. In the remaining of this section the sources of parameter variations are discussed, followed by a brief review of the developed methods for statistical analysis.

2.4.1 Sources of Parameter Variations

There are several classifications of parameter variations available in the literature, depending on their type [24, 20], source [173, 7, 141, 108], and effects [118, 27, 107]. Generally, parameter variations are investigated for:

- manufacturing process parameters – which typically describe devices and interconnects;

- environmental parameters – including the temperature, operating voltages, as well as wear-out influences across the circuit lifetime.

Variations in process parameters are further classified into:

- systematic, predictable variations – which depend on layout characteristics and can be theoretically modeled using deterministic factors, such as the surrounding layout topology [165];
- random variations – for parameters that are unknown at design time, and for which only statistical descriptions are available.

Nonsystematic, random variations are caused by slight deviations in the manufacturing steps, such as chemical-mechanical polishing (CMP) [96], rapid thermal annealing (RTA) [9] etc. These random variations affect differently the manufactured devices and can be classified in:

- die-to-die (D2D) variations – which have the same value for all the devices on a single die, resulting from the process changes from lot to lot, wafer to wafer, and reticle to reticle;
- intra-die variations – which have a different impact on each device across the same die.

Finally, intra-die variations can be differentiated according to their correlation as:

- spatially-correlated variations – which exhibit a gradual change between different locations on the die;
- uncorrelated variations – which affect different devices independently, such as random dopant fluctuations (RDF), or the line edge roughness (LER).

According to the source and type of variations, an adequate statistical model must be employed. The next section shows a series of methods for capturing and analyzing parameter variations.

2.4.2 Statistical Analysis Methods

Variability effects can be included in the analysis by modeling the corresponding parameters as random variables. As a result, the estimated performance metrics, which depend on these parameters, will be also described by random variables. There are several ways to characterize a statistical distribution, either by employing its probability density function (pdf) or the cumulative distribution function (CDF). If obtaining the entire pdf or CDF characterization is not possible or intended, a coarser description is employed, by

specifying a few moments of the distribution, typically the mean and variance (or the mean and standard deviation).

Early approaches to statistical modeling of process variations have assumed Gaussian distributions for the process parameters and the resulting performance metrics [67, 148, 2, 10, 45]. Hereby, the dependence of a metric on the individual parameter variations is expressed in terms of the individual sensitivities, e.g. the path delay is estimated as:

$$D_p = N\left(D_{p,nom}, \left(\frac{\partial D_p}{\partial L}\right)^2 \sigma^2_{\Delta L} + \left(\frac{\partial D_p}{\partial V_{th}}\right)^2 \sigma^2_{\Delta V_{th}} + \left(\frac{\partial D_p}{\partial T_{ox}}\right)^2 \sigma^2_{\Delta T_{ox}} + \cdots\right) \tag{2.39}$$

This method assumes a linear dependence on each process parameter variation ΔP_i given by the first-order sensitivity:

$$\Delta D_p\left(\Delta P_i\right) = \frac{\partial D_p}{\partial P_i}\left(\Delta P_i\right) \tag{2.40}$$

resulting into linear-dependence expressions for the delay, respectively leakage [45]:

$$Delay = D_{nom} + \sum_{i=1}^{p} \alpha_i \left(\Delta P_i\right) \tag{2.41}$$

$$Leakage = \exp\left(V_{nom} + \sum_{i=1}^{p} \beta_i \left(\Delta P_i\right)\right) \tag{2.42}$$

where α_i and β_i are the first-order sensitivities to the respective parameter P_i.

However, variations in several physical device parameters can significantly differ from a normal distribution [24], such as deviations in the critical dimension (CD, usually the gate length) caused by shifts in the depth of focus (DOF), which add a negative skewness to the distribution. In addition, performance metrics exhibit several nonlinear dependences on process parameters, which result in non-normal distributions even if the process parameters are normally distributed. Nonetheless, as indicated in [24], the maximum operation required for many delay estimations (see also Sec. 2.2.2) is strongly nonlinear and adds a positive skewness to the result. These observations have lead to several approaches to represent non-normal random distributions [113, 3, 92, 36]. A simple triangular distribution model, discretized as an impulse train, has been used in [113] to enable a fast delay computation from random variables. Discretized pdf representations have been employed in [99, 3, 100, 55] for representing the distributions, combined with Monte Carlo sampling. Hereby, 3-point, 5-point, and 7-point piece-wise linear (PWL) approximations of pdfs and CDFs are employed in [55] for representing the delays and arrival times, respectively. Discretized pdf and CDF representations were used also in [6, 5, 4], where they are propagated across the delay tree by means of statistical sum and maximum operations. The propagation of PWL approximations for pdfs and CDFs has been also discussed in [92] in the context of a runtime optimization method through error trade-offs. A non-linear function of process parameters $f_A\left(\Delta X_N\right)$ was introduced

in [36] to derive an extended canonical dependence for the delay assuming non-Gaussian parameters and nonlinear dependences:

$$A = A_{nom} + \sum_{i=1}^{n_{LG}} a_{LG,i} \cdot \Delta X_{LG,i} + f_A\left(\Delta X_N\right) + a_{n+1} \cdot \Delta R_a \tag{2.43}$$

where X_{LG} are linear-dependence Gaussian parameters with the respective sensitivities a_{LG}, and $\Delta X_N = (\Delta X_{N,1}, \Delta X_{N,2}, \dots)$ contains the nonlinear and non-Gaussian parameter variations. ΔR_a indicates a normalized Gaussian parameter representing uncorrelated random variations, with the corresponding sensitivity a_{n+1}. An alternative method is employed in [157] which uses connectivity graphs to estimate the results of process variations described by discretized pdfs on device performance and employs new operators and domains based on sampling in the random variable space.

Once an appropriate representation of the variable parameters has been found, the variability must be propagated across the models, from the parameter level to the top-level system performance metrics. Earlier approaches relied on the simple propagation of extracted sensitivities [140, 115], extracted through response surface methodology (RSM) or Monte Carlo simulations and represented in the form of response surface functions (RSF). A more recent approach [163, 68] proposes a method to compute the stochastic response of performance metrics by means of orthogonal polynomial expansions in an infinite dimensional Hilbert space. This approach has the disadvantage of slow convergence of the infinite series representations, together with the high complexity of finding the coefficient functions. A further statistical method for parametric yield prediction [85] proposes the division of the parameter space into feasibility regions of regular shape (i.e. parallelipipeds or ellipsoids) and performing a numerical integration of the joint pdf of the sources of variation. This modeling approach relies again on linear representations of the performance metrics using sensitivity matrices. A propagation method assuming independent lognormal distributions of process parameters has been presented in [135]. Here, the total leakage current expression is approximated using a lognormal distribution:

$$pdf\left(I_{tot}\right) = \left(\frac{1}{I_{tot}\sqrt{2\pi\sigma^2_{N,I_{tot}}}}\right) \exp\left[-\left(\frac{\log\left(I_{tot}\right) - \mu_{N,I_{tot}}}{\sigma_{N,I_{tot}}\sqrt{2}}\right)^2\right] \tag{2.44}$$

where $\mu_{N,I_{tot}}$ and $\sigma^2_{N,I_{tot}}$ are the mean and variance of the normal random variable corresponding to the lognormal distribution. An analytical approach for computing the pdf of arrival time has been proposed in [168] for buffer circuits, which uses a recursive algorithm implying the computation of first-order derivatives for the delay expression. This method is however suitable only for simple circuits, for which basic analytic models exist, therefore does not scale with circuit complexity. A quadratic dependence of the delay on process parameter variations has been used in [171] to propagate distributions using second-order sensitivity sets. This approach extends the modeling capabilities a step

beyond first-order approximations, remaining nevertheless limited by the quadratic estimations and still requiring sampling in the parameter space. A Taylor series expansion up to the fourth order has been used as alternative in [176], which improves the modeling accuracy. Finally, improved Monte Carlo approaches have been used several times recently for statistical analysis, in the form of importance sampling [153, 175, 172]. This method translates the variability from the parameter space to the performance metrics in the form of parametric yield estimations. The key idea behind importance sampling is to reuse the results of Monte Carlo simulations and to obtain estimates of one random variable by sampling in a different distribution. For instance, the expectation of a function $\Psi(X)$ of the random variable X described by pdf $p(x)$ can be obtained as:

$$E_p(\Psi(X)) = \int_{-\infty}^{\infty} \Psi(x)\, p(x)\, dx = \int_{-\infty}^{\infty} \Psi(x)\, W(x)\, q(x)\, dx \tag{2.45}$$

where $W(x) = \frac{p(x)}{q(x)}$. In this way, the samples of X can be obtained using a different pdf $q(x)$. By properly weighting each sample with the function $W(x)$ the error introduced by sampling with a different distribution is minimized [172].

Correlations can arise between the distributions during the analysis, either from reconverging paths [6] (topological correlations) or from spatially correlated process parameters [24] (spatial correlations). Bayesian networks have been used in [22] to represent the circuit and accurately track topological correlations. Spatial correlations are handled in [23] using Karhunen-Loève expansions. While providing very accurate representations, Karhunen-Loève computations are always sample-specific and involve large computational overheads by relying on eigenvalue and eigenvector computations from the sample matrices. Simplified correlation structures have been also employed for describing intra-die variations, such as grid [35] or quadtree models [1]. A distance-dependent correlation model for intra-die variations has been introduced in [104] which divides the chip die into several perfect correlation regions. Spatial correlations were also characterized in [101] through an optimal spatial model which matches a set of observations using generalized least square fitting. Finally, the principal component analysis (PCA) is often employed for transforming spatially correlated variables into a set of standard decorrelated principal components of variation [150, 43].

To conclude, most approaches to statistical modeling are limited by several factors, such as Gaussian or lognormal assumptions, linear dependences, or the propagation of distributions considering only first, second, or up to fourth order approximations. Relatively recent methodologies still assume Gaussian distributions and linear functions of variation [8, 71]. A complete description of the distributions propagated entirely across technology-accurate complex models, which is also scalable for any number of parameter variations, has not been yet developed. This thesis brings an important contribution to this challenge in chapters 3 and 4.

2.5 Technology Accuracy

As discussed in Sec. 2.2, a major challenge in performance modeling is to develop accurate models for the current technologies and particularly to embed accurately the existing parameter variations. In this section we discuss three important aspects of technology accuracy, namely the process characterization, yield prediction and optimization, and transistor-level models for statistical analysis.

2.5.1 Process Characterization

The accurate use of process parameter variations assumes first a reliable method for the process characterization and extraction of the parameter distributions. An early methodology presented in [120] points out the importance of considering the sources of parameter correlations. It relies on a relatively simple approach, namely a direct sampling combined with clustering of the interdependent device parameter sets. While keeping the measured process parameters organized in sets, this method preserves the inter-parameter correlations and stores a direct link between each set and the corresponding die location.

A direct measurement-based method for process characterization has been published in [122] and proposes the statistical characterization of process parameters by means of resistance measurements. Within this approach, a set of test sites are implemented on each reticle which contain long and narrow polysilicon resistors. By measuring the resistance of these polysilicon lines, the distribution of CD variations is captured. Statistical methods to analyze measurement data and extract statistical distributions have been presented in [174], based on moment matching of quadratic models up to the third moment. Hereby, any given parameter is estimated as a quadratic function of a Gaussian random variable, which is fitted by matching the first three moments (i.e. mean, variance, and skewness). The method presented in [164] assumes a variable degree of accuracy in the description of parameter distributions, given the difficulty of extracting process information at early design stages. As a consequence, it employs random variable representations using bounds for CDFs to handle the partially-specified uncertainty descriptions. Mathematical theories from random fields and convex analysis were used in [169] to extract a spatial correlation function and the corresponding matrix from measurement data. A similar approach has been employed in [46] where variogram functions are used to extract spatial covariation models. An empirical variogram function is extracted from the data and is used to eliminate the global component of variation, while emphasizing the intra-die variability. The empirical model based on variograms is then tested against data and refined through weighted least-squares regression. The models employed in this thesis for modeling process parameter variations and spatial correlations are discussed in chapter 4.

2.5.2 Yield Optimization

Mapping the variations in process parameter space to the performance metrics is one way to estimate the parametric yield, and can be performed as discussed in Sec. 2.4.2. A correlated factor is the statistical yield optimization, with respect to given criteria. Here, an important premise for accuracy is to consider the particularities of the manufacturing technology.

A method for statistical technology mapping has been proposed in [147] for optimizing the logic synthesis with respect to leakage power minimization. The proposed approach finds a circuit mapping considering the dependence of the overall cost function on the variance in making the dynamic selection of the gates from a given technology library. In [105], power minimization is achieved using a linear programming technique. However, this approach accounts for only two sources of variability, namely the effective channel length (L_{eff}) and the gate-length independent threshold voltage, which are both assumed to be Gaussian random variables. In addition, the delay is approximated by a first-order Taylor series expansion, while the leakage power is modeled using a first-order sensitivity. A probabilistic description of power-performance tradeoffs has been used in [93] for design space exploration. This method uses sets of Pareto-optimal points to encode solutions in the power-delay space, which are not dominated by any other solution in the feasibility set. The parametric yield has been modeled in [21] as a function of the mean and variance of circuit leakage. Hereby, the leakage is minimized considering gate sizes, gate lengths, and threshold voltage.

The development of circuit-level statistical models with a high technology accuracy is presented in chapter 4. The method employed in this thesis relies on the yield optimization with respect to a custom cost function of the delay, dynamic power, and leakage.

2.5.3 Transistor-Level Models

A key factor to achieving good accuracy during the statistical analysis is to rely on accurate models down to the transistor level. A statistical method for device characterization using discrete probability propagation has been proposed in [157], which uses a simplified connectivity graph-based transistor model. Only a few process parameters are captured by this model and there is no systematic definition of the required statistical operators to process the parameter distributions. A statistical model for the leakage of double-gate MOSFETs has been introduced in [11], which depends only on gate length and body thickness variations. Moreover, the leakage is expressed using a Taylor series expansion, for which only the mean and variance are computed, and is approximated with a Gaussian distribution, which strongly limits the accuracy. An extrapolated transistor model from a set of data points has been proposed in [40] for fast statistical circuit simulations. This modeling approach is rather empirical and does not capture all process parameters: only variations in L_{eff} and V_{th} are captured. Recently, a very simple

transistor-level model has been presented in [47] and models the gate length as:

$$L_{jk} = \mu_{L_{jk}} + a_{jk}X_1 + b_{jk}X_2 \tag{2.46}$$

where L_{jk} is the gate length of transistor j from standard cell k, $\mu_{L_{jk}}$ is the mean value, and a_{jk}, b_{jk} represent the PCA coefficients of the two principal components X_1 and X_2. The scalability and accuracy of this model are however very limited, as only the gate length is statistically modeled and it relies on only two principal components for the variation (which are described by standard normal random variables).

For good accuracy with the manufacturing process a more detailed, physically accurate, transistor model is required, which can embed variations in possibly all existing process parameters. As discussed in Sec. 2.4.2, many statistical analysis approaches rely on simple or empirical transistor models, such as the relatively-old alpha-power law MOSFET model [139,28]. In contrast, this thesis introduces a fully-statistical detailed transistor model based on BSIM4 equations and embedding pdf descriptions for all process parameters employed by state-of-the-art technologies.

2.6 Optimization Resources at the Circuit Level

Until now only optimizations concerning the task partitioning, assignment, and scheduling have been addressed, which achieve improvements in delay and power consumption through architectural choices and system-level design decisions. Nevertheless, once an architecture is chosen and the tasks are mapped and scheduled on the corresponding resources, the communication activity between the cores still has a strong influence on the overall latency and power dissipation. Thus, it becomes very attractive to investigate further optimization options at the level of communication segments. Especially the choice of a particular signaling circuit, followed by circuit-level optimization techniques, such as supply voltage scaling and body biasing, would have a substantial impact on speed and power. The effects of such optimization measures are discussed in this section.

2.6.1 Choice of Signaling

Communication performance is partly determined by the segment attributes, such as width, length, and frequency, but it also depends strongly on the signaling method and on the particular transceiver circuit which is connected to the bus. A strong call for alternative designs in on-chip global interconnects has been recently issued by the ITRS roadmap [81], to reduce delay and power consumption. The first suggested approach is to use different signaling methods, which would include both signal design and signal coding techniques.

A series of novel signaling methods have been proposed in the literature [12,17,37,52, 87,161] which target the reduction of delay and power dissipation in on-chip global inter-

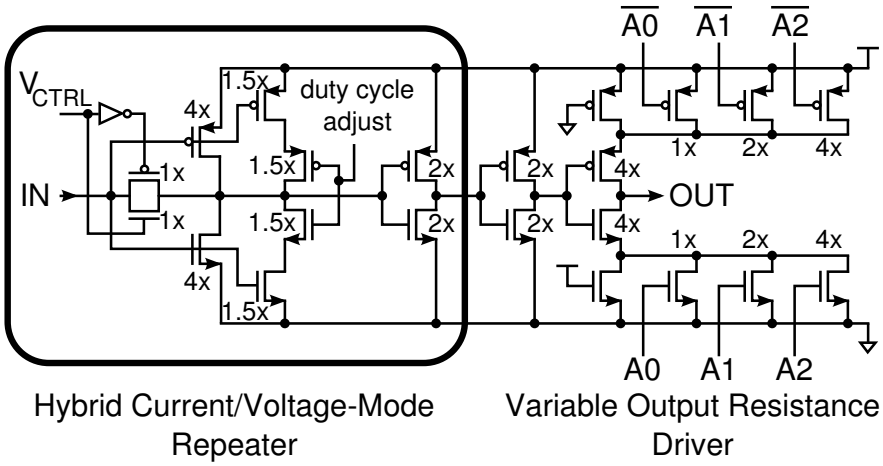

Fig. 2.17: Current/voltage mode repeater (after [17]).

connects. In [17], a hybrid current/voltage-mode signaling is proposed, which minimizes delay and static power consumption, being also compatible with repeater-insertion methods. The hybrid repeater circuit employed for this signaling method is drawn in Fig. 2.17 and consists of an amplifier with variable input resistance connected to a variable output resistance driver. When the control voltage V_{CTRL} is "0", the feedback transmission gate is opened and the driver operates in voltage mode. In this configuration, the input amplifier acts as a self-biased inverter. When V_{CTRL} switches to "1", the amplifier acts as a resistive termination for the communication line and the signaling occurs in current-mode. The three bits $A0$, $A1$, and $A2$ control the output resistance of the driver by switching the parallel transistors. The additional transistors connected to the internal node are used to control the duty cycle of the communication and are typically biased at $V_{dd}/2$ for a symmetrical duty cycle.

Voltage-mode and current-mode transmitters are similar in principle and typically switch a low-impedance output connected to the interconnect segment, as shown in the equivalent representation from Fig. 2.18(a). In current-mode signaling, one of the output transistors switches a static current path from the receiver to V_{dd} or ground. On the other hand, at the receiver side, a voltage-mode circuit exhibits a high input capacitive impedance, as illustrated symbolically in Fig. 2.18(b), whereas a current-mode receiver is characterized by a low-impedance input node, as shown by the circuit from Fig. 2.18(c). In the case of a current-mode connection, assuming a pull-down switch at the transmitter, a static current I_{cm} is switched between V_{dd} at the receiver side and the ground at the transmitter, as illustrated in Fig. 2.18(d).

Signaling in current mode improves the delay and hence the bandwidth of a communication segment by switching a low-impedance (resistive) sensing circuit at the receiver end and allowing a high static current to flow through the interconnect line. Current sensing techniques have been proven to effectively reduce propagation times in long interconnect lines [18,16] at the expense of a high static power dissipation. Dynamic power levels are, nevertheless, relatively low, due to the reduced voltage swings involved in

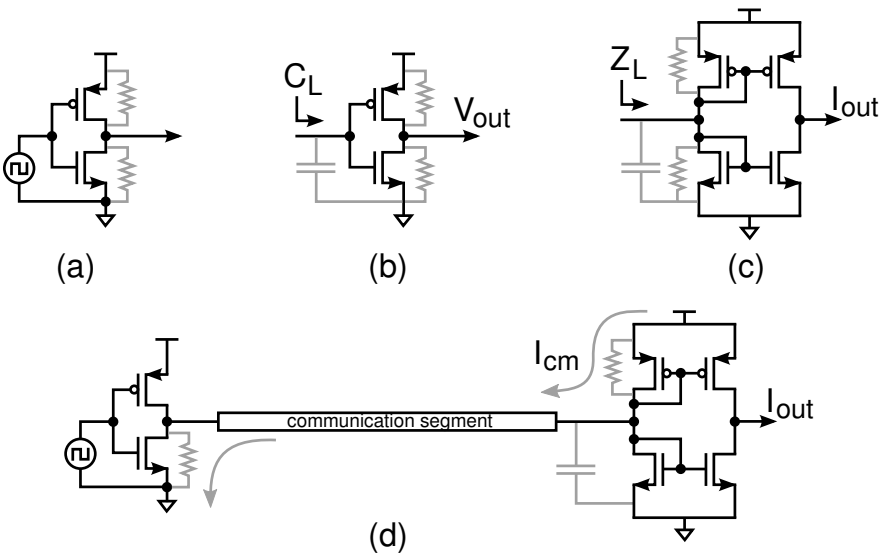

Fig. 2.18: Voltage and current sensing circuits: (a) hybrid-mode transmitter, (b) voltage-mode receiver, and (c) current-mode receiver (after [18]). Pull-down signaling path in current-mode (d).

current switching. As a result, hybrid-mode signaling schemes as the one used in [17] achieve a good trade-off by using fast current-mode signaling at high data activities, for maximum bandwidth, while switching to voltage-mode signaling at lower traffic rates, to reduce static power consumption.

Current-mode signaling methods have been employed recently also in differential schemes, such as for sending three data channels over two pairs of transmission lines [44], achieving a 37% increase in data rate over pure differential signaling. Two of the three channels are switched as half-swing differential signals, each over a pair of lines, hence reducing the occupied dynamic switching range in each channel and the total power consumption. The third channel could be thus inserted as a half-swing complementary common-mode signal over the two pairs of transmission lines used by the first two channels. The use of current-mode drivers has the benefit of increased data rate, but contribute also to the overall increase of 33% in power consumption.

A pulsed current-mode signaling is presented in [87] which speculates the LC behavior of interconnects at high frequencies to achieve a near speed-of-light propagation through a repeaterless link. The transmission of high-frequency, reduced-swing current pulses maximizes the effect of wire inductance, therefore the on-chip interconnect is operated as a transmission line with reduced dispersion. It is shown in [87], that full-rail optimally repeated RC lines exhibit a high bit energy consumption and are at least three times slower than the speed-of-light propagation. On the opposite, the reduced-swing current-mode operation allows for repeaterless global interconnects and the sharp current-pulse transmission achieves a near speed-of-light latency at low bit energies. A similar approach based on the same principle has been published in [37] and performs a PSK modulation with a high-frequency carrier for minimizing the delay, albeit at the cost of higher energy consumption.

Low-voltage swing signaling schemes have been widely investigated for saving power in interconnects. An analytic model for computing the optimum swing for minimum power consumption has been introduced in [152] and shows potential power reductions by a factor of 3 to 8 for voltage swings between 60 and 120 mV. The use of low-swing signaling has been demonstrated in a Pentium 4 [52], showing lower power dissipation and permitting the use of thin interconnect wires for reduced coupling capacitance and achieving compact layouts. Typically, low-swing voltage-mode signaling reduces power at the cost of delay increase, whereas current-mode signaling works fast, but increases static power dissipation. Nevertheless, in [161], a differential current-mode signaling has been introduced for improving both delay and power consumption. Hereby, control signals are employed to keep the power consumption to a minimum and allow the static current to flow only for a fraction of the cycle.

Bringing the benefits of the different signaling methods into the optimization of on-chip communication requires the ability to model and integrate the different signaling circuits into the system design framework. Chapter 4 presents the contributions of this thesis into this area and presents a complete methodology to accurately model the delay and power of signaling circuits considering parameter variations. The optimization framework is then able to select the optimum circuit from a library of models for each communication segment.

2.6.2 Voltage Scaling

The aggressive downscaling of supply voltages is one of the most effective solutions to reduce dynamic power consumption in digital circuits. In fact, V_{dd} scaling while keeping a fixed threshold voltage brings a quadratic reduction in dynamic energy at the cost of decreased circuit performance. The limits of voltage scaling and the behavior of circuits at very low supply voltages, in the subthreshold regime, have been investigated in [73]. These results are useful for predicting the achievable power improvement. An algorithm for computing the required voltage scaling for executing tasks with timing slacks has been introduced in [13], employing an architectural model of the application. Here, the focus lies on the distinction between static (off-line) and dynamic (runtime) voltage scaling and on the minimization of the runtime scaling overhead. An adaptive voltage scaling at the core level in multiprocessor chips has been proposed in [129] to compensate for asymmetries in power and performance due to process variations. All the aforementioned approaches investigate voltage scaling at the core level and do not specifically consider the power loss due to inter-core communication.

A series of relatively-new methods propose the combination of voltage scaling with bus architecture synthesis [126, 127, 125, 124]. In [126] and [127], a system-level model for the delay and power consumption of the communication tasks is employed, expressed

by:

$$T_c = \kappa \frac{V_i}{(V_i - V_{th})^\alpha} \tag{2.47}$$

$$E_c = \alpha_\tau \cdot C_{eff} \cdot V_i^2 \cdot T_c \tag{2.48}$$

where κ is a technology constant, V_i is the supply voltage, α is the saturation velocity, α_τ represents the switching activity, and C_{eff} is the effective switched capacitance during data communication. Further, [125] and [124] extend this method with a statistical analysis based on a first-order sensitivity dependence.

Two fundamental observations can be made. First, most of the published methods are focusing on voltage scaling at the core level, and do not consider energy benefits if the scaling would be applied also for the inter-core communication infrastructure. Second, the voltage scaling approaches for the communication structure use rather architectural-level models than circuit-level accurate estimations. It could be seen that voltage scaling can bring important gains in power if applied on the communication circuits. Nevertheless, for a successful application during the synthesis of communication architecture, accurate circuit-level models are required, which reflect the bus structure, length, type of signaling, and the manufacturing process. Sec. 4.2.4 and 4.3.3 present the contributions of this thesis to the integration of voltage scaling techniques with accurate circuit-level models of the communication structures.

2.6.3 Body Biasing

Body biasing is employed to change the threshold voltage of CMOS transistors by applying a voltage between the substrate and the source. In the case of forward body biasing (FBB), the body-to-source junction is directly biased, and the threshold voltage decreases. FBB achieves an increase in speed, but also in leakage. The opposite holds for the reverse body biasing (RBB), where the bulk junction is reversely biased, achieving a higher V_{th}, hence a lower leakage at the cost of reduced transistor performance.

Either a single chip-wide body bias, or many local body biases for different regions can be applied. It is to be noticed that a separate n-well is required for each PMOS body biasing regions. Similarly, for multiple NMOS body bias regions, a triple-well process is needed for the separation of bulks (see Fig. 2.19). Further, the body bias value can be either fixed or dynamically adjusted at run-time. A fixed body bias is typically chosen at the design or test time to optimize a selected performance metric and remains constant over the lifetime of the chip. On the opposite, the method which dynamically adjusts the body bias, also called adaptive body biasing (ABB), modulates the substrate voltage at run-time to achieve a tradeoff between performance and leakage power consumption.

Dynamic power consumption and leakage power consumption optimizations are obtained in [170] with a combination of adaptive body biasing and dynamic voltage scaling. Based on an analytic energy consumption expression derived from the alpha-power law

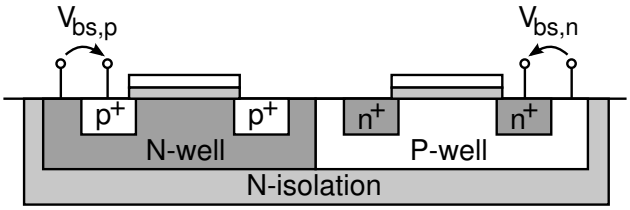

Fig. 2.19: Body biasing of NMOS and PMOS transistors in a triple-well process.

transistor model, the optimal body bias values are evaluated at every given frequency and adjusted at run-time to achieve a tradeoff between optimal energy consumption and clock speed. Energy-efficient architectural techniques including adaptive body biasing are discussed in [73] to reveal the tradeoff between leakage and performance implied by adjusting the threshold voltage. Within this context, it is pointed out that reverse body biasing has a worsening effect with respect to the short channel effects, which becomes more significant with technology scaling.

Another typical use of ABB is to reduce the influence of process parameter variations [42,41]. For instance, an ABB method for increasing frequency and reducing leakage in digital chips affected by variations has been implemented in [159] using a circuit for critical path monitoring and applying the optimal body bias for each digital cell. The authors of [119] approach the reduction of dynamic power, leakage power, and increase in the speed of digital circuits by means of process variations estimation circuits and demonstrate a digital on-chip controller for selecting an *a priori* stored body bias value. A forward body bias is used in [110] for reducing delay variations in digital circuits and an automatic body biasing architecture for minimizing the active power consumption has been published in [90]. The approach in [117] employs a fixed forward body bias after performing external measurements to identify parameter variations. The method proposed in [154] mitigates the effects of variability by testing several supply voltages with a body bias controller which adjusts the body bias to meet the frequency requirement, followed by the selection of the supply voltage value which achieved the minimum power consumption. The use of adaptive body biasing in chip multiprocessors to improve symmetry by speeding up slower cores has been examined in [79]. Finally, the limitations of ABB methods for addressing process variability are discussed in [26], which also points out the importance of delaying the body biasing step until test time, where a leakage measurement can evaluate precisely the influence of parameter variations.

The broad usage of body biasing techniques in digital circuits reveals a significant potential of this method to reduce leakage power consumption where performance drops are acceptable. Thus, a complex on-chip communication infrastructure, which dissipates large amounts of leakage power and can locally afford lower speeds due to the inherent slacks, represents a very promising candidate for body biasing. Since until now the body biasing methods have focused on optimizations at the core level, one of the main scopes of this thesis is to investigate the influence of body biasing on the communication circuits and to integrate this method in the optimization framework. Sec. 4.2.4 and 4.3.3 present

the application of body biasing on the circuit-level communication models, while the results are presented and discussed in Sec. 4.4.6.

2.7 Summary

Based on the observation that significant performance amounts and serious challenges in MPSoC designs are dictated by the inter-module communication, this section discussed the most important aspects which must be considered in the design of on-chip communication architectures. It has been shown that the design space exploration must rely on efficient abstractions of the application flow combined with accurate performance macromodels to achieve good estimations of the actual design performance. In this context, the concepts of delay and power macromodels have been detailed and several modeling approaches have been discussed. Moreover, the importance of statistical modeling combined with process accuracy of performance estimations has been emphasized in the context of complex state-of-the-art manufacturing technologies. It has been also pointed out that task scheduling has a significant influence on system latency and total energy consumption and several scheduling approaches have been discussed.

One of the paramount challenges in system and communication design is represented by the increasing parameter variations. Several statistical methods to analyze and model parameter variations have been examined and their drawbacks have been indicated. The most important challenge in this context is to accurately represent the parameter distributions and to model the dependence of performance models on parameter variations by accurate propagation of the distributions. Moreover, technology accuracy plays a key role in the context of both accurate performance models and statistical analysis and the shortcomings of modeling approximations and of the underlying transistor-level models have been evidenced. Finally, the additional optimization resources for communication at the circuit level have been illustrated, including the choice of signaling method, voltage scaling, and body biasing. Nevertheless, the contributions of this thesis to each of the important shortcomings and challenges in the communication synthesis have been mentioned.

Chapter 3

Variability-Aware Performance Macromodels

Contents

An important objective of this thesis is to provide an integrated framework for the parametrized joint optimization of delay and power at the system level during the communication architecture synthesis. To achieve this goal, the developed methodology requires an accurate description of the target MPSoC architecture and of the running applications, considering parameter variations in execution times, data flow, power consumption, and lower-level process parameters. This chapter presents a unified methodology for describing the synthesis-relevant parameters of the application profile, architectural details, and manufacturing process.

Accurate estimations of the parametric yield rely on the performance of the underlying statistical analysis. For this reason, the parameter variations must be modeled using adequate random variable representations. In addition to commonly-used Gaussian variation models, customized parameter distributions resulted from measurements and profiling analyses must be implemented. A generalized representation of both standard and custom, discretized distributions is implemented and presented in this chapter.

Performance macromodels require the composition of several underlying variable parameters in complex analytic or numerical expressions. The propagation of statistical distributions from the parameter level up to the performance model is therefore depending on several algebraic compositions. The method developed in this thesis propagates the entire representation of statistical distributions during the algebraic operations on random variables. This chapter describes the propagation method for pdfs and the set of statistical operators developed for the operations on random variables.

The random variable representations and their algebraic composition using statistical operators are employed to develop statistical performance macromodels for delay and power. Within this context, the interconnection between MPSoC architectural resources and the abstract representation of the models is discussed. The structures of the developed macromodels are presented and the variability representation inside each node is

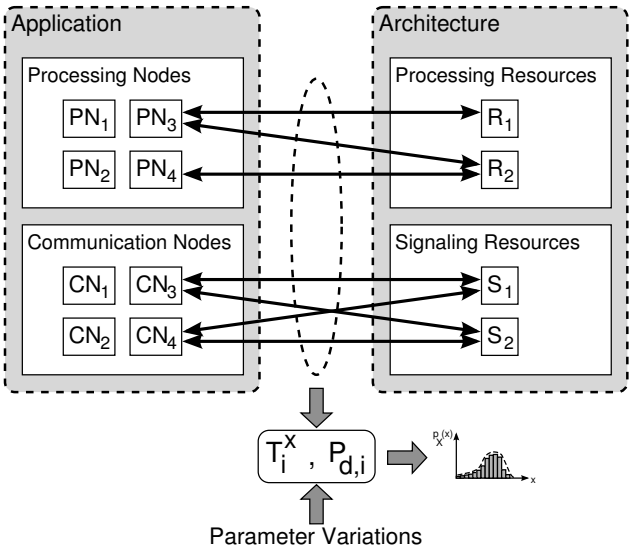

Fig. 3.1: Description of timing and dynamic power values depending on e.g. resource mapping and parameter variations.

illustrated. Particularities involving design decisions, such as reassigning a processing node to a different resource, or changes in the scheduling list of a resource are reflected by important changes in the macromodels. The required changes and a method to update the affected distributions is presented.

Variability-aware representations in the form of random variable performance models have a direct impact on the system-level design space exploration. Design decisions such as task mapping and scheduling have to be driven by the evaluation of performance metrics. For this, a method to build a cost function from a set of statistical distributions representing delay, dynamic power, and leakage power must be found. Such particularities are discussed here as well.

This chapter is organized as follows. Sec. 3.1 presents a method for the extraction and specification of application and architectural profiles with respect to variable parameters. In Sec. 3.2, a generalized random variable model is developed, based on a discretized representation of statistical distributions with adjustable accuracy. Further, Sec. 3.3 presents the development of a propagation technique for pdfs implemented using analytic and numeric statistical operators. The embedding of random variable models in the synthesis framework is discussed in Sec. 3.4. Afterwards, the performance macromodels for delay and energy consumption are presented in Sec. 3.5 and 3.6, respectively. Finally, the global optimization of resource allocation and scheduling is analyzed in Sec. 3.7.

3.1 Application and Architectural Profile

Application profiles are required for the extraction of processing and communication tasks and are used as the input to the synthesis algorithms. Hereby, a detailed descrip-

tion regarding variability is of key concern. As an example, the same task may exhibit different execution times on different cores and, in addition, the execution times are functions of several parameter variations. Further, dynamic and leakage power consumptions are as well dependent on the processing resource and exhibit variations which must be described in an appropriate way. Fig. 3.1 illustrates this concept: a particular mapping of processing and communication nodes on the architectural resources results into different values of the execution times and dynamic power dissipations. In addition, the influence of parameter variations results into the need for a statistical description of these metrics. Nonetheless, data dependencies and communication needs are to be extracted and expressed in a variable form, depending on the workload changes of the running applications.

Besides application-related details, an accurate description of the hardware platform must be provided, in the form of a resource set. Within this context, processing elements, memory units, interfaces, controllers, and other application-specific hardware blocks must be included.

This thesis provides a unified application interface to describe task details, data dependencies, timing, power, and data flow parameters, complete hardware resource sets, task-resource compatibility associations, and parametric yield constraints in a consistent form. Variable parameters are specified using the most appropriate distribution from a set of given distribution types, as well as in the form of a non-standard discretized probability density function (pdf) description.

3.1.1 Extraction of the Application Profile

One of the critical tasks for the automated synthesis of communication structures is developing a system profile at a high level of abstraction and extracting the on-chip communication events. This work considers embedded systems, realized as MPSoCs, which are suitable for applications with variable data flows. Within this context, an application profile consists of the architecture definition, a detailed data flow graph, execution costs, and the design constraints. A schematic representation of the profiling steps is depicted in Fig. 3.2.

The first application description is a behavioral model, written in a high-level language (such as C++), which enables the flexible division into processing tasks, as well as further refinements of the granularity (e.g. separate load/store operations with memory units etc.). At this point, the inter-task communication loads can be estimated by using arrays of equally-sized block structures which encapsulate the data transferred in the function calls between the processing tasks. The amount and frequency of the data transfers can be tracked down automatically using a language-specific tool (e.g. the GNU profiler *gprof*), followed by an estimation of the variability parameters. Hereby, the most appropriate variability model is selected from a set of random variable descriptions adapted to the profile, ranging from standard distributions to estimated distribution functions,

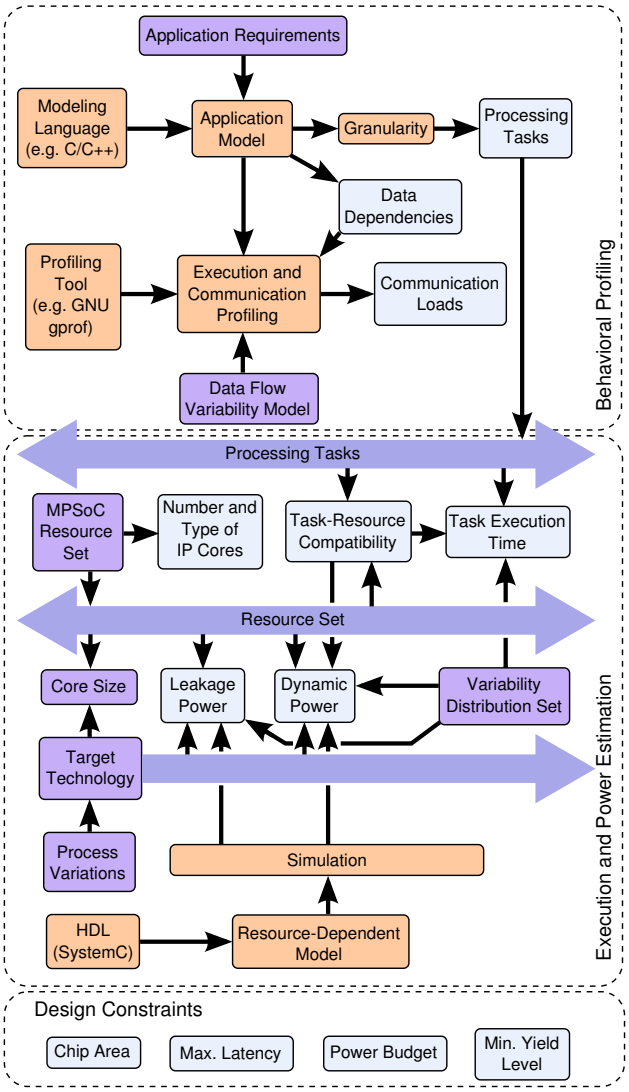

Fig. 3.2: Schematic workflow showing the application profiling steps.

mostly in the form of discretized pdfs (see also Sec. 3.2).

Granularity is an important factor in the modeling accuracy of processing tasks and has a strong influence on the possibilities of partitioning the application onto the available architectural resources. Nevertheless, the complexity of the design optimization is strongly affected by the task granularity choice. Due to the main focus on communication synthesis, this thesis sets the granularity of processing nodes at the task level. Finer granularities, such as instruction-level or operation-level nodes, would result into a finer modeling of the processing nodes at the cost of a significantly higher complexity. Since the instructions and operations of a task are usually executed on the same core, the increase in modeling overhead does not improve the representation of communication nodes, therefore a finer granularity is not justified in this case.

The information obtained from the behavioral modeling and profiling activities contains the number of processing tasks, their data dependencies (links between the tasks)

and the inter-task communication loads, specified as statistical distributions. This information serves as input to the communication synthesis framework and is specified in an application description INI file. The section of the INI file describing the relationship between the processing tasks (processing nodes) includes the following:

- N_{PN}, the number of PNs
- $PN_{i,name} = \{name\}$, the name of each PN
- $PN_{i,dd} : PN_j, PN_k, \ldots$, the data dependencies of each PN (children links)
- $L_c^{i,j} = \{distribution\}$, the communication load between PN_i and PN_j, expressed as random variable distribution (predefined distribution type or discretized pdf)

3.1.2 Architecture and Technology Specification

The choice of a target architecture and technology depends strongly on the application constraints, such as latency, area, power budget, and also the design and manufacturing costs. Typically, the manufacturing costs and performance levels (latency, power budget) determine the choice of a particular technology node, but also the availability of several intellectual property (IP) modules for the given technology is to be considered. Next, the set of available IP modules for the design is collected into a resource set, specified by the number and type of resources. In combination with the selected technology, the leakage power of each resource is estimated. If no leakage information is provided with the IP library, a leakage simulation of each resource is required. Depending on the desired level of accuracy, this step may involve synthesizing the resources and performing gate-level power simulations, or a higher-level model written in hardware description language (HDL) is employed together with technology-dependent power estimations. The estimated leakage power, including process and environmental variations, is then described statistically, using the set of available random variable distribution types.

After specifying the resource set, a task-resource compatibility graph is defined to specify allowed mappings during the synthesis. Given the compatibility information, performance attributes such as execution time and dynamic power consumption are estimated and specified in the form of random variable distributions. A resource-dependent simulation using HDL models enables a first-order approximation of the execution time and power consumption levels of each processing task on the compatible resources. Due to process and environmental parameter variations, both execution time and power consumption are expressed as statistical distributions.

The information describing the target architecture is also stored in the configuration file and includes:

- N_{RT}, the number of available resource types (processors, ASIC, memory, interfaces etc.)

- $RT_{i,name} = \{name\}$, the name of each resource type
- N_{RT_i}, the number of available resources of type RT_i
- $P_{l,RT_i} = \{distribution\}$, the leakage power consumption of resource type RT_i
- $PN_{i,co} : RT_j, RT_k, \ldots$, the compatible resource types for executing PN_i
- $T^x_{i,RT_j} = \{distribution\}$, the execution time of PN_i on the compatible resource type RT_j
- $P^{RT_j}_{d,i} = \{distribution\}$, the dynamic power consumption for executing PN_i on the compatible resource type RT_j

Communication transfers between the resources are strongly dependent on task allocation and scheduling, as well as on the transceiver type and bus attributes, and are therefore estimated during the synthesis using accurate models. Further, process-dependent parameters and parameter variations are specified in the input configuration files considering die area and spatial intra-die variability. First, the NMOS and PMOS transistor parameters are specified at their nominal value, according to the BSIM4 description for the given technology. Next, the die area is divided into a custom-sized grid and the position of each resource in the grid is specified. Floorplanning of the IP resources is outside the scope of this thesis, therefore the positions of resources in the grid are considered given. After that, the process parameters which exhibit deviations are specified using the distribution models and by specifying a spatial correlation behavior (see Sec. 4.1.2). Finally, design constraints such as area, latency, power budget, and the required yield level can be directly specified in the profile.

3.1.3 Variability Description

As discussed in Sec. 3.1.1 and 3.1.2, the application and architecture description contains several variable quantities, including communication loads, leakage and dynamic power consumptions, execution times, and process parameter variations. These variable quantities must be specified in the input configuration files, therefore a unified statistical description is required.

Sec. 3.2 will describe in detail the developed random variable models employed in this work, which represent the set of available statistical descriptions for the variable quantities. The developed models include standard distributions, such as Gaussian, uniform, lognormal, Cauchy, Pareto, Weibull, but also custom pdf representations of any estimated distribution, specified in discretized form. For instance, the leakage power of resource type RT_i can be expressed as a Gaussian distribution in the INI file in the form of:

$$P_{l,RT_i} = N\,(14.\text{e-}3,\ 5.\text{e-}3) \quad [\text{W}]$$
$$P^{\sigma_{rel}}_{l,RT_i} = 3$$

which specifies a normal distribution of mean μ = 14 mW and standard deviation σ = 5 mW. The $P_{l,RT_i}^{\sigma_{rel}}$ factor indicates the relevant sigma domain considered for approximating the spread of the distribution, which is set in this case to three sigma (the minimum and maximum leakage can be approximated with $\mu - 3\sigma$ and $\mu + 3\sigma$, respectively). Better accuracy can be achieved by setting the σ_{rel} domain to higher values, e.g. 4σ or 6σ.

Another convenient way for expressing multiple values which are in the same domain is to specify a common order of magnitude. Assuming several execution times in the nanosecond domain, expressed e.g. as uniform distributions between their minimum and maximum value, they can be specified as:

$$\begin{aligned} T_{1,RT_j}^{x} &= U(12,18) \\ T_{2,RT_j}^{x} &= U(10,14) \\ T_{3,RT_j}^{x} &= U(7,12) \\ &\cdots \\ T_{RT_j}^{OM} &= \text{1.e-9} \quad [\text{s}] \end{aligned}$$

The common order of magnitude (OM) of 1 ns indicates that all the above specified timing values are expressed in nanoseconds.

3.2 Random Variable Model

As discussed in the previous sections, the large majority of design values considered in the communication synthesis framework are inherently exposed to variations, either environmental (e.g. process, thermal, and voltage) or functional (data flow variations). Due to this non-negligible aspect, all the key parameters in the design are represented as random variables (RVs).

There are several descriptions available to completely characterize random variables, like e.g. the probability density function (pdf), the cumulative distribution function (CDF), or the characteristic function ($\Phi_X(\nu) = E\left\{e^{j\nu X}\right\}$, for operations in the symbolic frequency domain). From these available descriptions, the pdf has been selected for internal representations of random variables in the framework, as it relates directly to the experimental distribution of an estimated random variable and can be therefore seamlessly approximated with the histogram of sampled RV values.

Standard distribution functions, such as Gaussian, lognormal, or uniform, are implemented using the exact pdf analytic expression. Other distribution types, particularly custom non-normal distributions, or the results of nonlinear operations on random variables, are implemented using a discretized model with variable accuracy.

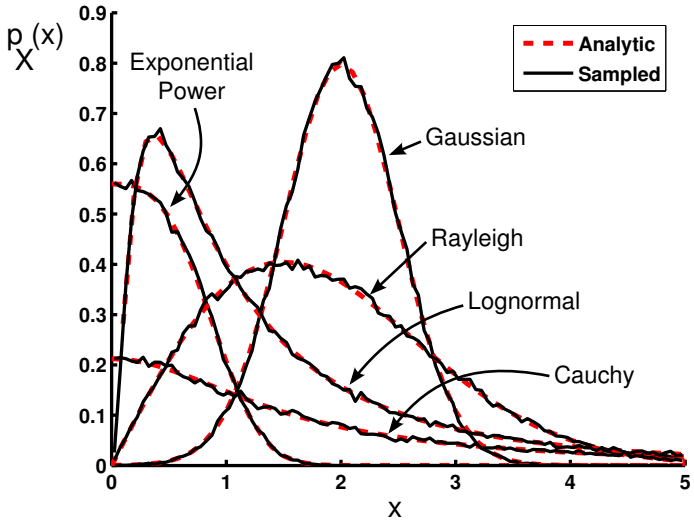

Fig. 3.3: Analytic and sampled pdfs for several standard distributions.

3.2.1 Employed Standard Distributions

The synthesis framework implements a limited subset of standard random distributions for representing parameter variations which can be accurately modeled by well-known distribution laws. Examples include:

- Gaussian distribution $N(\mu, \sigma)$
- Exponential power distribution $E(a, b)$
- Cauchy (Lorentz) distribution $C(a)$
- Rayleigh distribution $R(\sigma)$
- Uniform distribution $U(a, b)$
- Lognormal distribution $L(\zeta, \sigma)$

Internally, the corresponding random variables are characterized only by the parameters of the known distribution law. After storing the distribution type and parameters, samples from each distribution type are obtained with very good accuracy using the functions from the GNU Scientific Library (GSL) [65]:

- `gsl_ran_gaussian` (R_g, σ) for Gaussian distributions
- `gsl_ran_exppow` (R_g, a, b) for exponential power distributions
- `gsl_ran_cauchy` (R_g, a) for Cauchy distributions
- `gsl_ran_rayleigh` (R_g, σ) for Rayleigh distributions

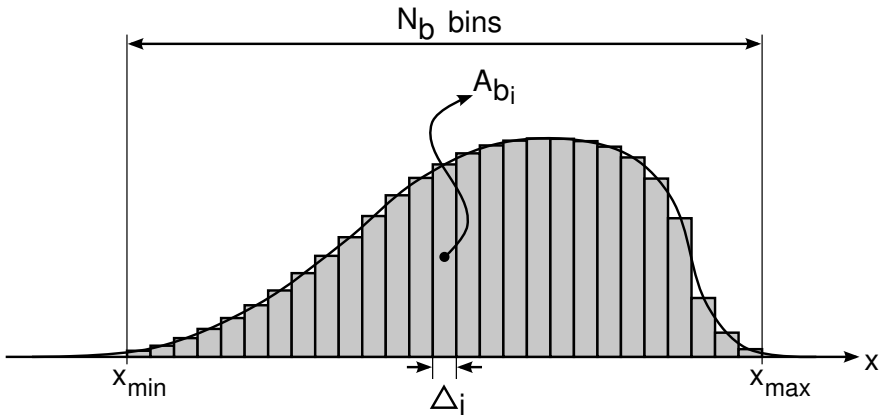

Fig. 3.4: Discretized pdf over N_b bins.

- `gsl_ran_flat` (R_g, a, b) for uniform distributions
- `gsl_ran_lognormal` (R_g, ζ, σ) for lognormal distributions

where R_g is the selected random generator type. Figure 3.3 illustrates the accuracy of the sampled values in a comparison between the analytic expression of the pdf and the distribution of samples generated for a Gaussian $N\,(2, 0.5)$, exponential power $E\,(1, 3)$, Cauchy $C\,(1.5)$, Rayleigh $R\,(1.5)$, and lognormal $L\,(0, 1)$ random variable. Each sampled pdf has been extracted from 100 000 generated samples.

The quality of the generated samples can be selected by choosing from a series of different random number generators, including e.g. the following [65]:

- `taus`, Tausworthe generator (870k doubles/second)
- `ranlxs0`, second-generation RANLUX algorithm (571k doubles/second)
- `ranlxd1`, double precision output, second-generation RANLUX (254k doubles/second)

As explained in [65], these generators offer several "luxury" levels, extremely long periods, and satisfy most statistical tests. A `taus` generator has been employed in the example from Fig. 3.3.

3.2.2 Discretized pdf Model

For parameter variations which cannot be described by standard distributions, a more general estimated distribution model is developed in this work. Besides non-standard distributed parameters, also nonlinear operations applied to standard distributions result into nonstandard pdfs, which fail to be represented by the models discussed in the previous section. In contrast, the model developed here is designed to fit any form of pdf with a precision which can be adjusted through several parameters.

Considering a random variable X described by an arbitrary pdf $p_X(x)$, a discretized function $\widehat{p}_X(x)$ is defined and computed as:

$$\widehat{p}_X(x) = \frac{A_{b_i}}{\Delta_i} = \frac{1}{\Delta} \int_{x_{min}+(i-1)\Delta}^{x_{min}+i\Delta} p_X(\tau)\, d\tau \tag{3.1}$$

where b_i is the discrete bin corresponding to the continuous value x, A_{b_i} is the bin area, and Δ_i is the bin width. Assuming that the support of $p_X(x)$ is distributed into equally-spaced bins, then $\Delta_i = \Delta,\ (\forall)\, i$ and the bin area is equal to the integral from (3.1).

As shown in Fig. 3.4, the support of $p_X(x)$ is limited for practical reasons to the interval $[x_{min}, x_{max}]$, beyond which the values of random variable X are considered to occur with no significant frequency for the given application. This interval is divided into N_b bins with widths $\Delta_i,\ i = 1 \ldots N_b$ and heights equal to the integral of $p_X(x)$ over the bin range. In the general case, the bin widths are not equal, allowing for an optimization of the bin width with respect to the pdf curvature. Nevertheless, using very dense and equally-spaced bins simplifies the analysis without a significant impact on accuracy.

Statistical moments can be estimated from the discretized pdf by replacing the continuous expressions with equivalent discrete definitions. For instance, the first-order moment (expected value) is defined starting from its continuous definition as:

$$E(X) = \int_{-\infty}^{\infty} x p_X(x)\, dx \tag{3.2}$$

$$\widehat{E}(X) = \sum_{i=1}^{N_b} \frac{r_{i-1} + r_i}{2} \Delta_i b_i \tag{3.3}$$

Here, the value of x is approximated by the center of the bin, whereas the integral of the pdf over the bin range is estimated by the bin area. In a similar way, the centered second-order moment can be derived as:

$$Var(X) = \int_{-\infty}^{\infty} (x-\mu)^2 p_X(x)\, dx \tag{3.4}$$

$$\widehat{Var}(X) = \sum_{i=1}^{N_b} \left(\frac{r_{i-1} + r_i}{2} - \widehat{E}(X) \right)^2 \Delta_i b_i \tag{3.5}$$

3.2.3 Typical Usage and Accuracy Control

The customized discrete pdf model is employed to represent variable parameters in two cases. First, process parameters exhibiting strong deviations from standard Gaussian or other well-known distributions can be specified using this RV model. Also other performance parameters estimated in the application profiling step can be well represented

using the discrete RV model. The complete specification of such parameters P_k in the input configuration includes the following information:

- $DD\,(N_b)$, specifying a discrete distribution model with N_b bins
- $r_i,\ \text{for}\ i = \overline{0, N_b}$, the ranges representing the bin separation points, for bins with distinct widths Δ_i
- $P_{k,min}, P_{k,max}$, the limits of the approximation interval, for bins with equal widths $\Delta = \dfrac{P_{i,max} - P_{i,min}}{N_b}$
- $b_i,\ \text{for}\ i = \overline{1, N_b}$, the normalized height of each bin, such that $\sum\limits_{i=1}^{N_b} b_i \Delta_i = 1$

The above specification replaces the usual $P_k = \{distribution\}$ for parameters specified using the discrete RV model. The number of bins and bin ranges are established according to the accuracy requirements. Bin heights are computed from the experimental distributions obtained during the profiling of performance parameters, such as execution times, communication loads, or power consumption. In this case, the bin height will be given by:

$$b_i = \frac{N_b}{N_s\,(P_{k,max} - P_{k,min})} \sum_{j=1}^{N_s} \Big(H\,(s_j - r_{i-1}) - H\,(s_j - r_i) \Big) \tag{3.6}$$

where N_s is the number of samples obtained for the given parameter, s_j is a particular sample of P_k, and $H\,(x)$ is the Heaviside step function. For a process parameter described by a non-standard distribution law $p_{P_k}\,(x)$, which is available in any form (analytic closed-form expression, look-up table etc.), the bin heights can be computed using numerical integration. An example with a good tradeoff between accuracy and speed is Simpson's 3/8 rule [38]:

$$b_i = \frac{r_i - r_{i-1}}{8} \Big(p_{P_k}\,(r_{i-1}) + 3p_{P_k}\,(x_1) + 3p_{P_k}\,(x_2) + p_{P_k}\,(r_i) \Big) \tag{3.7}$$

where x_1 and x_2 are equally-spaced points between the bin ranges r_{i-1} and r_i ($p_{P_k}\,(x)$ is assumed to satisfy the normalization condition $\int_{-\infty}^{\infty} p_{P_k}\,(x)\,dx = 1$).

A second case of usage for the discrete model is within the synthesis framework, for estimating the results of algebraic operations on random variables. As shown later in this chapter, the composition of RVs using various operators often results into non-standard distributions. As discussed in chapter 2, maximum operations add a positive skewness to distributions. In addition, the division of two random variables usually results into heavy-tailed distributions. Moreover, for most algebraic operators there is no known analytic expression for the distribution of their result. Given these considerations, the discretized RV model is also employed for representing the result of algebraic operations applied to random variables, for which the result does not obey to standard distribution laws. First, the support of the result distribution is estimated and divided into bins. After

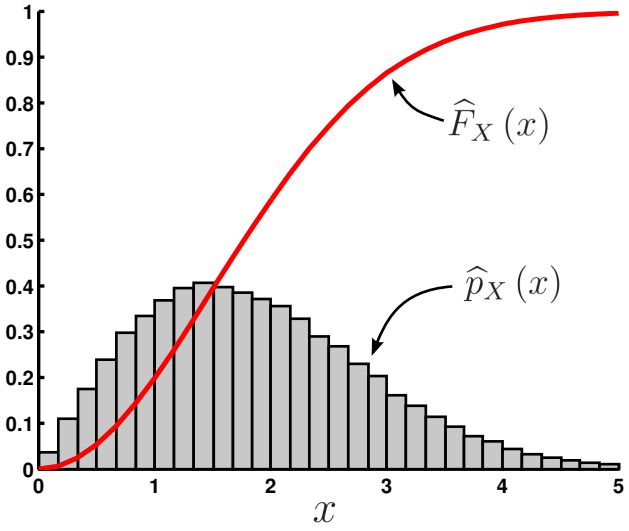

Fig. 3.5: Cumulative distribution function computed from a discrete pdf.

that, the height of each bin is computed by applying the respective operator on the input variables. Sec. 3.3 discusses the use of discretized RVs for operations on random variables in more detail.

Any distribution type can be approximated with adjustable precision using the discretized RV model. Hereby, the tradeoff between accuracy and computation speed can be adjusted by modifying the following parameters:

- The support of the approximated pdf, given by the $[x_{min}, x_{max}]$ interval;
- N_b, the number of bins inside the pdf support;
- N_s, the number of samples, if the distribution is approximated from a set of sampled parameters (in the profiling step);
- The numerical integration method for computing the bin heights b_i;
- The type of random generator and its "luxury" level, in the particular case in which the distribution is approximated using Monte Carlo samples from other known distributions.

It can be observed that the accuracy of the discrete pdf model is not intrinsically limited. By increasing the number of bins and the support limits, any desired accuracy level can be achieved, at the cost of increased computational overhead.

3.2.4 Sampling Technique for Discretized pdfs

Once a random variable has been characterized using the discretized pdf model, it is very useful to develop a function which generates samples from this distribution. First, the

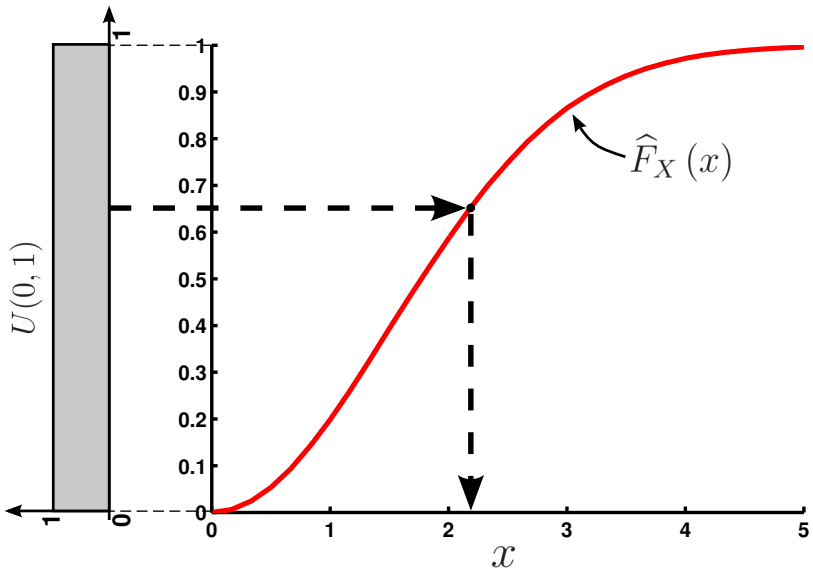

Fig. 3.6: Sampling method using a standard uniform distribution and the CDF.

cumulative distribution function (CDF) is computed from the discretized pdf:

$$\widehat{F}_X(x) = \sum_{j<i} b_j \Delta_j + b_i (x - r_{i-1}) \tag{3.8}$$

where b_i is the bin containing the value x (bounded by the ranges r_{i-1}, r_i). The sum accumulates the areas of the bins before b_i, while the second term in equation (3.8) calculates the area enclosed between value x and the left boundary of the current bin. Due to this partial contribution of the bin b_i, the computed CDF $\widehat{F}_X(x)$ is continuous in every point. Fig. 3.5 illustrates an example of discretized pdf and the resulting CDF computed with (3.8).

Once the CDF has been computed, samples from the random variable X, described by $\widehat{p}_X(x)$, can be obtained in the following way. Considering another RV Y uniformly distributed in the interval $[0, 1]$, samples x from RV X are obtained from samples y of RV Y as:

$$x = \widehat{F}_X^{-1}(y) \tag{3.9}$$

or, alternatively, x is the solution of equation $\widehat{F}_X(x) = y$, where $\widehat{F}_X(x)$ is strictly monotone. Due to the monotonicity of the CDF, this equation can be efficiently solved in a few steps using e.g. the bisection method [38]. Fig. 3.6 illustrates this concept.

3.3 Method for the Propagation of Distributions

Since the data flows across performance macromodels (PMs) originate from multiple design parameters which are exposed to variations, computing the performance metric at the PM output requires inevitably to apply arithmetic computations at the PM nodes on

random variables. In other words, the statistical distributions are propagated from the parameter-level description across the PM, towards the output performance metric.

Several methods for the propagation of variability have been discussed in Sec. 2.4.2 and their disadvantages have been pointed out. Either only a few moments of the distributions are propagated, or linear models are assumed for the dependencies on variable parameters. In addition, only a few methods consider non-Gaussian parameter distributions.

In contrast, the method presented here brings the following novel improvements:

- propagates the complete pdf representation across each operation (not only a few moments);
- is compatible with any nonlinear closed-form expression of performance models;
- its flexibility (operation-based propagation of complete pdf) allows for including any number of parameter variations in the performance model expression;
- employs both standard distribution models and custom discrete pdf models for non-standard distributions;
- has several adjustable parameters to allow the tradeoff between accuracy and evaluation speed.

The proposed method relies on the evaluation of the result pdf in each operation node of a performance macromodel. The input random variables are represented using their pdfs and statistical operators are developed to compute the pdf of the result. For the sum, maximum, difference, and product operations, the result pdf is directly evaluated using analytic operators. For other operations, the result pdf is obtained through statistical estimation. In the following, the variability propagation method is discussed in the particular case of the commonly-used operators in performance macromodels (sum, maximum, difference, and product), then the analysis is extended to other operation types. The presentation of statistical operators considers first the uncorrelated case. The implications and methods for handling several correlation types between the random variables are discussed in Sec. 3.3.5.

3.3.1 Statistical Sum and Maximum Operators

Statistical sum and maximum operations have been extensively studied in the field of statistical timing analysis, as indicated in [24]. Several approaches [19, 160, 97, 89] rely on analytic representations, albeit assuming normal distributions. Here a derived analytical method is presented which has been adapted for the random variable models developed in this work.

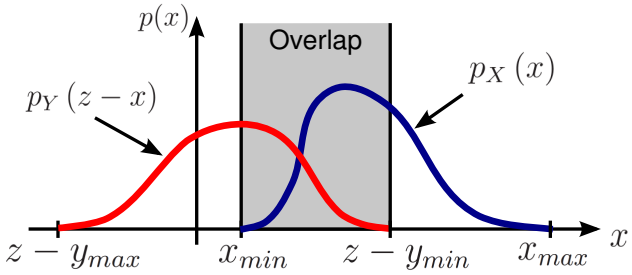

Fig. 3.7: Limits of the overlap during the sum computation.

Let Z be a RV equal to the sum of two random variables X and Y. The pdf of the sum is defined by the convolution product:

$$p_Z(z) = (p_X * p_Y)(z) = \int_{-\infty}^{\infty} p_X(x) \cdot p_Y(z-x)\, dx \tag{3.10}$$

The interval over which the sum pdf extends is bounded by $z_{min} = x_{min} + y_{min}$ and $z_{max} = x_{max} + y_{max}$. After computing the bounds, the interval $[z_{min}, z_{max}]$ is divided into N_b uniform bins. Then, for each bin $i = \overline{1, N_b}$, bounded by the ranges $r_{inf} = z_{inf} + (i-1)\,\Delta$ and $r_{sup} = z_{inf} + i\Delta$, the overlapping range of the two pdfs $p_X(x)$ and $p_Y(z-x)$ must be identified.

As illustrated in Fig. 3.7, the lower bound of the overlap is given by $max(x_{min}, z - y_{max})$, which ranges in the current bin between:

$$z_{low}^{inf} = max(x_{min}, -y_{max} + r_{inf}) \tag{3.11}$$
$$z_{low}^{sup} = max(x_{max}, -y_{max} + r_{sup}) \tag{3.12}$$

Similarly, the upper bound of the overlapping range varies between:

$$z_{up}^{inf} = min(x_{max}, -y_{min} + r_{inf}) \tag{3.13}$$
$$z_{up}^{sup} = min(x_{max}, -y_{min} + r_{sup}) \tag{3.14}$$

First, the left value of the integral from (3.10) is computed for $z = r_{inf}$, using Simpson's 3/8 rule:

$$Int_l = \frac{z_{up}^{inf} - z_{low}^{inf}}{8}\left[f\left(z_{low}^{inf}\right) + 3f(z_1) + 3f(z_2) + f\left(z_{up}^{inf}\right)\right] \tag{3.15}$$

where $f(z) = p_X(z) \cdot p_Y(r_{inf} - z)$ and the two points z_1 and z_2 separate the overlap region in equal slots:

$$z_1 = z_{low}^{inf} + \left(z_{up}^{inf} - z_{low}^{inf}\right)/3 \tag{3.16}$$
$$z_2 = z_{low}^{inf} + \left(z_{up}^{inf} - z_{low}^{inf}\right) \cdot 2/3 \tag{3.17}$$

After that, the right value of the integral is evaluated, for $z = r_{sup}$:

$$Int_r = \frac{z_{up}^{sup} - z_{low}^{sup}}{8}\left[f(z_{low}^{sup}) + 3f(z_1) + 3f(z_2) + f\left(z_{up}^{sup}\right)\right] \tag{3.18}$$

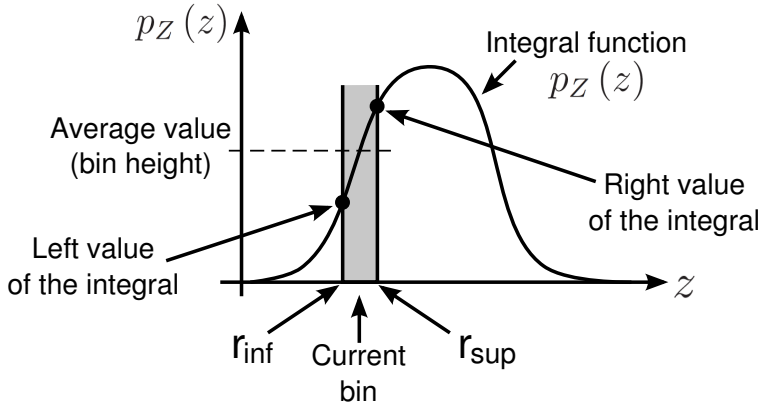

Fig. 3.8: Limits of the overlap during the sum computation.

where $f(z) = p_X(z) \cdot p_Y(r_{sup} - z)$ and the two points z_1 and z_2 are given by:

$$z_1 = z_{low}^{sup} + \left(z_{up}^{sup} - z_{low}^{sup}\right)/3 \tag{3.19}$$

$$z_2 = z_{low}^{sup} + \left(z_{up}^{sup} - z_{low}^{sup}\right) \cdot 2/3 \tag{3.20}$$

As illustrated in Fig. 3.8, the left and right values of the integral correspond to the values of $p_Z(z)$ at the bin boundaries. Finally, the bin height b_i is evaluated as the average value between the left and the right integral values. In addition, to ensure that the resulted pdf satisfies the normalization condition $\sum_{i=1}^{N_b} b_i \Delta_i = 1$, every bin b_i must be multiplied by the factor:

$$f_{i,norm} = \frac{1}{(r_i - r_{i-1}) \sum_{j=1}^{N_b} b_j} \tag{3.21}$$

Fig. 3.9 shows the estimated delay of three processing tasks, evaluated as the sum between earliest starting time and task execution time. Both starting and execution times are estimated as statistical distributions and the total delay has been evaluated using the implemented sum operator. For illustration purposes, the evaluations have been approximated with 30 bins for each random variable. The results are compared with direct Monte Carlo samplings from the distributions using 1 000 000 samples for each variable and are found to be in very good agreement.

For the maximum operation, let W be another random variable computed as $W = \max(X, Y)$. Then, given the CDFs $F_X(x)$ and $F_Y(y)$ of RVs X and Y, respectively, the pdf of W is given by:

$$p_W(w) = F_X(w) \cdot p_Y(w) + F_Y(w) \cdot p_X(w) \tag{3.22}$$

After estimating the CDFs $\widehat{F}_X(x)$ and $\widehat{F}_Y(y)$ using (3.8), the distribution of W can be evaluated for every bin between $\max(x_{min}, y_{min})$ and $\max(x_{max}, y_{max})$.

A rather interesting case is the particular maximum operation between a given random variable X and a constant $Y = c$. This case is often found in performance models

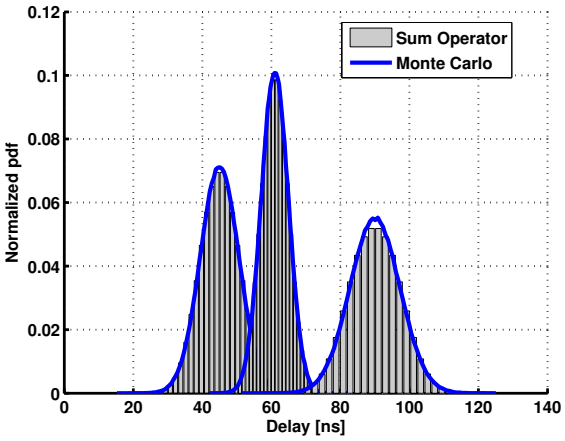

Fig. 3.9: Estimated delay of three processing tasks computed using the sum operator and through Monte Carlo sampling.

where only a subset of parameters are described statistically, the rest being considered constant. In the trivial case where the constant is outside of the support of $p_X(x)$, the result is either the unchanged pdf $p_X(x)$ (if $c \leq x_{min}$), or the constant c (if $c \geq x_{max}$). Otherwise, if the constant is found between x_{min} and x_{max}, the pdf of W will be computed as:

$$p_W(w) = \begin{cases} 0, & w < c \\ F_X(c) + p_X(c), & w = c \\ p_X(w), & w > c \end{cases} \tag{3.23}$$

The first branch in (3.23) is empty, since the lower bound of $p_W(w)$ is given by $\max(x_{min}, c)$. Because the pdf of Y is equal to a Dirac delta delayed by c, the first term in (3.22) becomes $F_X(c)$, as illustrated in Fig. 3.10. Finally, knowing that the CDF of Y is a Heaviside step function delayed by c (as shown in Fig. 3.10), the third branch is identical to the pdf of X.

3.3.2 Statistical Difference Operator

Statistical subtractions are required for computing timing differences, such as timing slacks between the scheduled tasks. Let $Z = X - Y$ be the difference of two random variables. The pdf of Z can be computed using the implementation of the sum operator after transforming the pdf of Y as follows:

$$p_Z(z) = \int_{-\infty}^{\infty} p_X(x) \cdot p'_Y(z - x)\, dx \tag{3.24}$$

$$p'_Y(y) = p_Y(-y) \tag{3.25}$$

Hence, the sum operator can be applied after mirroring the pdf of the subtrahend with respect to the ordinate axis, as shown in Fig. 3.11.

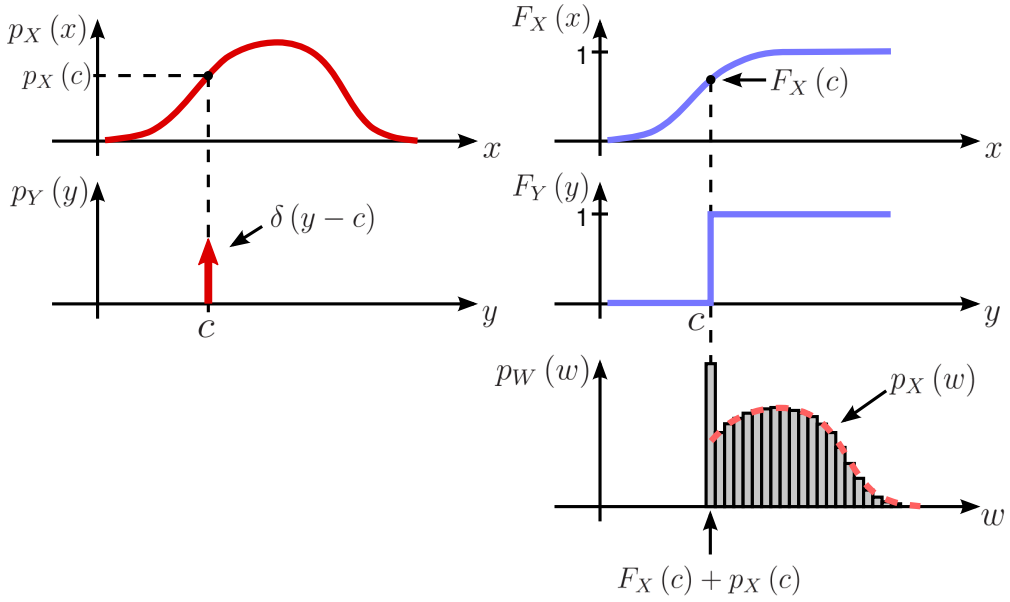

Fig. 3.10: Evaluation of the maximum between a random variable and a constant.

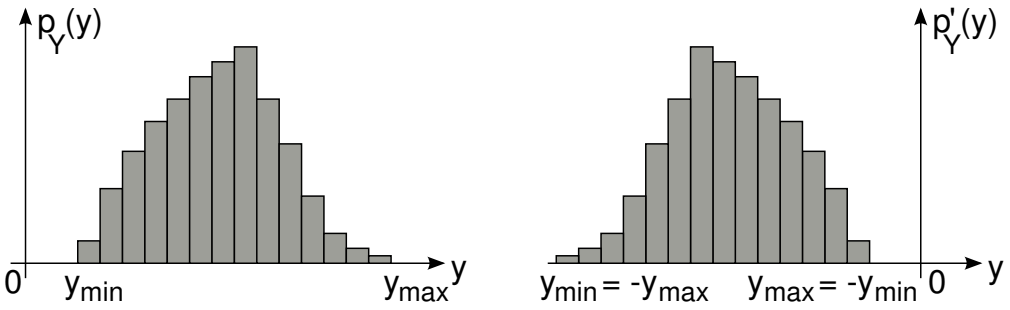

Fig. 3.11: Subtrahend distribution mirrored across the ordinate.

3.3.3 Statistical Product Operator

Another important contribution of this work is the development of an efficient statistical product operator. Statistical sum and maximum operations have received a large attention within the statistical timing analysis, whereas the statistical product operation has not yet been systematically analyzed and implemented for the general case of two arbitrary random distributions. Even though a statistical formulation exists [69], a serious challenge is posed by the integration limits, with many particular cases of repartition across the four quadrants and the inclusion of zero in the distribution support.

First, a brief observation is required with respect to the repartition of variables across the real axis over both positive and negative values. While process parameters usually exhibit positive values, their variations are often modeled with respect to the nominal value as [151]:

$$P_k = P_{k,nom} + \Delta P_{k,inter} + \Delta P_{k,spatial}(x_i, y_i) + \Delta P_{k,random,i} \tag{3.26}$$

In this representation, the nominal value of the parameter $P_{k,nom}$ is positive, whereas

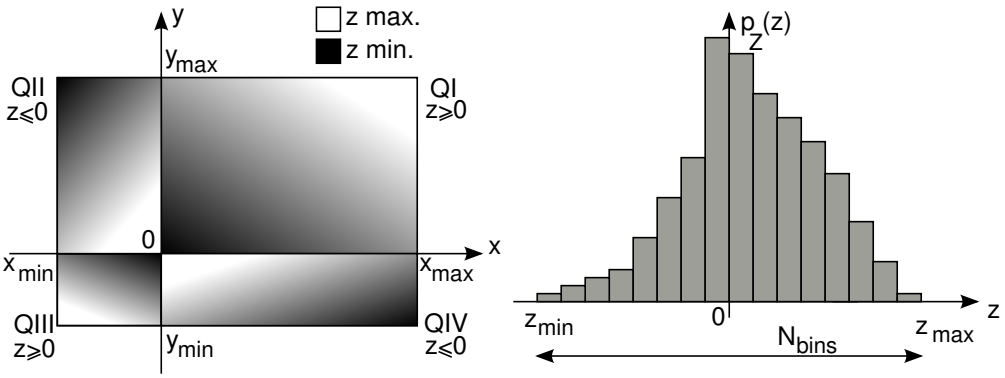

Fig. 3.12: Repartition of X and Y random variables across the four quadrants and discretized pdf of the product $Z = XY$.

random variables like the inter-die variation $\Delta P_{k,inter}$, the intra-die spatially-correlated variation $P_{k,spatial}\left(x_i, y_i\right)$, and the intra-die uncorrelated component $\Delta P_{k,random,i}$ are modeled with random variables usually centered on zero and extending over both positive and negative values.

First, the limits $[z_{min}, z_{max}]$ for the support of $p_Z\left(z\right)$ are determined. As shown in Fig. 3.12, z_{max} can be found as one of the following products: $x_{max}y_{max}$ in the first quadrant (QI), $x_{max}y_{min}$ in the second quadrant (QII), $x_{min}y_{min}$ in the third quadrant (QIII), and $x_{min}y_{max}$ in the fourth quadrant (QIV). Likewise, z_{min} is given by $x_{min}y_{min}$ in QI, $x_{min}y_{max}$ in QII, $x_{max}y_{max}$ in QIII and $x_{max}y_{min}$ in QIV.

More interesting are the cases where the distributions span across more than one quadrant. For spans in QI and QII (Fig. 3.13(a)), $z_{min} = x_{min}y_{max}\left(< 0\right)$ and $z_{max} = x_{max}y_{max}\left(> 0\right)$. In QII+QIII (Fig. 3.13(b)), $z_{min} = x_{min}y_{max}\left(< 0\right)$ and $z_{max} = x_{min}y_{min}\left(> 0\right)$. Similarly, in spans across QIII+QIV the z range is given by $[x_{max}y_{min}, x_{min}y_{min}]$ and in QI+QIV by $[x_{max}y_{min}, x_{max}y_{max}]$. If the distributions span across all four quadrants (as in Fig. 3.13(e)), then $z_{min} = \min\left\{x_{min}y_{max}, x_{max}y_{min}\right\}$ and $z_{max} = \max\left\{x_{max}y_{max}, x_{min}y_{min}\right\}$.

After establishing the limits for the distribution of the product, the domains of X and Y are partitioned onto the four quadrants, resulting into four $\{X, Y\}$ partitions. The product pdf $p_Z\left(z\right)$ is then computed separately on each quadrant and the results are merged afterwards. To do this, the domain between z_{min} and z_{max} is divided into N_b bins. Then, for each bin, the pdf of Z is evaluated as explained in the following.

The general integration formula for the product distribution is given by:

$$p_Z\left(z\right) = \int_{-\infty}^{\infty} p_X\left(x\right) p_Y\left(\frac{z}{x}\right) \frac{1}{|x|} dx \tag{3.27}$$

Thereby, the largest challenge in computing the integral is finding the integration limits.

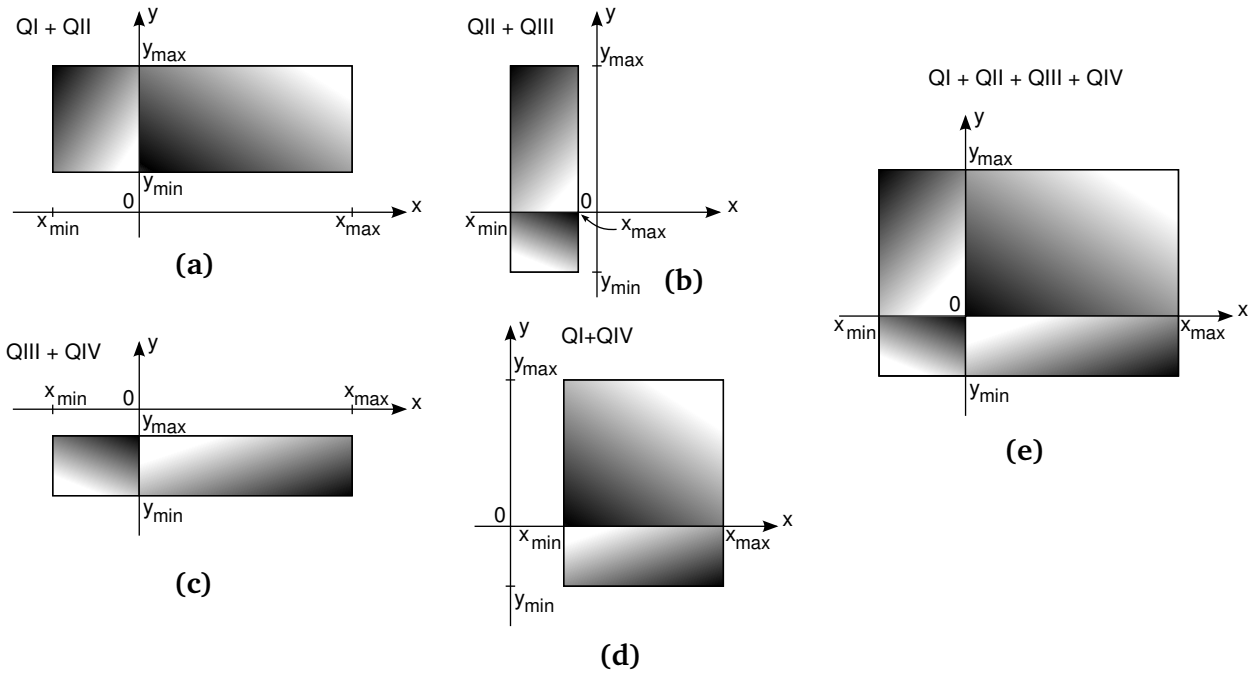

Fig. 3.13: Variable spans across multiple quadrants.

The supports of the two pdfs are limited by the following conditions:

$$\text{For } p_X : x_{min} \leq x \leq x_{max} \tag{3.28}$$

$$\text{For } p_Y : y_{min} \leq \frac{z}{x} \leq y_{max} \tag{3.29}$$

The $\{X, Y\}$ partition in the first quadrant is bounded by $x_{min}^{QI} = \max\{0, x_{min}\}$, $y_{min}^{QI} = \max\{0, y_{min}\}$, $x_{max}^{QI} = \max\{0, x_{max}\}$, and $y_{max}^{QI} = \max\{0, y_{max}\}$. Here, both supports of p_X and p_Y are non-negative and (3.29) becomes:

$$\frac{z}{y_{max}^{QI}} \leq x \leq \frac{z}{y_{min}^{QI}} \tag{3.30}$$

Consequently, the lower and upper limits of integration (L_l and L_u) are given by:

$$L_l = \max\left\{x_{min}^{QI}, \frac{z}{y_{max}^{QI}}\right\} \tag{3.31}$$

$$L_u = \min\left\{x_{max}^{QI}, \frac{z}{y_{min}^{QI}}\right\} \tag{3.32}$$

Three particular cases can be identified. First, if the curve $xy = x_{max}^{QI} y_{min}^{QI}$ which intersects the lower-right corner of the partition is greater than the curve $xy = x_{min}^{QI} y_{max}^{QI}$ which passes through the upper-left corner (Fig. 3.14(a)), then $x_{min}^{QI} y_{min}^{QI} \leq x_{min}^{QI} y_{max}^{QI} < x_{max}^{QI} y_{min}^{QI} <$

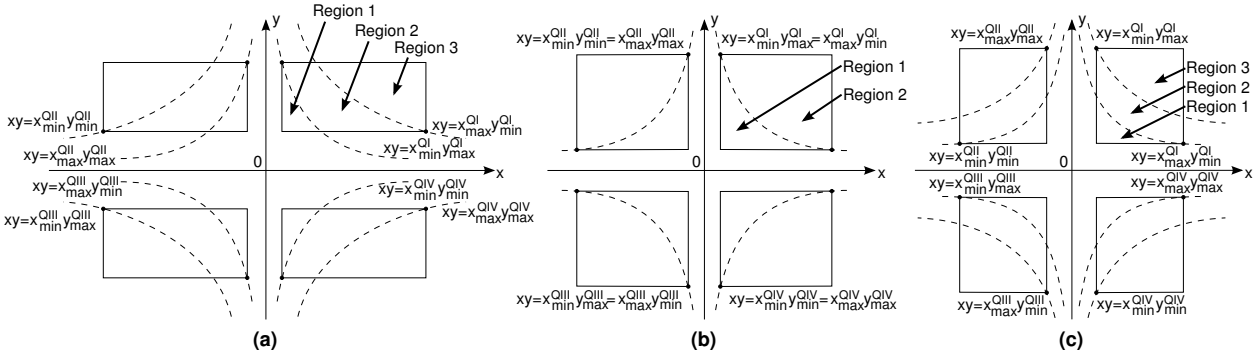

Fig. 3.14: Relative positions of the $\{X, Y\}$ partition corners. Opposite corners on distinct xy curves (a,c) and on the same curve (b).

$x_{max}^{QI} y_{max}^{QI}$. In this case, the product pdf is given by:

$$p_Z(z) = \begin{cases} \displaystyle\int_{x_{min}^{QI}}^{z/y_{min}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QI} y_{min}^{QI} \le z \le x_{min}^{QI} y_{max}^{QI} \text{ (Region 1)} \\ \displaystyle\int_{z/y_{max}^{QI}}^{z/y_{min}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QI} y_{max}^{QI} \le z \le x_{max}^{QI} y_{min}^{QI} \text{ (Region 2)} \\ \displaystyle\int_{z/y_{max}^{QI}}^{x_{max}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QI} y_{min}^{QI} \le z \le x_{max}^{QI} y_{max}^{QI} \text{ (Region 3)} \end{cases} \tag{3.33}$$

In the second case, the partition corners lie on the same curve $xy = x_{min}^{QI} y_{max}^{QI} = x_{max}^{QI} y_{min}^{QI}$ (Fig. 3.14(b)), for which:

$$p_Z(z) = \begin{cases} \displaystyle\int_{x_{min}^{QI}}^{z/y_{min}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QI} y_{min}^{QI} \le z \le x_{min}^{QI} y_{max}^{QI} \text{ (Region 1)} \\ \displaystyle\int_{z/y_{max}^{QI}}^{x_{max}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QI} y_{max}^{QI} \le z \le x_{max}^{QI} y_{max}^{QI} \text{ (Region 2)} \end{cases} \tag{3.34}$$

The third case occurs for $x_{max}^{QI} y_{min}^{QI} < x_{min}^{QI} y_{max}^{QI}$ (Fig. 3.14(c)) and:

$$p_Z(z) = \begin{cases} \displaystyle\int_{x_{min}^{QI}}^{z/y_{min}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QI} y_{min}^{QI} \le z \le x_{max}^{QI} y_{min}^{QI} \text{ (Region 1)} \\ \displaystyle\int_{x_{min}^{QI}}^{x_{max}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QI} y_{min}^{QI} \le z \le x_{min}^{QI} y_{max}^{QI} \text{ (Region 2)} \\ \displaystyle\int_{z/y_{max}^{QI}}^{x_{max}^{QI}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QI} y_{max}^{QI} \le z \le x_{max}^{QI} y_{max}^{QI} \text{ (Region 3)} \end{cases} \tag{3.35}$$

In the second quadrant, bounded by $x_{min}^{QII} = min\{x_{min}, 0\}$, $y_{min}^{QII} = max\{0, y_{min}\}$, $x_{max}^{QII} = min\{x_{max}, 0\}$, and $y_{max}^{QII} = max\{0, y_{max}\}$, the values of Z are negative and (3.29) is replaced by:

$$\frac{z}{y_{min}^{QII}} \le x \le \frac{z}{y_{max}^{QII}} \tag{3.36}$$

The lower and upper integration limits are evaluated as:

$$L_l = \max\left\{x_{min}^{QII}, \frac{z}{y_{min}^{QII}}\right\} \tag{3.37}$$

$$L_u = \min\left\{x_{max}^{QII}, \frac{z}{y_{max}^{QII}}\right\} \tag{3.38}$$

If $x_{min}^{QII}y_{max}^{QII} < x_{min}^{QII}y_{min}^{QII} < x_{max}^{QII}y_{max}^{QII} \leq x_{max}^{QII}y_{min}^{QII}$ (Fig. 3.14(a)), the pdf of the product is computed in the following way:

$$p_Z(z) = \begin{cases} \displaystyle\int_{x_{min}^{QII}}^{z/y_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{min}^{QII}y_{max}^{QII} \leq z \leq x_{min}^{QII}y_{min}^{QII} \text{ (Region 3)} \\ \displaystyle\int_{z/y_{min}^{QII}}^{z/y_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{min}^{QII}y_{min}^{QII} \leq z \leq x_{max}^{QII}y_{max}^{QII} \text{ (Region 2)} \\ \displaystyle\int_{z/y_{min}^{QII}}^{x_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{max}^{QII}y_{max}^{QII} \leq z \leq x_{max}^{QII}y_{min}^{QII} \text{ (Region 1)} \end{cases} \tag{3.39}$$

If $x_{min}^{QII}y_{min}^{QII} = x_{max}^{QII}y_{max}^{QII}$ (Fig. 3.14(b)), the pdf is computed as:

$$p_Z(z) = \begin{cases} \displaystyle\int_{x_{min}^{QII}}^{z/y_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{min}^{QII}y_{max}^{QII} \leq z \leq x_{min}^{QII}y_{min}^{QII} \text{ (Region 2)} \\ \displaystyle\int_{z/y_{min}^{QII}}^{x_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{min}^{QII}y_{min}^{QII} \leq z \leq x_{max}^{QII}y_{max}^{QII} \text{ (Region 1)} \end{cases} \tag{3.40}$$

Finally, in the case in which $x_{max}^{QII}y_{max}^{QII} < x_{min}^{QII}y_{min}^{QII}$ (Fig. 3.14(c)), the distribution is given by:

$$p_Z(z) = \begin{cases} \displaystyle\int_{x_{min}^{QII}}^{z/y_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{min}^{QII}y_{max}^{QII} \leq z \leq x_{max}^{QII}y_{max}^{QII} \text{ (Region 3)} \\ \displaystyle\int_{x_{min}^{QII}}^{x_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{max}^{QII}y_{max}^{QII} \leq z \leq x_{min}^{QII}y_{min}^{QII} \text{ (Region 2)} \\ \displaystyle\int_{z/y_{min}^{QII}}^{x_{max}^{QII}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{(-x)}dx, & x_{min}^{QII}y_{min}^{QII} \leq z \leq x_{max}^{QII}y_{min}^{QII} \text{ (Region 1)} \end{cases} \tag{3.41}$$

In the third quadrant the bounds of the $\{X, Y\}$ partition are $x_{min}^{QIII} = \min\{x_{min}, 0\}$, $y_{min}^{QIII} = \min\{y_{min}, 0\}$, $x_{max}^{QIII} = \min\{x_{max}, 0\}$, and $y_{max}^{QIII} = \min\{y_{max}, 0\}$. Since $z > 0$, the inequality (3.29) becomes:

$$\frac{z}{y_{max}^{QIII}} \leq x \leq \frac{z}{y_{min}^{QIII}} \tag{3.42}$$

Hence, the integration limits are:

$$L_l = \max\left\{x_{min}^{QIII}, \frac{z}{y_{max}^{QIII}}\right\} \tag{3.43}$$

$$L_u = \min\left\{x_{max}, \frac{z}{y_{min}^{QIII}}\right\} \tag{3.44}$$

Here, the distribution is computed if $x_{max}^{QIII} y_{max}^{QIII} \leq x_{max}^{QIII} y_{min}^{QIII} < x_{min}^{QIII} y_{max}^{QIII} < x_{min}^{QIII} y_{min}^{QIII}$ (Fig. 3.14(a)) as:

$$p_Z(z) = \begin{cases} \int_{z/y_{max}^{QIII}}^{x_{max}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{max}^{QIII} y_{max}^{QIII} \leq z \leq x_{max}^{QIII} y_{min}^{QIII} \text{ (Region 1)} \\ \int_{z/y_{max}^{QIII}}^{z/y_{min}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{max}^{QIII} y_{min}^{QIII} \leq z \leq x_{min}^{QIII} y_{max}^{QIII} \text{ (Region 2)} \\ \int_{x_{min}^{QIII}}^{z/y_{min}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{min}^{QIII} y_{max}^{QIII} \leq z \leq x_{min}^{QIII} y_{min}^{QIII} \text{ (Region 3)} \end{cases} \tag{3.45}$$

In the second case $x_{max}^{QIII} y_{min}^{QIII} = x_{min}^{QIII} y_{max}^{QIII}$ (Fig. 3.14(b)) and the distribution becomes:

$$p_Z(z) = \begin{cases} \int_{z/y_{max}^{QIII}}^{x_{max}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{max}^{QIII} y_{max}^{QIII} \leq z \leq x_{max}^{QIII} y_{min}^{QIII} \text{ (Region 1)} \\ \int_{x_{min}^{QIII}}^{z/y_{min}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{max}^{QIII} y_{min}^{QIII} \leq z \leq x_{min}^{QIII} y_{min}^{QIII} \text{ (Region 2)} \end{cases} \tag{3.46}$$

The third case occurs for $x_{min}^{QIII} y_{max}^{QIII} < x_{max}^{QIII} y_{min}^{QIII}$ (Fig. 3.14(c)) and the pdf is evaluated as:

$$p_Z(z) = \begin{cases} \int_{z/y_{max}^{QIII}}^{x_{max}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{max}^{QIII} y_{max}^{QIII} \leq z \leq x_{min}^{QIII} y_{max}^{QIII} \text{ (Region 1)} \\ \int_{x_{min}^{QIII}}^{x_{max}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{min}^{QIII} y_{max}^{QIII} \leq z \leq x_{max}^{QIII} y_{min}^{QIII} \text{ (Region 2)} \\ \int_{x_{min}^{QIII}}^{z/y_{min}^{QIII}} p_X(x)\, p_Y\left(\frac{z}{x}\right) \frac{1}{(-x)} dx, & x_{max}^{QIII} y_{min}^{QIII} \leq z \leq x_{min}^{QIII} y_{min}^{QIII} \text{ (Region 3)} \end{cases} \tag{3.47}$$

Finally, in the fourth quadrant $x_{min}^{QIV} \geq 0$ and $y_{max}^{QIV} \leq 0$, hence $z < 0$ and the relationship (3.29) becomes:

$$\frac{z}{y_{min}^{QIV}} \leq x \leq \frac{z}{y_{max}^{QIV}} \tag{3.48}$$

As a consequence, the lower and upper limits of the integral are computed as:

$$L_l = \max\left\{ x_{min}^{QIV}, \frac{z}{y_{min}^{QIV}} \right\} \tag{3.49}$$

$$L_u = \min\left\{ x_{max}^{QIV}, \frac{z}{y_{max}^{QIV}} \right\} \tag{3.50}$$

Hence, the following cases of repartition can be distinguished in QIV. First, if $x_{max}^{QIV} y_{min}^{QIV} <$

$x_{max}^{QIV} y_{max}^{QIV} < x_{min}^{QIV} y_{min}^{QIV} \leq x_{min}^{QIV} y_{max}^{QIV}$ (Fig. 3.14(a)), the pdf becomes:

$$p_Z(z) = \begin{cases} \int_{z/y_{min}^{QIV}}^{x_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QIV} y_{min}^{QIV} \leq z \leq x_{max}^{QIV} y_{max}^{QIV} \text{ (Region 3)} \\ \int_{z/y_{min}^{QIV}}^{z/y_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QIV} y_{max}^{QIV} \leq z \leq x_{min}^{QIV} y_{min}^{QIV} \text{ (Region 2)} \\ \int_{x_{min}^{QIV}}^{z/y_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QIV} y_{min}^{QIV} \leq z \leq x_{min}^{QIV} y_{max}^{QIV} \text{ (Region 1)} \end{cases} \tag{3.51}$$

Second, if the opposite corners lie on the same xy curve ($x_{max}^{QIV} y_{max}^{QIV} = x_{min}^{QIV} y_{min}^{QIV}$ as in Fig. 3.14(b)), the product pdf is computed as:

$$p_Z(z) = \begin{cases} \int_{z/y_{min}^{QIV}}^{x_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QIV} y_{min}^{QIV} \leq z \leq x_{max}^{QIV} y_{max}^{QIV} \text{ (Region 2)} \\ \int_{x_{min}^{QIV}}^{z/y_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QIV} y_{max}^{QIV} \leq z \leq x_{min}^{QIV} y_{max}^{QIV} \text{ (Region 1)} \end{cases} \tag{3.52}$$

Third, if the opposite corners lie on reversed xy curves ($x_{min}^{QIV} y_{min}^{QIV} < x_{max}^{QIV} y_{max}^{QIV}$, see Fig. 3.14(c)), the integration limits are as follows:

$$p_Z(z) = \begin{cases} \int_{z/y_{min}^{QIV}}^{x_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QIV} y_{min}^{QIV} \leq z \leq x_{min}^{QIV} y_{min}^{QIV} \text{ (Region 3)} \\ \int_{x_{min}^{QIV}}^{x_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{min}^{QIV} y_{min}^{QIV} \leq z \leq x_{max}^{QIV} y_{max}^{QIV} \text{ (Region 2)} \\ \int_{x_{min}^{QIV}}^{z/y_{max}^{QIV}} p_X(x)\, p_Y\left(\frac{z}{x}\right)\frac{1}{x}dx, & x_{max}^{QIV} y_{max}^{QIV} \leq z \leq x_{min}^{QIV} y_{max}^{QIV} \text{ (Region 1)} \end{cases} \tag{3.53}$$

A particular case occurs for $z = 0$, where the integral is not always defined (particularly if $x = 0$). Since this case occurs only if $x = 0 \vee y = 0$, the pdf is computed separately, such that:

$$\int_{0-\varepsilon}^{0+\varepsilon} p_Z(z) = \int_{0-\varepsilon}^{0+\varepsilon} p_X(x)\, dx + \int_{0-\varepsilon}^{0+\varepsilon} p_Y(y)\, dy \tag{3.54}$$

in the neighborhood of 0, for an arbitrary small value ε.

This completes the definition of $p_Z(z)$ for all real values across the four quadrants. It is important to notice, that multiplications in separate quadrants result into overlapping regions on the z axis. This is especially the case when the joint $\{X, Y\}$ distribution spans across all four quadrants. If the distribution span covers only two quadrants (see Fig. 3.13(a–d)), the values of z from one of them are always negative, while the values from the other are positive, hence their coverages on the z axis are disjoint. Suppose the joint distribution $\{X, Y\}$ spans across all four quadrants, as in Fig. 3.13(e). In this case, the

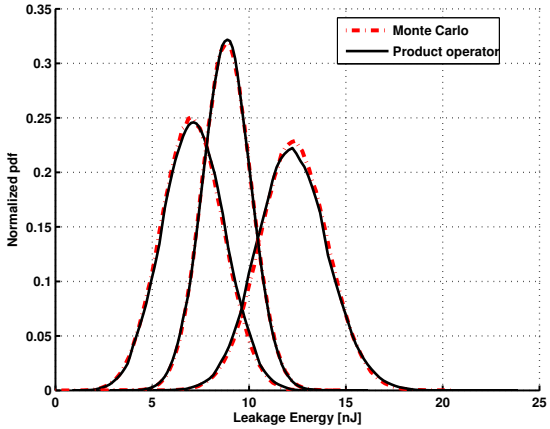

Fig. 3.15: Leakage energy distributions for three slacks, computed using the product operator and through direct sampling (Monte Carlo).

components from the first and third quadrants result into overlapping positive regions, given by:

$$z \in [0, x_{max}y_{max}] \text{ from QI} \tag{3.55}$$
$$z \in [0, x_{min}y_{min}] \text{ from QIII} \tag{3.56}$$

Likewise, the negative components resulting from the second and fourth quadrants are:

$$z \in [x_{min}y_{max}, 0] \text{ from QII} \tag{3.57}$$
$$z \in [x_{max}y_{min}, 0] \text{ from QIV} \tag{3.58}$$

Since each of the overlapping components contributes to the final distribution of Z, the pdf segments computed for the overlapping branches must be added.

Considering the previous observations, the algorithm for computing the product distribution is given in Listing 3.1. As an example, Fig. 3.15 shows the result from a performance macromodel subset which multiplies three slack distributions between processing tasks with the estimated average leakage power of the core on which the tasks are running. The average leakage power has been estimated in a discrete form and varies between 266 and 321 mW for the given core. The results of the implemented product operator are compared with direct Monte Carlo samplings and multiplications. As shown in Fig. 3.15, the results are in a very good agreement. The implemented operator has also been tested on variables spanning across multiple quadrants and the results show the same accuracy level.

3.3.4 Numerical Implementation of other Statistical Operators

The previous sections have presented the development of a series of analytic operators for the propagation of statistical distributions. These operators offer the important advantage

PRODUCTOPERATOR(X,Y)

Compute $z_{min},\ z_{max}$
Divide $[z_{min}, z_{max}]$ into N_b bins
for $i \leftarrow 1$ **to** N_b
do /* Retrieve the bin ranges: */
  $z_{inf} \leftarrow z_{min} + (i-1)\,\frac{z_{max}-z_{min}}{N_b}$;
  $z_{sup} \leftarrow z_{min} + i \cdot \frac{z_{max}-z_{min}}{N_b}$;
  /* Compute $p_Z(z_{inf})$: */
  $p_Z(z_{inf}) \leftarrow 0$; /* Reset to zero */
  if $z_{inf} = 0$
    then $p_Z(z_{inf}) \leftarrow p_X(0) + p_Y(0)$;
    else if $z_{inf} < 0$
      then /* Check QII and QIV: */
        if QII_span= $true$ /*$\{X, Y\}$ spans over QII: */
          then /* Compute integration limits: */
            $L_l = \max\left\{x_{min}^{QII}, \frac{z_{inf}}{y_{min}^{QII}}\right\}$;
            $L_u = \min\left\{x_{max}^{QII}, \frac{z_{inf}}{y_{max}^{QII}}\right\}$;
            if $L_l = L_u$
              then $p_Z(z_{inf}) += 0$; /* Integral is zero */
              else /* Apply Simpson's 3/8 rule: */
                $x_0 \leftarrow L_l$;
                $x_1 \leftarrow L_l + (L_u - L_l)/3$;
                $x_2 \leftarrow L_l + (L_u - L_l) \cdot 2/3$;
                $x_3 \leftarrow L_u$;
                Compute $f(x) = p_X(x)\, p_Y\left(\frac{z_{inf}}{x}\right)\frac{1}{(-x)}$ in $x_0,\ x_1,\ x_2$, and x_3;
                $p_Z(z_{inf}) += (L_u - L_l)\,\frac{f(x_0)+3f(x_1)+3f(x_2)+f(x_3)}{8}$;

        if QIV_span= $true$ /*$\{X, Y\}$ spans over QIV: */
          then Get limits and evaluate the integral similarly.

      else /* $z_{inf} > 0$: check QI and QIII */
        Evaluate $p_z(z_{inf})$ in QI and QIII.

  Compute $p_Z(z_{sup})$ in a similar way.

  /* Compute the average value of the current bin: */
  $p_Z[i] = \frac{p_Z(z_{inf})+p_Z(z_{sup})}{2}$;

/* Check the normalization condition for $p_Z(z)$: */
$S_b = \sum_{i=1}^{N_b} p_Z[i]$;
for $i \leftarrow 1$ **to** N_b
do $p_Z[i] *= \frac{N_b}{S_b(z_{max}-z_{min})}$;

Listing 3.1: Algorithm for computing the product pdf.

of both high accuracy and high speed and, in addition, they are implemented for the most often used operations in the macromodels.

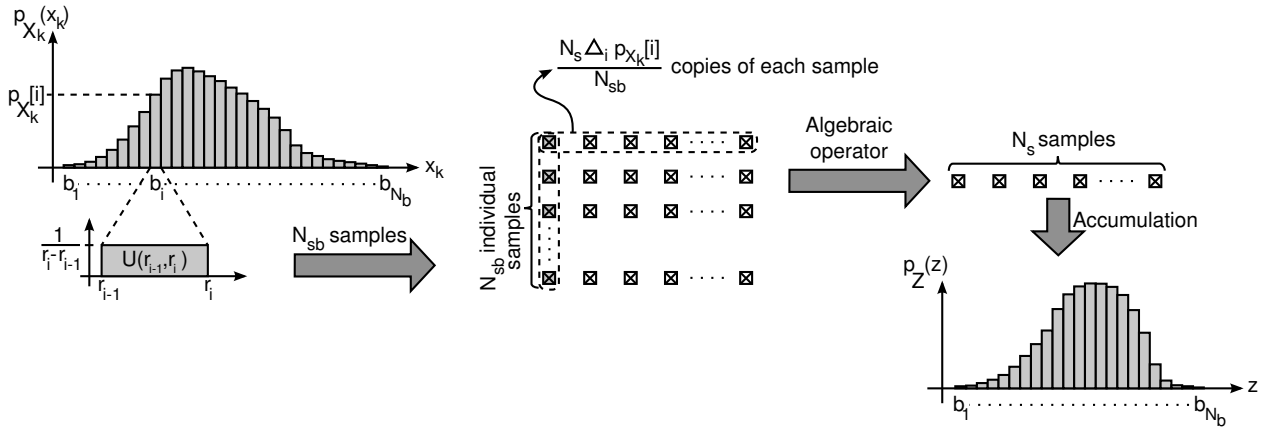

Fig. 3.16: Fast numerical implementation with adjustable accuracy.

Nevertheless, additional statistical operations are required in performance modeling, such as exponential, logarithm, n-th root, or even division, for which no general analytical expressions exist. For instance, the general case of division of two random variables results often in unknown, mostly Pareto-like (heavy-tailed) distributions. Only a few very particular cases [106, 131] are known, like e.g. the ratio of two normal RVs with zero mean, which is a Cauchy distribution (also long-tailed, for which parameters like mean or variance are undefined).

One of the main goals of this work is the development of a general methodology which permits the application of any algebraic operation on a distribution, in order to easily include further extensions to the developed models. To achieve this goal, this work extends the implementation of the previous analytic operators with numerical approximations of several other algebraic operations. Several adjusting parameters which influence the accuracy are discussed and the precision of the numerical operators is analyzed.

Let $\Psi(\cdot)$ be an arbitrary algebraic operator (other than the operators implemented in the previous sections) defined as:

$$\Psi : \mathbb{P}^n \rightarrow \mathbb{P} \tag{3.59}$$

$$\mathbb{P} = \left\{ p_X(x) : \mathbb{R} \rightarrow \mathbb{R}_0^+ \middle| \int_{-\infty}^{\infty} p_X(x)\, dx = 1 \right\} \tag{3.60}$$

where n is the number of statistical distributions on which it is applied and $\mathbb{P}$ is the set of real probability density functions.

If $Z = \Psi(p_{X_1}(x_1), \ldots, p_{X_n}(x_n))$ is the result of the operator, then the pdf $p_Z(z)$ is estimated from N_s samples obtained numerically. As illustrated in Fig. 3.16, the input pdf of each RV $X_k,\ k = \overline{1,n}$ is represented in discrete form over N_b bins, where each bin b_i is approximated with a uniform distribution $U(r_{i-1}, r_i)$. After that, N_{sb} individual samples are generated from each uniform distribution and copied f times, where f is a factor dependent on the respective bin area and is computed as:

$$f = \frac{N_s \cdot \Delta_i \cdot p_{X_k}[i]}{N_{sb}} \tag{3.61}$$

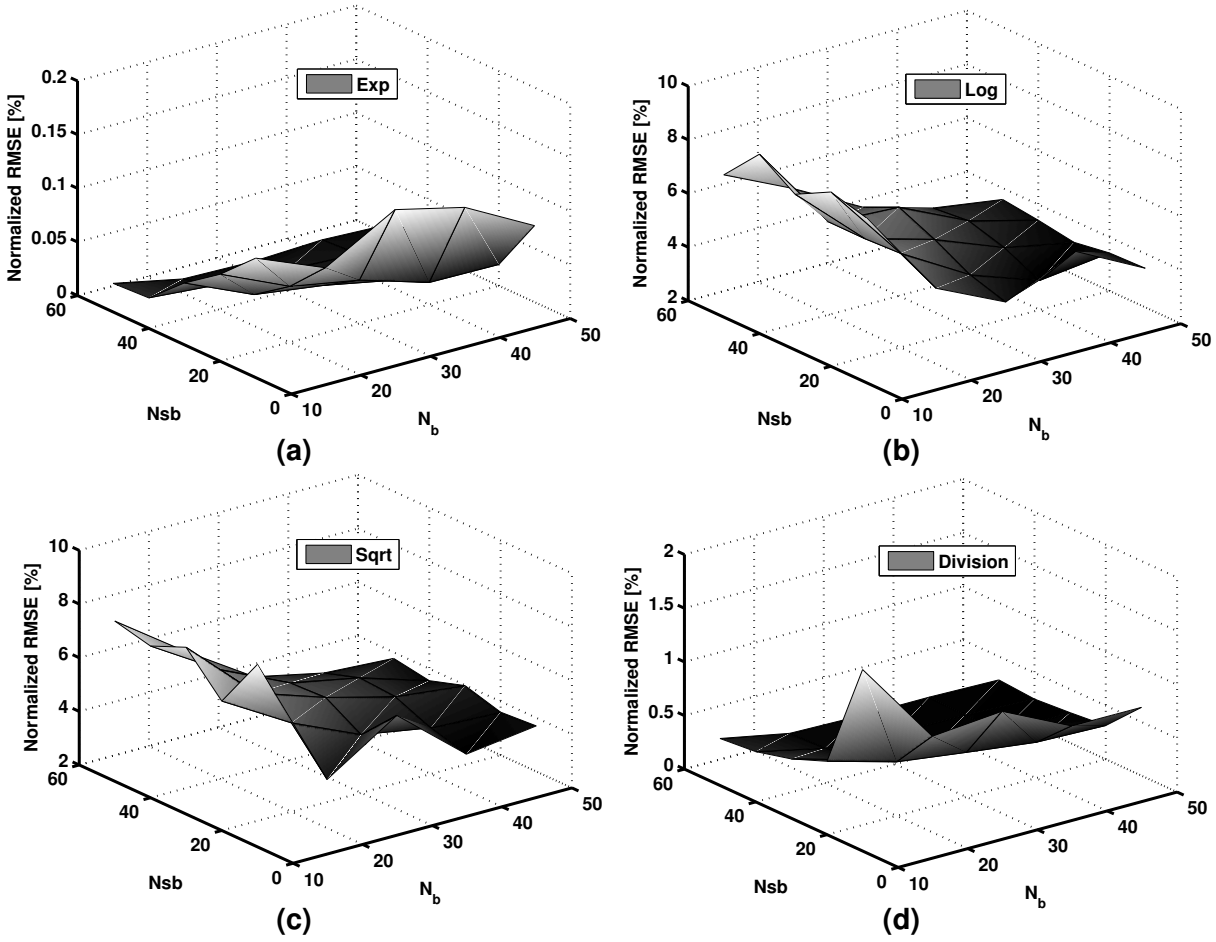

Fig. 3.17: Accuracy of the implemented statistical operators for several values of N_b and N_{sb}.

where $p_{X_k}[i]$ is the height of bin b_i for RV X_k. In this way, the algebraic operation described by $\Psi(\cdot)$ must be applied only on the individual N_{sb} samples from each bin. The resulting N_s samples obtained after applying the operation on all n variables are accumulated into the discretized pdf $p_Z(z)$, which represents the estimated distribution of the result.

First, it is important to notice the performance improvement with respect to Monte Carlo sampling. Due to the fact that the samples from each bin are multiplied by the factor f, this technique requires only $N_b \cdot N_{sb}$ samples of each input variable, while pure Monte Carlo needs N_s samplings. The speedup achieved by this method with respect to Monte Carlo (MC) is given by the relative computational overhead:

$$overhead = \frac{N_b \cdot N_{sb}}{N_s} \cdot 100\ [\%]\ \text{MC} \tag{3.62}$$

The accuracy of this method is controlled by adjusting the number of bins N_b, the number of individual samples extracted from each bin N_{sb}, and the total number of desired samples N_s. In addition, by balancing the ratio of $N_b N_{sb}$ to N_s, a good tradeoff between speed and accuracy can be achieved.

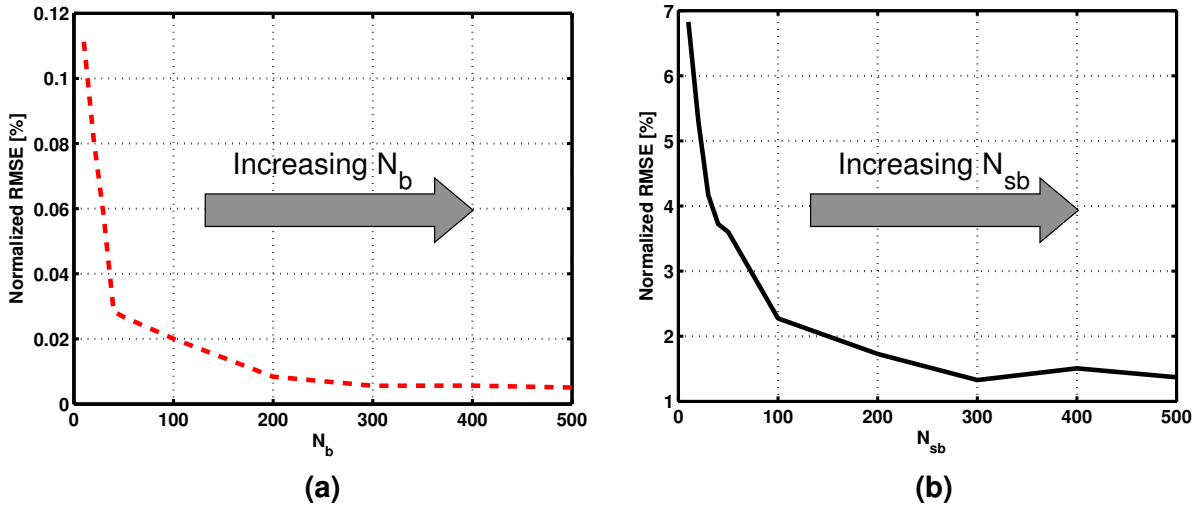

Fig. 3.18: Impact of increasing the number of bins N_b or individual samples N_{sb} on operator accuracy.

The impact of various settings on the accuracy is illustrated in Fig. 3.17 for several operations. Here, the total number of samples N_s was set to 1 000 000 and each statistical operator has been applied on 30 random distributions from which the results are averaged. The obtained pdfs have been compared with the pdfs obtained from 1 000 000 Monte Carlo evaluations (samples from the input distributions on which the algebraic operations are applied) and the achieved accuracy is evaluated using the normalized RMS error, computed as:

$$NRMSE = \frac{RMSE}{z_{max} - z_{min}} \; [\%] \tag{3.63}$$

where the RMS error is evaluated from the differences between the bin heights of the estimated pdfs using the statistical operators ($p_{Z,stat}$) and Monte Carlo ($p_{Z,MC}$) as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_b} \left(p_{Z,stat}\,[i] - p_{Z,MC}\,[i] \right)^2}{N_b}} \tag{3.64}$$

The higher accuracy of the exponential operator with respect to the other investigated operations is due to the very large values which result from exponentiation, and, as a consequence, to the large output span ($z_{max} - z_{min}$) which reduces the normalized error. In addition, the the change in N_b has a relatively higher impact on the NRMSE than the change in N_{sb} due to the computation of the RMS error over the number of bins.

The results show that good accuracies can be obtained already with values for N_b and N_{sb} between 30÷50. With $N_b = 50$ and $N_s = 50$, the relative computational overhead achieved by these statistical operators with respect to Monte Carlo is 0.25 % (for the selected number of 1 000 000 samples), as indicated by (3.62).

Further increasings in the number of bins or in the number of individual samples result into moderate improvements in the accuracy, as shown in Fig. 3.18. The most im-

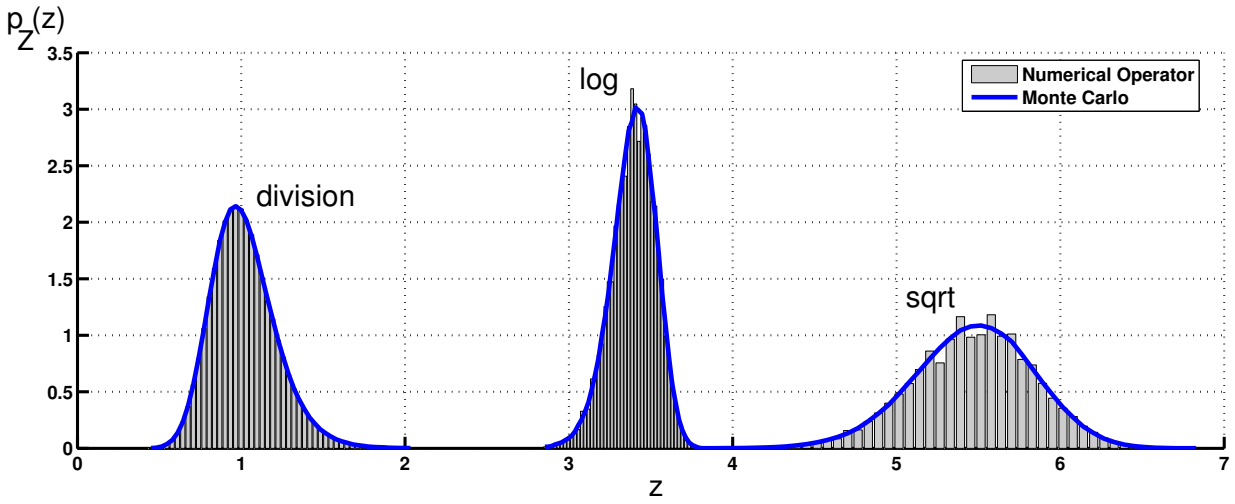

Fig. 3.19: Pdfs obtained for $N_b = 50$ and $N_{sb} = 50$ compared with Monte Carlo for different statistical operators.

portant gain in accuracy is achieved for values of N_b and N_{sb} up to 50÷100, as shown here for the exponential operator (Fig. 3.18(a)) and the square root operator (Fig. 3.18(b)).

Fig. 3.19 offers a better image of the relative accuracy achieved by the implemented operators with respect to Monte Carlo, showing the resulting pdfs for $N_b = 50$ and $N_{sb} = 50$ in the case of logarithm, square root, and division operations (the results of the exponential are orders of magnitude larger). The results show a very good accuracy obtained by the presented method with the estimated overhead of 0.25 % with respect to Monte Carlo sampling.

3.3.5 Handling Correlations

As discussed in Sec. 2.4.2, topological correlations generated by reconverging paths must be carefully tracked. The significant impact of correlations on the statistical computations is illustrated in Fig. 3.20 for the maximum delay computed from multiple paths with different correlations. With increasing correlation, the maximum delay distribution exhibits an increasing skewness. If a confidence point of e.g. 99 % is chosen for evaluating the delay, under the influence of correlations the position of this point may move significantly, leading to underestimations of the actual delay.

To account for topological correlations, each reconvergent node (Fig. 3.21(a)) must be identified first. Nodes with multiple inputs are tested as potential sinks for reconverging paths by recursively collecting the parents of each inbound node in individual lists (see Fig. 3.21(b)). If the lists are disjoint, then the node is not reconvergent, and the maximum operation is computed as described in Sec. 3.3.1. Otherwise, if the lists contain at least one common parent, then the node is a sink for reconverging paths. In this case, a lower and an upper bound of the statistical distribution are computed. The upper bound is computed as a maximum operation (Sec. 3.3.1), whereas the lower bound is computed as

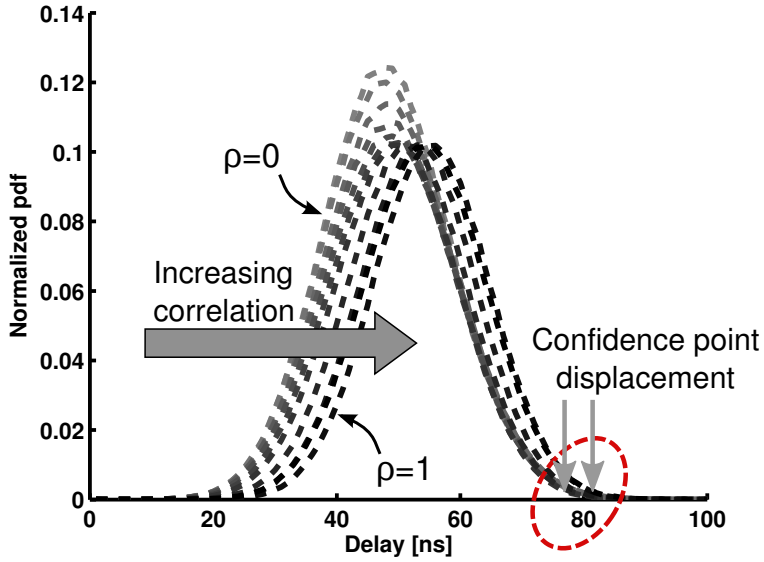

Fig. 3.20: Influence of correlations on statistical result distributions (example for maximum operator).

the minimum between the CDFs of the inbound distributions:

$$p_Z^{low}(z) = \frac{d}{dz}\widehat{F}_Z^{low}(z) = \frac{d}{dz}\left[\min\left(\widehat{F}_{X_1}(z), \widehat{F}_{X_2}(z), \ldots\right)\right] \tag{3.65}$$

where X_i are the inbound RVs and the CDFs $\widehat{F}_{X_i}(x_i)$ are computed according to (3.8). These bounds provide a first estimation of the influence of topological correlations. A further refinement of the estimation between the bounds can be employed for increasing the accuracy, as described by the method proposed in [6].

For numerical operators, topological correlations are tracked in a similar way. After identifying reconvergent nodes, given the order of the first common parent between two reconverging paths, a correlation coefficient is estimated. A first-order common parent generates a total correlation $\rho = 1$ between its output nodes. After each operation, the correlation coefficient decreases according to a piecewise-linear model:

$$\rho(n) = \begin{cases} 1 - \dfrac{n}{N_L}(1 - \rho_r), & n \leq N_L \\ \rho_r, & n > N_L \end{cases} \tag{3.66}$$

where n is the number of statistical operations after the common parent output, N_L is the number of operations over which the correlation coefficient decreases linearly, and ρ_r is a residual correlation in the long reconverging paths. In typical cases N_L ranges from 10 to 20 nodes and ρ_r is between 0.2 and 0.4.

After estimating the correlation coefficient, samples from two correlated random variables X and Y can be obtained by defining two new variables A and B computed using

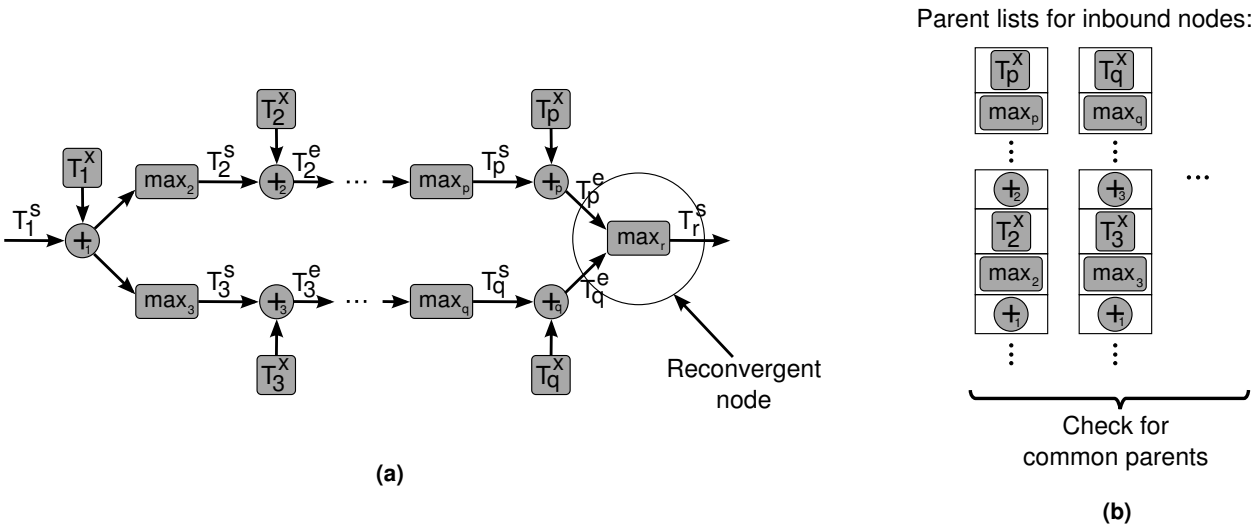

Fig. 3.21: Topological correlations at reconvergent nodes (a) tracked by testing inbound nodes for common parents (b).

a standard normal RV C as follows [100]:

$$A = X + \sqrt{\rho \cdot E\{X^2\}} \cdot C \tag{3.67}$$

$$B = Y + \sqrt{\rho \cdot E\{Y^2\}} \cdot C \tag{3.68}$$

In this way, correlated samples of A and B can be computed from the samples of the uncorrelated variables X, Y, and C.

In addition to topological correlations, the spatial correlations between intra-die parameters must be accounted. In this work a principal component analysis (PCA) method is employed to express spatially-correlated RVs in terms of independent standard normal variables. For instance, if the drain current of a transistor is expressed as a function of process parameters as $I_d = f(P_1, P_2, \dots)$, after the PCA decorrelation the expression becomes:

$$I_d = f(f_1(N_1, N_2, \dots, N_n), \dots) \tag{3.69}$$

where $N_1, \dots, N_n$ are a set of uncorrelated standard Gaussian RVs, representing the principal components of the decomposition. The employed PCA method is described in detail in Sec. 4.1.2.

3.3.6 Random Variable Algebra

The statistical operators developed in Sec. 3.3.1–3.3.4 are implemented within a random variable algebra class. This RV algebra module computes the required statistical operations on a list of input random variables. Within the framework, the random variables are represented as objects corresponding to each of the implemented RV type discussed in Sec.3.2.

Several settings are available to control the behavior of the algebraic operations, which include:

- Parameters for the precision of the RV representation, such as the number of bins N_b, the number of individual (N_{sb}) and total samples (N_s) for the numerical operators, and also margins for the estimation of the pdf support from a limited number of samples;
- Parameters for the embedded random number generator, such as the generator type (see Sec. 3.2.1), seed, and methods for seeding the generator to an arbitrary time-dependent value;
- Settings for the estimation of the moments of random variables, such as the number of samples used in moment estimations.

The algebra module applies the required statistical operation on the list of input RVs and provides the result in the form of a new random variable which embeds the complete pdf representation of the statistical result. During algebraic operations the module automatically recognizes particular cases, such as operations with constants, as well as standard embedded RV types (Gaussian, uniform etc.) which lead to several simplifications in the applied operations. Through the use of a central RV algebra module, all the precision and performance tradeoffs can be adjusted efficiently during the synthesis.

3.4 Embedding Technique for Random Variables

Variability-aware performance macromodels are developed in this work by embedding the RV models discussed in Sec. 3.2 within macromodel data structures. Global performance macromodels, as well as smaller circuit-level models, are represented as linked graphs containing parametric and computational nodes.

As discussed in Sec. 2.4, due to parameter variability considerations, each node in the graph embeds a RV representation. This random variable describes either a parameter in the case of parametric nodes (e.g. estimated execution time T^{x}_{i,RT_j}, dynamic power consumption $P^{RT_j}_{d,i}$, leakage P_{l,RT_i}, communication load $L^{i,j}_{c}$, or process parameter variations), or the result of a statistical operation in the case of operational nodes. This section discusses the various cases in which the random variables are embedded within the performance macromodels developed throughout this thesis.

3.4.1 Variability Sources and RV Leaf Nodes

The leaf nodes in a performance model represent typically parameter values, whereas internal nodes specify operations. Parametric nodes specify either system-level parameters

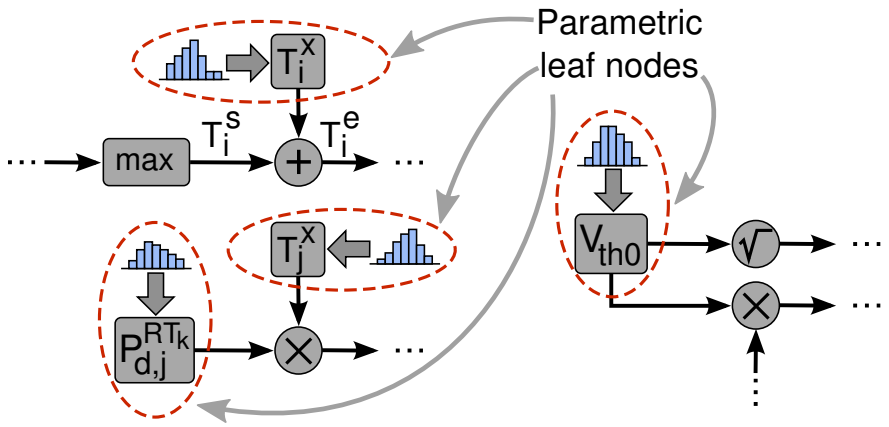

Fig. 3.22: Random variable representations embedded into the leaf nodes of performance models for variable parameters.

estimated in the profiling step, such as execution times, communication loads, and average power dissipation, or technology-related parameters, such as the process parameter variations. In both cases, the parameters exposed to variations are represented as random variables. Fig. 3.22 illustrates the representation of parametric leaf nodes as random variables which embed the complete pdf.

For this purpose, the RV models developed in Sec. 3.2 are employed to represent the values stored in each PM node. As a consequence, every parametric node embeds a RV representation of the parameter value, to provide a unified interface between the nodes. Constant values are represented as particular cases of random variables which return the same constant value when they are sampled. Further, the statistical operators automatically recognize constant values and perform the operations accordingly (typically shift or scale the distribution with the constant value).

A typical PM node stores a random variable and a collection of input and output links. Parametric nodes are typically leaf nodes in a PM and contain therefore only output links towards other (operational) nodes. On this output links, they provide the stored RV representation for statistical computations.

3.4.2 Variability Propagation and Estimation of Results

The typical internal nodes in a performance macromodel are operational nodes, which implement a statistical operation on random variables. Such an operational node exhibits input links from the random variables on which the operation is to be applied and stores a random variable for the result of the operation. This resulting RV is then provided on the output links to the other nodes connected downstream.

The variability descriptions are propagated at pdf level from node to node. They start at the parametric leaf nodes, where the pdfs have been estimated from simulations, profiling operations, or technology descriptions. Following, at each operational node, a

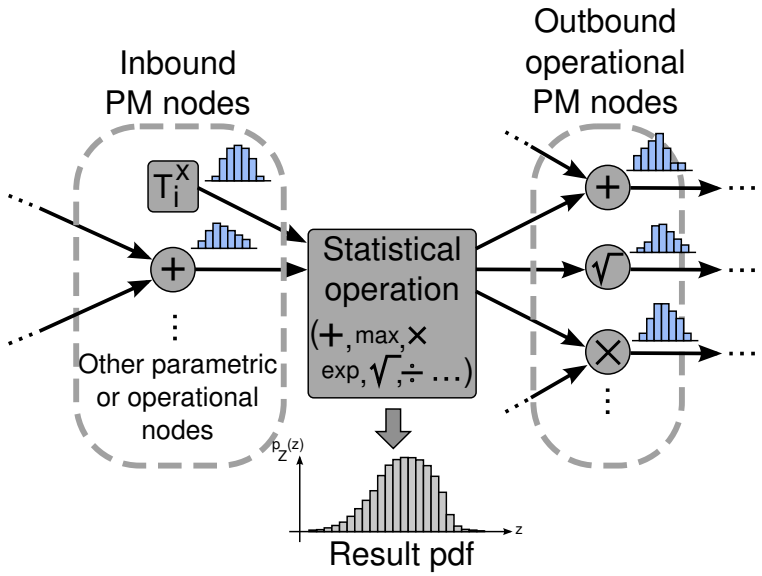

Fig. 3.23: Pdf propagation at each operational node in a PM.

statistical operator is applied to the random variables embedded in each of the inbound nodes. At this point, inbound nodes can represent either leaf nodes or other operational nodes. After the statistical operation is applied to the inbound RVs, the resulting RV is stored in the node and provided to the other nodes connected through output links. Fig. 3.23 illustrates the computation and storage of the result pdfs at each operational node.

In this way, the complete pdf representation is updated after each operation and further propagated from node to node towards the output of the performance macromodel. When a performance macromodel is evaluated, all the PM nodes must compute and store their local pdf. For this purpose, the evaluation request is first sent to the output PM node and from this node it is recursively propagated upstream in the graph until all PM nodes have evaluated their pdfs, as illustrated in Fig. 3.24. After all PM nodes in the graph have computed the pdfs, the distribution stored in the output node represents the result of the performance macromodel.

3.4.3 Changes and Updates Propagated Downstream

During the design space exploration, several re-mapping and re-scheduling decisions take place. For instance, if a task k is re-mapped from a resource of type RT_i to a different resource of type RT_j, its execution time changes from T^x_{k,RT_i} to T^x_{k,RT_j}. As a consequence, the parametric node containing the pdf of task's k execution time must be updated to the new random variable describing T^x_{k,RT_j}. Since this parametric node changed its embedded distribution, all the computed pdfs in the downstream nodes become invalid and must be reevaluated. As a consequence, an update must be triggered together with this change and propagated downstream to all the involved nodes. This update trigger occurs as

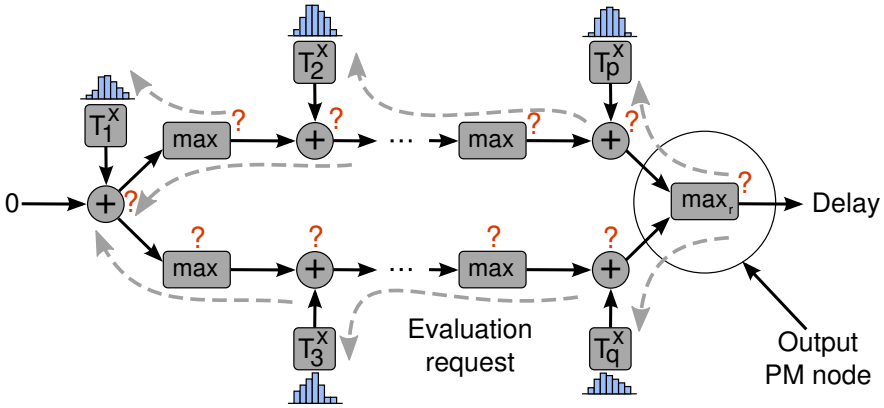

Fig. 3.24: Evaluation of a PM propagated upstream from the output node.

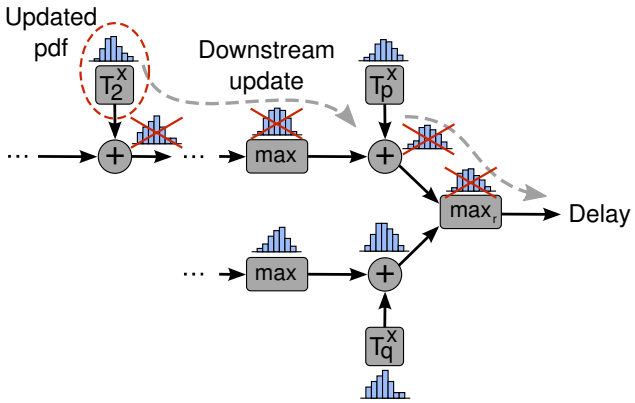

Fig. 3.25: Downstream propagation of a pdf update triggered by a change in system configuration.

shown in Fig. 3.25, and all the nodes affected by the propagated update must reset their evaluations and recompute the pdfs.

Similar updates are triggered by a scheduling change. Since scheduling dependencies add supplementary links between the PM nodes (see Fig. 2.5), a change in schedule involves deleting and adding the corresponding scheduling links. This changes the list of inbound nodes for the max nodes, which must consequently update their computed pdf. After recomputing the affected pdfs, a similar update must be propagated downstream in the PM.

Also communication changes, such as the choice of a different signaling circuit, or the tuning of supply voltage and body bias at the circuit level in the communication segments, as will be shown in chapter 4, has an impact on the delay and power consumption of the communication tasks. Such changes affect again the pdfs stored in the macromodel nodes and the corresponding updates must be propagated downstream.

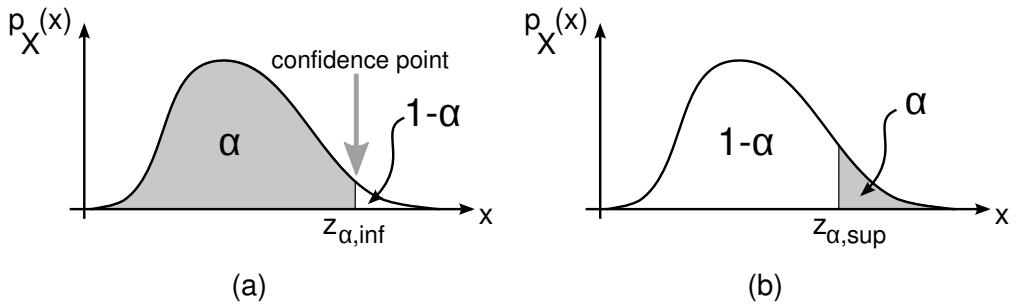

Fig. 3.26: Inferior quantile (a) and superior quantile (b) used as confidence points for design decisions.

3.4.4 Result Interpretation

Generally, due to the different sources of variation which occur along the design flow, most of the design parameters are obtained at the end of a synthesis step as random variables with a given distribution. Usually, design decisions operate with determined values, such as the evaluation of cost functions and the decision of selecting or rejecting a configuration with respect to its cost. Hence, it is necessary to extract a single value from a given distribution which accounts for particular design preferences, typically expressed as a desired yield level, a minimum performance limit, or a tradeoff between these two.

In this work, cost functions and the resulting design decisions are evaluated from the statistical distributions by means of quantile functions. The inferior quantile $z_{\alpha,inf}$ of a distribution is defined for a given value $\alpha \leq 1$ by the following relationship:

$$\int_{-\infty}^{z_{\alpha,inf}} p_X(x)\,dx = \alpha \tag{3.70}$$

which bounds a fraction equal to α from the pdf (see Fig. 3.26(a)). By denoting with $F_X(x)$ the cumulative distribution function and with $F_{c_X}(x) \triangleq P(X > x) = 1 - F_X(x)$ the complementary cumulative distribution function of a RV X, the inferior quantile is determined as:

$$z_{\alpha,inf} \triangleq z\Big|_{\substack{P(X\leq z)=\alpha \\ 0<\alpha<1}} = F_X^{-1}(\alpha) \tag{3.71}$$

A similar definition can be given for the superior quantile $z_{\alpha,sup}$, which bounds the superior part of the distribution, as shown in Fig. 3.26(b):

$$z_{\alpha,sup} \triangleq z\Big|_{\substack{P(X>z)=\alpha \\ 0<\alpha<1}} = F_{c_X}^{-1}(\alpha) \tag{3.72}$$

Either the inferior or the superior quantile can be used in design decisions during the synthesis.

In addition to comparing the costs of two solutions, design constraints must be checked when a possible solution is found. Since the constraints are constant numbers, the selected quantile from the result pdf is used to check if the solution meets the constraints.

For instance, if the quantile corresponding to $\alpha = 99\%$ is chosen, accepting a solution means that 99 % of the yield will meet the design constraints. The parametric yield obtained for the design is therefore reflected by the quantile selected during the design space exploration.

3.5 Performance Macromodels for Delay Estimation

This work develops statistical delay macromodels with variable granularity. There are mainly two macromodel hierarchy levels: system-level PMs, which compute the performance parameters for the complete design, and resource-level PMs, which estimate the characteristics of a single resource element (e.g. a signaling circuit). At system level, delays are estimated for processing tasks with a relatively coarser granularity and rely on execution time estimations specified as random variables T^x_{k,RT_i}. Communication tasks are, on the other hand, estimated using the detailed circuit-level models presented in chapter 4. Since the methodology is presented for both cases, the introduced macromodels can be easily extended to achieve the desired granularity for all tasks.

A delay macromodel is a directed acyclic graph consisting of linked PM nodes. As mentioned in Sec. 3.4.1, the PM nodes represent either parameters (leaf nodes) or statistical operations (internal nodes). PM nodes which apply a statistical operation are implemented using the statistical operators developed in Sec. 3.3. In addition, each PM node embeds a random variable representation to describe either the stored parameter or the result of the statistical operation.

3.5.1 Structure and Properties

The origin of the delay PM consists of the starting time value T^s_{global}. This is usually the deterministic zero value, but can also be any other timing value, even statistical, at which the timing computation starts. This latter case is particularly useful e.g. if the modeled system represents a link in a larger processing chain and if the end time of the previous link is also estimated statistically.

As illustrated in Fig. 3.27, the execution times of tasks assigned to resources are given by the parametric nodes which embed the random variables T^x_i for each processing node i. The earliest starting time T^s_i of a processing node is computed as the statistical maximum of the end times of the nodes on which it depends. After that, the execution time is added to this starting time by means of a statistical sum operator to compute the end time T^e_i of each task.

For each processing node (PN) and communication node (CN) in the extended task graph a maximum node, a sum node, and a parametric node for the execution time are added in the delay macromodel. These three PM nodes constitute a structural element of the macromodel and all structural elements are maintained into a list biunivocally associ-

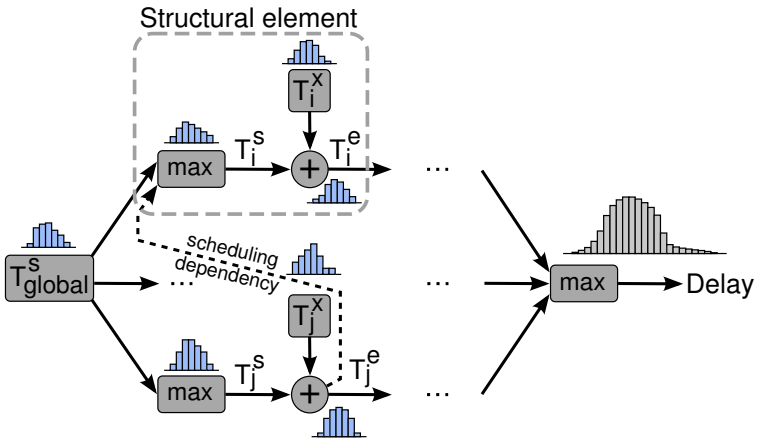

Fig. 3.27: Statistical performance macromodel for delay estimations.

ated to the list of processing nodes. Structural elements are interconnected by fixed data dependencies between the nodes and by temporary scheduling links. When a change in task scheduling occurs, the scheduling links between structural elements must be also updated appropriately.

There are a series of particularities introduced by the statistical nature of computations, such as the reflection of design space explorations in the statistical distributions computed within the macromodel. When a processing node is assigned to a different resource, its execution time changes. As explained in Sec. 3.4.3, after updating the random variable T_i^x, a value reset is propagated downstream, to update all the PM nodes which depend on this parameter. In addition, scheduling updates require the deletion of scheduling links between old successors in the scheduling list and the insertion of new links between the nodes of the new sequence. The removal of links between PM nodes, as well as the insertion of new links, require updates of the statistical computations which are also propagated downstream in the macromodel. For instance, if the scheduling order of two PNs/CNs assigned to the same resource is changed, the following changes occur in the delay PM:

- The scheduling links connecting the nodes are removed;
- The PM nodes (PMNs) where the links pointed and all their downstream PMNs must be updated;
- The new scheduling links connecting the nodes in their new positions are added;
- The PMNs where the new links point and all their downstream PMNs must be updated.

Further, if a PN/CN is assigned to a new resource, the following changes must be performed:

- The RV of the T^x input parameter node is updated;

PN	R_0		R_1		R_2		R_3		R_4	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1	21	4	27	4	16	3	22	3	17	3
2	36	4	41	5	32	4	38	4	32	4
3	51	5	56	5	45	4	51	5	46	4
4	39	4	46	5	36	4	43	4	37	4
5	14	3	20	4	11	3	17	3	12	3

Tab. 3.1: Parameters of the execution times of five PNs on different resources. Values given in nanoseconds.

- The affected sum node is reset and all its downstream PMNs;
- The modified PN/CN is removed from the scheduling list of the previous resource:
  - The scheduling links between the node and its predecessor and successor in the scheduling list are updated: first both removed, then a new link is added from the predecessor to the successor;
  - Whenever a scheduling link is removed or added, the PMN where the link pointed/points must be reset together with all its downstream PMNs.
- The PN/CN is inserted into the scheduling list of the new resource:
  - A scheduling link is removed and two new links are added to the PM;
  - The affected PMNs are reset downstream correspondingly.

3.5.2 Application Examples

Four execution scenarios with five processing nodes have been selected for illustrating the results of the delay PM. Assuming a set of five processing resources on which the PNs can be mapped, the execution times of each PN on each resource are described by statistical distributions with mean and standard deviation values as specified in Tab. 3.1.

In the first execution scenario, the PNs are independent and are mapped to individual resources, as illustrated by the task graph in Fig. 3.28(a). Since there are no data or resource dependencies, the PNs are executed in parallel, leading to the best execution time from the considered scenarios. The system latency is evaluated as a distribution using the delay PM and is shown in Fig. 3.29.

The second scenario represents the worst-case execution time, in which all PNs are mapped to the same resource R_2, as depicted by the task graph in Fig. 3.28(b). Here, the PNs are serialized by the scheduling dependencies, which leads to the largest execution time, as shown by the distribution plotted in Fig. 3.29.

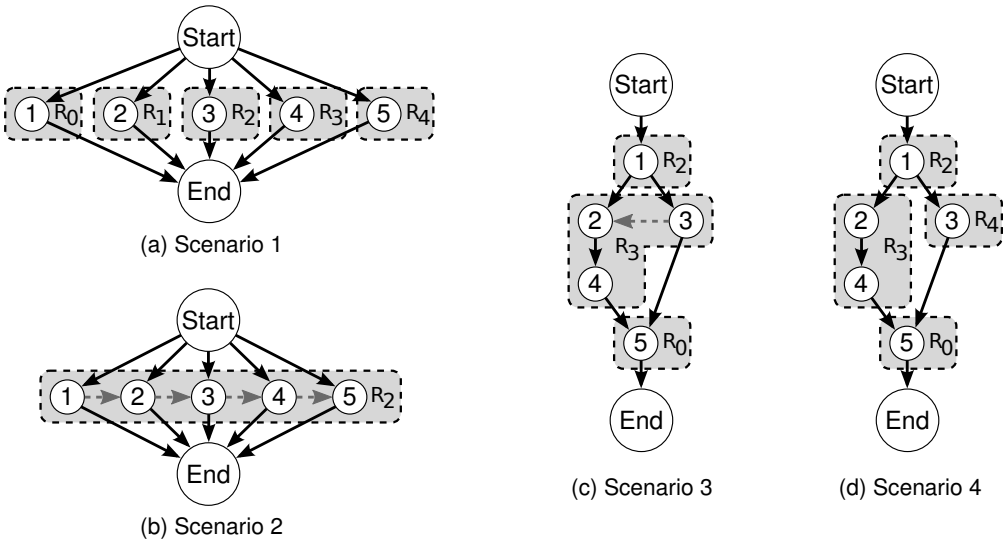

Fig. 3.28: Execution sequences and resource mappings for the four test scenarios.

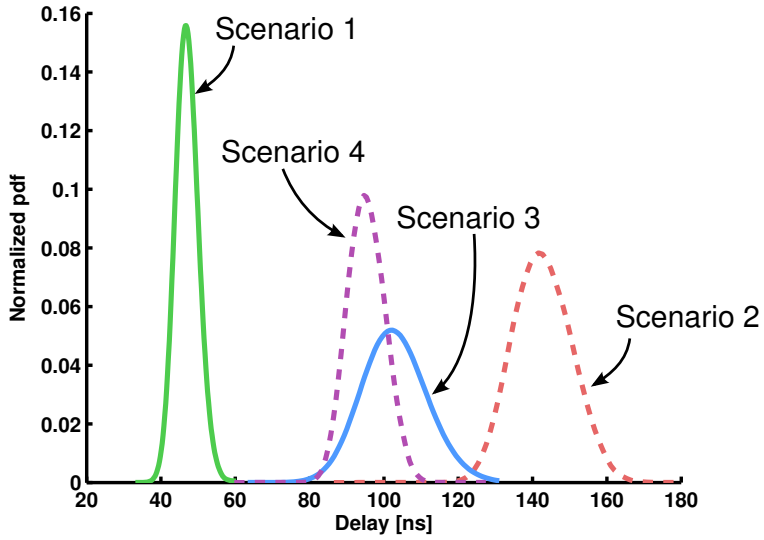

Fig. 3.29: Statistical delays evaluated using the delay PM.

The next two mixed scenarios illustrated in Fig 3.28(c) and (d) add data and scheduling dependencies and, depending on resource mapping and scheduling, achieve execution times between the best and worst-case scenarios. In the third scenario, PNs 2, 3, and 4 are assigned to the same resource R_3, therefore their execution is serialized by the additional scheduling dependency. In the fourth scenario, PN_3 is assigned to a different resource, therefore it can run in parallel with PN_2 and PN_4. This parallel execution leads to a shorter execution time with respect to the third scenario, which is also reflected by the results from Fig. 3.29.

3.6 Performance Macromodels for Energy Consumption

Power and energy consumption are commonly employed within the system design lexicon as interchangeable notions. Without affecting the generality of these terms, it is to be noted that across the design flow steps, the two notions may not always be compatible. As discussed in Sec. 2.2.3, the estimation of total energy consumption as performance metric during the synthesis has a series of advantages over the estimation of power dissipation. First, estimating the energy consumption takes into account the effect of leakage power dissipations over long periods of idle time. This allows further to optimize the system such that long slacks between the tasks are avoided to reduce the amount of leakage dissipated by the idle resources. Second, the overall reduction of total energy consumption is of primary importance in battery-powered embedded applications. And last, the decision of mapping a task on a resource with lower dynamic power consumption, but which requires a relatively long execution time, over assigning the task to a resource with a higher power level but which requires a very short execution time might be detrimental under the global system circumstances.

To include all these aspects in the framework in a unified way, energy consumption has been chosen as the performance metric evaluated by the macromodels. It is also important to remember, that the average power dissipation can be obtained from the estimated energy through division by the execution time evaluated using the delay PM. Several approaches in the literature, such as [50,56], use energy as the performance metric within the context of power-optimized hardware/software co-synthesis.

3.6.1 Dynamic Energy Macromodels

The application profile specification described in Sec. 3.1.2 defined the average dynamic power consumption $P_{d,i}^{RT_j}$ as a statistical distribution for each node PN_i on every compatible resource type RT_j. The average dynamic energy consumption of executing PN_i is approximated by the statistical product between the specified average power and the PN execution time:

$$E_{d,i} = P_{d,i}^{RT_j} \cdot T_{i,RT_j}^{x} \tag{3.73}$$

Hence, the total dynamic energy, evaluated by the PM, is given by:

$$E_{d,total} = \sum_{i=1}^{N_{PN}} E_{d,i} \tag{3.74}$$

The structure of the implemented PM is depicted in Fig. 3.30. Each PN introduces in the dynamic energy PM a structural element containing a product node which combines two parametric nodes: the average dynamic power dissipation $P_{d,i}^{RT_j}$ and the execution time T_i^x.

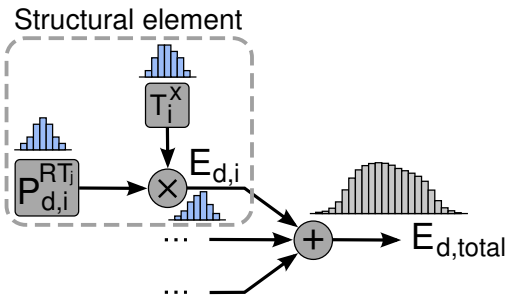

Fig. 3.30: Statistical performance macromodel for estimating the dynamic energy consumption.

Note, that the macromodel structure depicted in Fig. 3.30 includes only the contribution of the processing nodes. The detailed modeling of communication nodes and the corresponding structures in the performance macromodels are presented in chapter 4.

If a PN is assigned to a different resource, the dynamic energy PM is modified as follows. First, the random variables of the T_i^x and $P_{d,i}^{RT_j}$ nodes are updated to the values corresponding to the new resource. After that, the value of the affected multiplication node is reset, as well as the values of all its downstream PMNs.

3.6.2 Leakage Energy Macromodels

The leakage energy consumption of a resource R_k is estimated from the product between the average leakage power P_{l,RT_j} specified as a distribution in the application profile (see Sec. 3.1.2) and the total time in which the resource is idle:

$$E_{l,R_k} = \sum_{i=1}^{N_k-1} \left(T_{i+1}^s - T_i^e\right) P_{l,RT_j} + \left(T_{global}^e - T_{N_k}^e\right) P_{l,RT_j} \tag{3.75}$$

where N_k is the number of tasks mapped on R_k and T_{global}^e is the total system latency. The total idle time is computed as the sum of slacks between the execution of the scheduled tasks, therefore the start and end times of each processing node must be known.

Since the starting times T_i^s and the end times T_i^e are computed within the delay macromodel (see Sec. 3.5.1), they can be directly used by connecting links between the output of the nodes from the delay PM which compute them and the nodes of the leakage energy PM.

The leakage energy PM is built starting from a list with all processing resources. For each resource in the list, a structural element is created containing the following subelements:

- For each slack between the PNs scheduled on the given resource, a pair of links from the delay PM and one difference node are connected as shown in Fig. 3.31. This difference node evaluates the slack between two successively scheduled PNs PN_i and PN_{i+1}. The minuend link connects to the output of a PMN from the delay

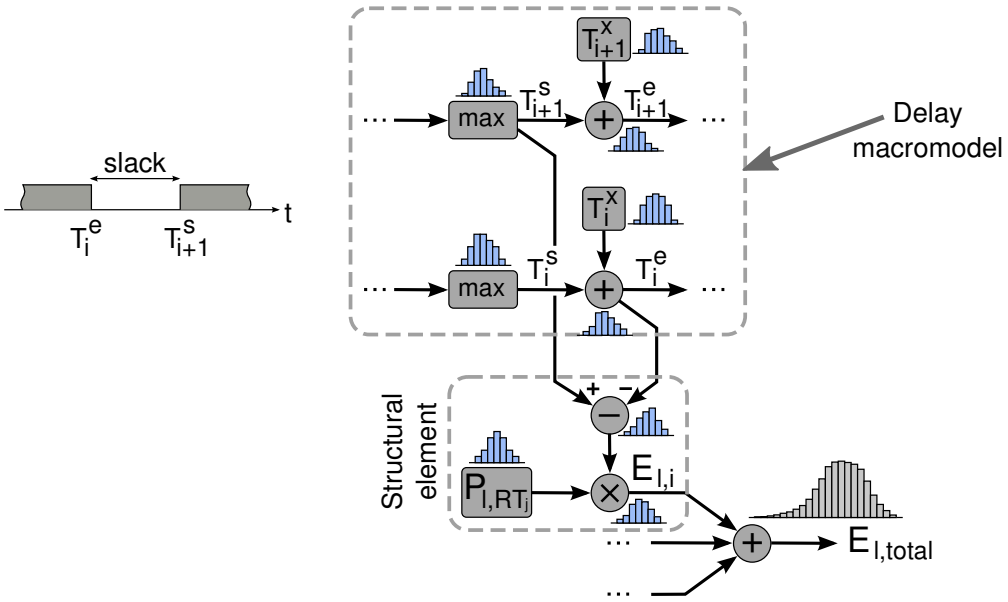

Fig. 3.31: Statistical performance macromodel for leakage energy estimation.

PM, which evaluates the start time T_{i+1}^s of PN_{i+1}, while the subtrahend link connects to the end time T_i^e of the previous node PN_i.

- A similar set of links and a difference node for computing the slack of the last PN in the scheduling list (PN_{N_k}), where the minuend is the system end time, as evaluated at the output of the delay PM.

- An additional difference node which evaluates the timing gap between the initial system start time (as specified by the root of the delay PM) and the start time of the first PN scheduled on this resource.

- A sum node to add all the slacks, a parametric node, which stores the average leakage power of the resource, and a product node which multiplies the power with the sum of the slacks.

Finally, the PM output is implemented by a sum node, which adds the energy consumptions of all active processing resources.

Whenever the scheduling order of two PNs changes, the following modifications must be performed on the leakage energy PM. First, the complete structural element corresponding to the resource on which the scheduling order changed must be updated, since both the PN succession and the slacks have changed. In addition, the PM output node must be reset whenever a structural element is updated. It is important to note, that although other slacks, from different resources, might change during this schedule switch (due to data dependencies), the affected structural elements are automatically updated to the new values through the reset propagated downstream from the delay PM. This update is facilitated by the direct connections between the two PMs.

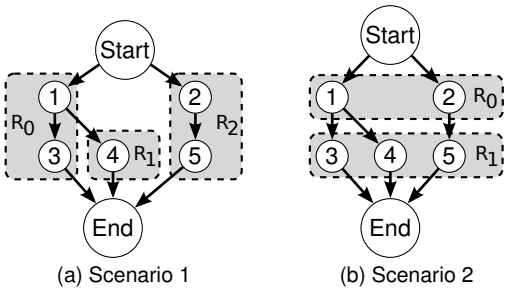

Fig. 3.32: Delay-optimized resource mapping (a) and mapping with improved energy consumption (b).

	R_0		R_1		R_2	
	μ	σ	μ	σ	μ	σ
P_{l,RT_j}	22	2.2	9	1.8	43	3.2
$P_{d,1}^{RT_j}$	22	2.9	17	2.4	27	3.2
$P_{d,2}^{RT_j}$	19	1.8	16	2.1	26	3.2
$P_{d,3}^{RT_j}$	26	3.7	21	3.2	32	3.9
$P_{d,4}^{RT_j}$	24	3.6	21	3.1	31	3.9
$P_{d,5}^{RT_j}$	19	2.7	16	2.6	24	3.1

Tab. 3.2: Power parameters for five PNs and three different resources. Values given in milliwatts.

Furthermore, if a PN is reassigned to a new resource, the structural elements corresponding to the previous and to the new resource must be updated. Again, all the slacks affected by this change are updated automatically through the connections with the delay PM.

3.6.3 Application Examples

Let the five PNs described by the execution times from Tab. 3.1 be mapped on the resources R_0, R_1, and R_2 according to the task graphs in Fig. 3.32. The average leakage power of each resource and the average dynamic power required for executing the PNs are specified using statistical distributions with the mean values and standard deviations given in Tab. 3.2.

The dynamic and leakage energy consumptions evaluated with the developed PMs are displayed in Fig. 3.33 for the two scenarios. It can be observed that the first mapping scenario is optimized for speed, as it uses three different resources for executing parallel tasks. The higher leakage consumption is caused by the idle time of R_1 waiting for the results from PN_1 and by the resource R_2 which has the highest leakage power. The second mapping scenario shown in Fig. 3.32 achieves an improved energy consumption by mapping the tasks only on R_0 and R_1. Most of the leakage in this case is due to the slack between the execution of PN_2 on R_0 and the total end time T_{global}^{e} and to the slack

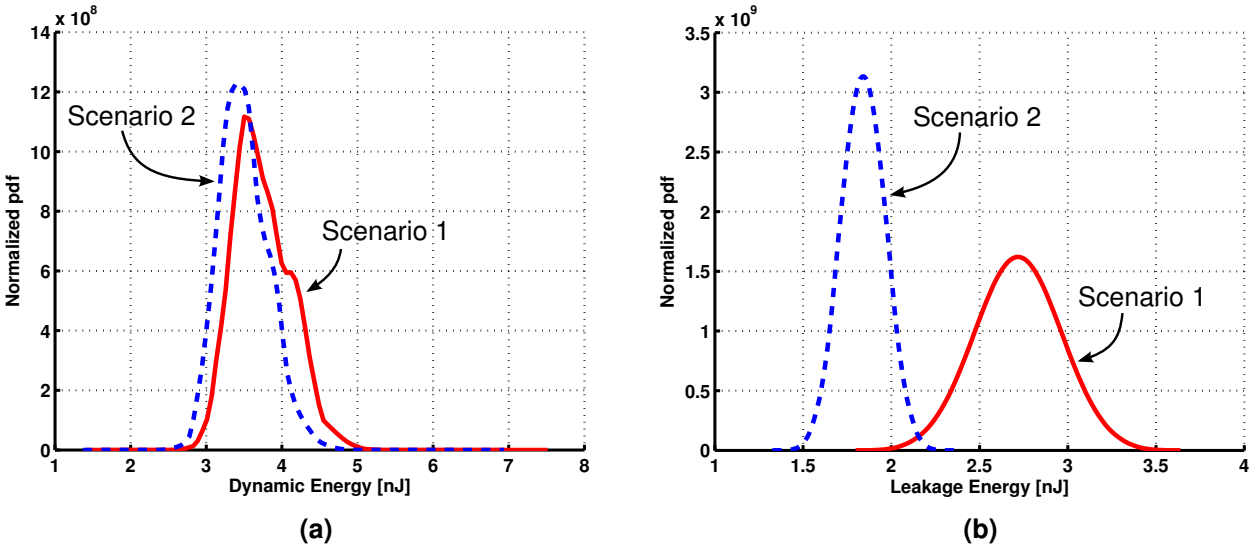

Fig. 3.33: Statistical dynamic energy (a) and leakage energy (b) consumptions evaluated using the energy PMs.

between the initial time T^s_{global} and T^s_3 on R_1. As shown in Fig. 3.33(a), the dynamic power is only moderately optimized, mainly by avoiding the power-intensive execution on R_2 (see Tab. 3.2). In addition, the unused resources can be turned off, therefore they practically do not participate to the total energy consumption. A more radical power reduction could be achieved by mapping all the tasks on R_1, however at the expense of a much higher latency, since the execution of all PNs would be serialized.

3.7 Partitioning, Assignment, and Scheduling Optimization

The estimations obtained from the developed PMs are used for guiding the partitioning of processing tasks, assignment on resources (mapping), and for scheduling optimizations. Given the estimated metrics for delay and energy consumption, a global optimization method is employed for the exploration of solution space. Within the optimization context, particularly interesting decisions are to change the task mapping configuration and to try several scheduling sequences on the resources. A further important aspect is the exploration of communication resources and optimization of the communication links.

The implemented exploration of the solution space is guided by the evaluation of a parametrized cost function which describes the design performance. By balancing several parameters, the cost function can be tuned to reflect multiple design preferences regarding the tradeoff between speed and energy consumption and the desired yield level.

```
INITIALSCHEDULE()
 1  for each R_i ∈ S_R
 2  do Start scheduling list L_{s,i};
 3     Get list of assigned PNs L_{PN,i};
 4     for each PN_j ∈ L_{PN,i}
 5     do if L_{s,i} = ∅
 6           then Append PN_j to L_{s,i};
 7           else for each PN_k ∈ L_{PN,i}
 8                do if !PN_k.ISUPSTREAM(PN_j)
 9                      then Insert PN_j in L_{s,i} before PN_k;
10                if PN_j ∉ L_{s,i}
11                  then Append PN_j to L_{s,i};
12     UPDATEPMS();
```

Listing 3.2: Algorithm for finding the initial scheduling configuration.

3.7.1 Methods for Solution Space Exploration

Several design variables are searched during the exploration. One of them is the configuration of assigned resources for each PN. After an initial resource mapping, PNs are reassigned in the optimization process to other compatible resources. Another design variable is the scheduling order of PNs on the assigned resources. Once an initial scheduling configuration is found, according to the dependencies in the task graph, the order of tasks in the scheduling lists is changed in the search for a better solution. Furthermore, additional design characteristics are explored, such as the choice of communication resources with different signaling methods, as well as circuit-level optimizations including voltage scaling and body biasing, as described in Sec. 4.4.5.

The mapping between processing nodes and compatible resources is defined by the function $Resmap : S_{PN} \rightarrow S_R$, such that $Resmap\,(PN_j) = R_i,\ \forall PN_j \in S_{PN},\ R_i \in S_{R,j}$, where S_{PN} is the set of processing nodes, S_R is the set of available resources, and $S_{R,j} \subset S_R$ is the subset of compatible resources for PN_j.

A static non-preemptive scheduling method is employed within this work, in which each processing resource maintains a list with the scheduled PNs. An initial valid scheduling configuration must be found as the starting point for explorations in the solution space. The algorithm which finds the initial scheduling is presented in Listing 3.2. and schedules each node from the list of assigned nodes before the first node which is not placed upstream in the task graph (the method PN_k.ISUPSTREAM(PN_j) checks if PN_k is found upstream of PN_j).

A nested loop optimization algorithm based on simulated annealing has been implemented in this work for the solution space exploration. Nevertheless, the developed statistical methodology and the macromodels can be employed in other combinatorial optimization approaches as well.

3.7.2 Cost Function Evaluation

The cost function $\mathcal{C} : \mathbb{P}^n \rightarrow \mathbb{P}$ is defined on the set of real pdfs defined by (3.60) and embeds the performance metrics which must be optimized in the synthesis. Within this work, the cost function is implemented as a weighted sum of the delay, dynamic energy, and leakage energy estimated using the performance macromodels:

$$\mathcal{C}\left(T^e_{global}, E_{d,total}, E_{l,total}\right) = w_T \cdot T^e_{global} + w_{E_d} \cdot E_{d,total} + w_{E_l} \cdot E_{l,total} \tag{3.76}$$

The weights w_T, w_{E_d}, and w_{E_l} are adjustable, therefore the combined cost function can lead to a design optimized for speed, for energy consumption, or for a weighted combination of them.

As discussed in Sec. 3.4.4, the cost pdf $\mathcal{C}()$ is interpreted with a quantile function. The extracted quantile employed for design decisions is also adjustable and offers a wide range for defining the desired parametric yield.

3.7.3 Optimization Loop

The simulated annealing-based optimization loop is performed over a combinatorial solution space having the following coordinates:

- The values of $Resmap\left(PN_j\right),\ \forall PN_j \in S_{PN}$;
- The scheduling order of PNs on each processing resource;
- The type of signaling resource, supply voltage value, and body bias, on each communication segment, as presented in chapter 4.

The first iteration performs an uniform random assignment of PNs on the compatible resources. Hereby, any possible assignment is valid, however the achieved performance is strongly influenced by the data and resource dependencies. The initial scheduling is performed as presented in Listing 3.2. Starting from this initial configuration, iterative jumps in the neighborhood of the current solution are performed by changing either one of the above-mentioned coordinates. The type of jump at each iteration is determined by different adjustable probabilities and typically considers the following aspects:

- Re-assigning a PN to a different resource requires a change in the scheduling lists of the affected resources;
- A switch between two PNs in the scheduling list of a resource is performed more often than a PN remapping, to allow for finding the optimal scheduling for a given resource mapping configuration;

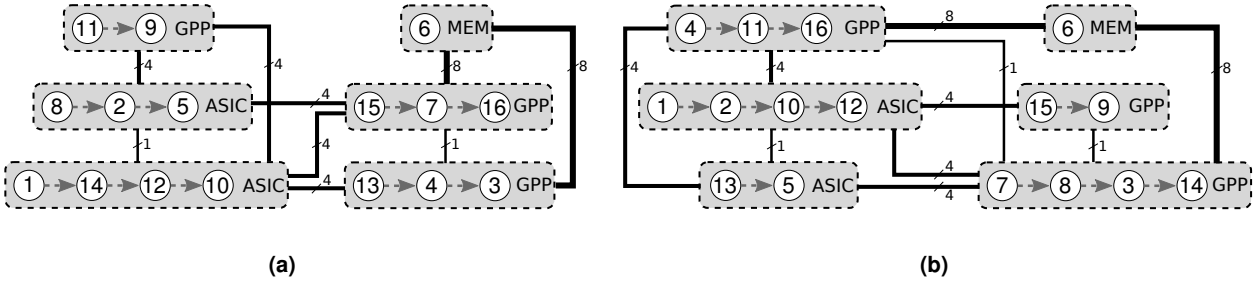

Fig. 3.34: Initial random assignment and scheduling (a) and optimized configuration (b).

- Since PN remapping also changes the communication segments needed between the resources, a change of the signaling resource on a segment occurs with a higher probability, to allow finding the optimal communication circuits;
- For a given resource mapping and signaling resource configuration, several supply voltage and body bias changes are performed.

While remapping a PN to another compatible resource results always into a valid configuration, there is only a restricted subset of resources which allow a switch in their scheduling list. More exactly, two PNs in the scheduling list of a resource can be switched only if neither one of them is placed upstream of the other in the task graph. As a consequence, before performing the scheduling switch, a list is built with the resources which allow such a scheduling switch and one of them is selected with a uniform random distribution.

3.7.4 Optimization Results

To illustrate the results of the assignment and scheduling optimization, a task graph with 16 processing nodes has been mapped on a resource set containing three general purpose processors, two ASICs, and one memory block. The average execution times of the processing nodes varied between 10 and 20 ns (mean), the inter-task communication loads were selected between 5 and 15 Mb (mean), and the inter-resource communication segments had lengths ranging between 1 and 25 mm. Dynamic power varied in mean value between 10 and 30 mW and the leakage power was selected between 10 and 40 mW.

The initial mapping and scheduling configuration is shown in Fig. 3.34(a), where the PNs are assigned randomly to the resources. The communication segments which connect the resources are first dictated by the data dependencies between the PNs and their width is corresponding to the distance between the resources: longer buses have a wider bit width, whereas local buses have only one data wire. It is assumed that the resource floorplan is fixed.

The optimization used equal weights for the delay, dynamic energy, and leakage energy. Further, the simulated annealing used an exponential cooling schedule with factor

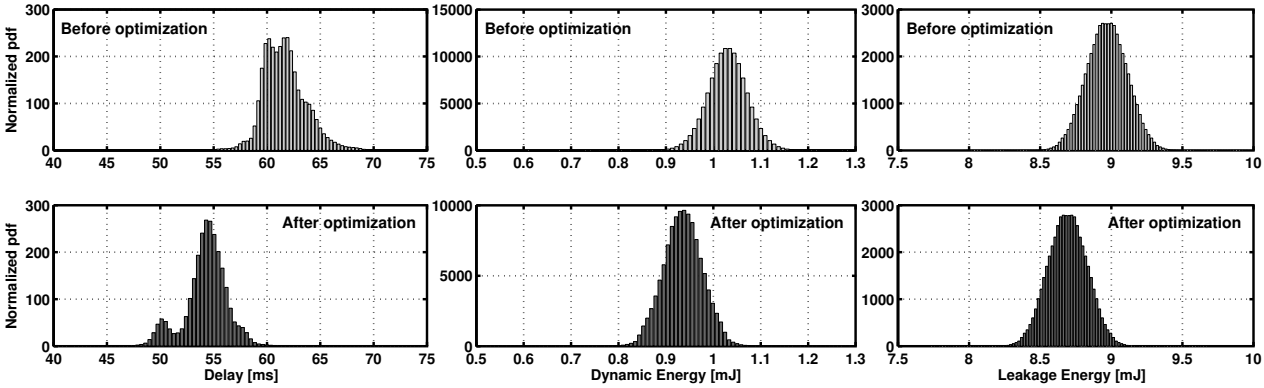

Fig. 3.35: Delay, dynamic energy, and leakage energy results before and after the optimization.

0.9 and 100 steps, each with 10 000 inner iterations. At each iteration, remapping a PN to a different resource occurred with a probability $P_{remap} = 0.33$, while switching the scheduling order of two PNs was performed with the probability $P_{reschedule} = 0.67$. Balanced weights have been employed in the cost function, such that improvements in delay and energy consumption have the same influence on the quality of a solution. Furthermore, a 99% level has been set for the parametric yield by means of quantiles.

Fig. 3.34(b) illustrates the best solution achieved during the optimization. Tasks with large execution times on the general purpose processors, as long as compatible, have been mapped on ASICs. Further, data-dependent tasks with large communication loads have been mapped preferably on the same resource, to avoid the time-expensive inter-resource communication. Also, intensive memory access tasks have been mapped on the resources which are communicating with the memory block via high-speed buses. Finally, the optimized scheduling minimized the slacks on the leakage-intensive resources.

The estimated pdfs for the total delay, dynamic energy, and leakage energy are presented in Fig. 3.35 for the initial and for the optimized configuration. The results show that the optimization achieved an improvement of 13.2% in delay, 9.8% in dynamic energy, and 4% in leakage energy at the considered 99% confidence point.

3.8 Summary

Starting from the need of accurately specifying statistical parameter distributions in the application and architectural profile and of employing them throughout the synthesis, this chapter developed a complete methodology for the statistical modeling of performance metrics.

The first contribution of this chapter is a generalized random variable model, capable of representing non-standard estimated distributions using discretized pdfs with adjustable accuracy. The typical usage, accuracy control, as well as a sampling method have been presented.

Another important contribution is the development of a propagation method for statistical distributions across the modeling expressions. Analytic expressions for the most often used operators, including the detailed derivation of a statistical product operator, have been presented. In this context, a further important contribution is the development of a fast generalized method for implementing statistical operators with a precision comparable to Monte Carlo at a very small fraction of the execution time.

Further, embedding the random variable model in the system representation has been explained. The implied particularities, such as the downstream propagation of updates and the result interpretation using quantile functions, have also been discussed.

Finally, the complete structures of statistical macromodels for delay and energy consumption have been presented and their application has been illustrated using a few examples. Moreover, the global optimization of resource mapping and scheduling using the statistical macromodels has been illustrated and analyzed in the context of an application example proving the efficiency of the developed methodology.

Chapter 4

Technology-Accurate, Variability-Aware Circuit-Level Models

Contents

Optimizations achieved at the level of task mapping and scheduling have been presented in Sec. 3.7. Nevertheless, more radical optimizations of the communication architecture can be achieved at the circuit level, as discussed in Sec. 2.6, through the choice of the signaling method and voltage optimizations in the transceiver circuit, such as supply voltage scaling and body biasing.

This chapter provides a selection of circuit-level models for two different signaling methods, developed by identifying the main current paths during circuit operation and by deriving analytic expressions from equivalent circuit representations. Technology-related information is included in the circuit models by using transistor-level current expressions derived from the BSIM 4.3 model [167]. In addition, the models support variability descriptions for every process parameter specified in the BSIM model card using the random variable models and the extended set of statistical operators developed in chapter 3. Apart from the selection of a signaling circuit, voltage scaling and body biasing techniques can be applied on the circuit models to show the immediate influence of these adjustments on the performance metrics.

The content of this chapter is organized as follows. First, Sec. 4.1 develops a statistical current-source transistor model derived from BSIM4 equations and describes the modeling approach for spatially-correlated parameter variations using grids and correlation decay models. On this basis, Sec. 4.2 derives analytic circuit-level models for delay and energy consumption for pulsed current-mode signaling circuits and analyzes their performance at various segment lengths, as well as under voltage scaling and body biasing. Another analytic model is developed for voltage-mode signaling circuits in Sec. 4.3 and the performance of the analyzed signaling methods is compared. In the following, Sec. 4.4 presents the embedding of the circuit-level models in the top-level performance macromodels and illustrates the use of both signaling methods to achieve an optimized communication architecture.

4.1 Variability-Aware Transistor Model

A technology-accurate circuit model should include all the relevant effects exhibited by state-of-the-art manufacturing processes. In addition, for a good accuracy, parameter variations must be considered within the model. Considering these observations, this work derives circuit-level models for the communication structures starting from an accurate statistical transistor model.

The underlying transistor-level model employed in this work is derived from the BSIM4 transistor model, currently accepted on a very wide scale as the *de facto* standard in transistor modeling for very deep sub-micron CMOS technologies. As a consequence, by using BSIM4 equations, this model includes all the model parameters currently employed

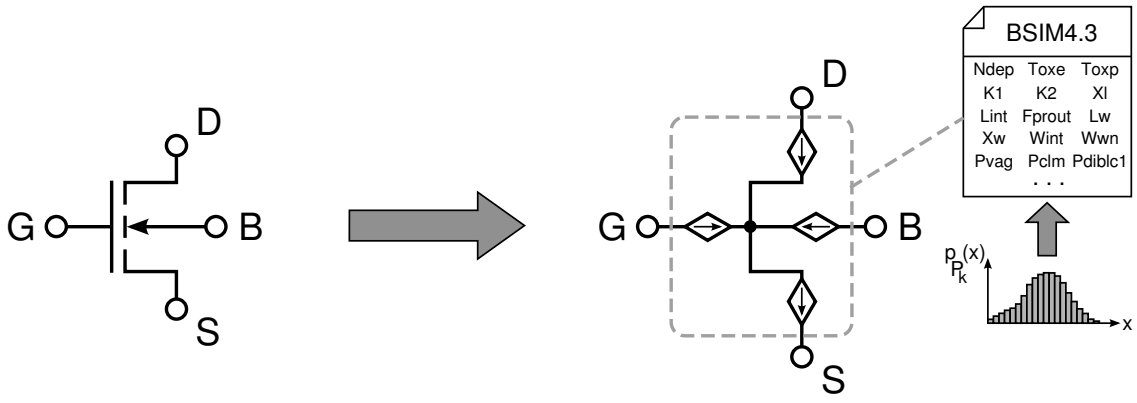

Fig. 4.1: Statistical current-source transistor model based on BSIM4.3 equations and parameters.

to describe process characteristics. In addition, the statistical analysis methodology developed in chapter 3 is applied to the model, thus allowing for the propagation of variability from each model parameter to the final current expression.

In this way, a completely statistical transistor-level model is developed, with extended modeling capabilities and good technology accuracy, which enables the use of parameter variations for each process parameter specified in the BSIM4 standard.

4.1.1 BSIM4.3-Based Current Source Model

A current source transistor model is employed in this work, as illustrated in Fig. 4.1. The current equations are derived from the BSIM4.3 transistor model, which is also used in the SPICE and Spectre [30] model cards for the 90 nm technology used throughout this work.

According to the BSIM4.3 specification [167], the effective gate voltage V_{gse} is computed considering the maximum electrical field in the polysilicon gate and the electric field in the gate oxide as:

$$V_{gse} = \boldsymbol{V_{fb}} + \Phi_s + \frac{q\varepsilon_{si}\boldsymbol{N_{gate}}\boldsymbol{T_{ox,e}^2}}{\boldsymbol{\varepsilon_{r,ox}^2}}\left(\sqrt{1+\frac{2\boldsymbol{\varepsilon_{r,ox}^2}\left(V_{gs}-\boldsymbol{V_{fb}}-\Phi_s\right)}{q\varepsilon_{si}\boldsymbol{N_{gate}}\boldsymbol{T_{ox,e}^2}}}-1\right) \tag{4.1}$$

where $\boldsymbol{V_{fb}}$ is the flat-band voltage, $\boldsymbol{N_{gate}}$ is the poly-gate doping concentration, $\boldsymbol{T_{ox,e}}$ is the electrical gate oxide thickness, and $\boldsymbol{\varepsilon_{r,ox}}$ is the gate dielectric constant. Φ_s is the surface potential and is computed as:

$$\Phi_s = 0.4 + \frac{k_B T}{q}\ln\left(\frac{\boldsymbol{N_{dep}}}{n_i}\right) + \boldsymbol{\Phi_n} \tag{4.2}$$

where $\boldsymbol{N_{dep}}$ is the channel doping concentration, $\boldsymbol{\Phi_n}$ is the non-uniform vertical doping effect on the surface potential, and n_i denotes the intrinsic carrier concentration of silicon which exhibits a non-linear dependence on the parameters measurement temperature.

Further, the total threshold voltage is modeled by the following relationship:

$$\begin{aligned} V_{th} = {} & \boldsymbol{V_{th0}} + \left(K_{1,ox}\sqrt{\Phi_s - V_{bseff}} - \boldsymbol{K_1}\sqrt{\Phi_s}\right)\sqrt{1+\frac{\boldsymbol{L_{peb}}}{L_{eff}}} - K_{2,ox}V_{bseff} \\ & + K_{1,ox}\left(\sqrt{1+\frac{\boldsymbol{L_{pe0}}}{L_{eff}}} - 1\right)\sqrt{\Phi_s} + \left(\boldsymbol{K_3} + \boldsymbol{K_{3b}}V_{bseff}\right)\frac{\boldsymbol{T_{ox,e}}\Phi_s}{W_{eff}+\boldsymbol{W_0}} \\ & - \frac{1}{2}\left[\frac{\boldsymbol{D_{vt0,w}}}{\cosh\left(\boldsymbol{D_{vt1,w}}\frac{L_{eff}W_{eff}}{l_{tw}}\right)-1} + \frac{\boldsymbol{D_{vt0}}}{\cosh\left(\boldsymbol{D_{vt1}}\frac{L_{eff}}{l_t}\right)-1}\right](V_{bi}-\Phi_s) \\ & - \frac{0.5}{\cosh\left(\boldsymbol{D_{sub}}\frac{L_{eff}}{l_{t0}}\right)-1}\left(\boldsymbol{\eta_0} + \boldsymbol{\eta_b}V_{bseff}\right)V_{ds} \end{aligned} \tag{4.3}$$

where $\boldsymbol{V_{th0}}$ is the long-channel threshold voltage at zero body bias, $\boldsymbol{K_1}$ is the body-effect coefficient, $\boldsymbol{L_{peb}}$ models the lateral non-uniform doping effect, $\boldsymbol{L_{pe0}}$ is the lateral non-uniform doping at $V_{bs}=0$, $\boldsymbol{K_3}$ is the narrow width coefficient, $\boldsymbol{K_{3b}}$ represents the body effect coefficient of $\boldsymbol{K_3}$, $\boldsymbol{W_0}$ is a narrow width parameter, $\boldsymbol{D_{vt0,w}}$ and $\boldsymbol{D_{vt1,w}}$ are the first and second coefficients of narrow-width effects, $\boldsymbol{D_{vt0}}$ and $\boldsymbol{D_{vt1}}$ are the first and second coefficients of short-channel effects, $\boldsymbol{D_{sub}}$ is the coefficient of the DIBL effect on the output resistance, $\boldsymbol{\eta_0}$ is the DIBL coefficient in the subthreshold region, and $\boldsymbol{\eta_b}$ is the body-bias coefficient for the subthreshold DIBL effect. The coefficients $K_{1,ox} = \boldsymbol{K_1 T_{ox,e}}/\boldsymbol{T_{ox,m}}$ and $K_{2,ox} = \boldsymbol{K_2 T_{ox,e}}/\boldsymbol{T_{ox,m}}$ model the dependence of $\boldsymbol{K_1}$ and $\boldsymbol{K_2}$ of the gate oxide thickness, where $\boldsymbol{K_2}$ is the charge-sharing parameter and $\boldsymbol{T_{ox,m}}$ is the oxide thickness at which the parameters are extracted. V_{bi} designates the built-in voltage of the source and drain junctions and is expressed as:

$$V_{bi} = \frac{k_B T}{q}\ln\left(\frac{\boldsymbol{N_{dep}}\boldsymbol{N_{sd}}}{n_i^2}\right) \tag{4.4}$$

where $\boldsymbol{N_{sd}}$ is the doping concentration of the source and drain diffusion regions.

Short-channel and DIBL effects are included by defining the characteristic channel length as:

$$l_t = \sqrt{\frac{\varepsilon_{si}\boldsymbol{T_{ox,e}}X_{dep}}{\boldsymbol{\varepsilon_{r,ox}}}}\left(1+\boldsymbol{D_{vt2}}V_{bs}\right) \tag{4.5}$$

while the characteristic length at zero body bias is given by $l_{t0} = \sqrt{\frac{\varepsilon_{si}\boldsymbol{T_{ox,e}}X_{dep0}}{\boldsymbol{\varepsilon_{r,ox}}}}$. Hereby, $\boldsymbol{D_{vt2}}$ represents the body-bias coefficient of short-channel effects. $X_{dep} = \sqrt{\frac{2\varepsilon_{si}(\Phi_s - V_{bs})}{q\boldsymbol{N_{dep}}}}$ is the depletion width and $X_{dep0} = \sqrt{\frac{2\varepsilon_{si}\Phi_s}{q\boldsymbol{N_{dep}}}}$ is the depletion width at zero body bias. In addition, the characteristic length considering the narrow width effect in short channels is given by:

$$l_{tw} = \sqrt{\frac{\varepsilon_{si}\boldsymbol{T_{ox,e}}X_{dep}}{\boldsymbol{\varepsilon_{r,ox}}}}\left(1+\boldsymbol{D_{vt2w}}V_{bs}\right) \tag{4.6}$$

where $\boldsymbol{D_{vt2w}}$ is the body-bias coefficient of narrow width effects at small channel lengths.

The effective body to source voltage V_{bseff} limits the body bias to an upper boundary and is evaluated as:

$$V_{bseff} = V_{bc} + \frac{1}{2}\left[(V_{bs} - V_{bc} - \delta) + \sqrt{(V_{bs} - V_{bc} - \delta)^2 - 4\delta V_{bc}}\right] \tag{4.7}$$

where δ is a constant equal to 10^{-3} V and $V_{bc} = 0.9\left(\Phi_s - \frac{\boldsymbol{K_1^2}}{4\boldsymbol{K_2^2}}\right)$ represents the maximum boundary for V_{bs}.

The effective channel length and effective channel width are computed as:

$$L_{eff} = L + \boldsymbol{X_l} - 2\left(\boldsymbol{L_{int}} + \frac{\boldsymbol{L_l}}{L^{\boldsymbol{L_{ln}}}} + \frac{\boldsymbol{L_w}}{W^{\boldsymbol{L_{wn}}}} + \frac{\boldsymbol{L_{wl}}}{L^{\boldsymbol{L_{ln}}} W^{\boldsymbol{L_{wn}}}}\right) \tag{4.8}$$

$$\begin{aligned} W_{eff} = \frac{W}{\boldsymbol{N_f}} + \boldsymbol{X_w} - 2\Bigg[\boldsymbol{W_{int}} + \frac{\boldsymbol{W_l}}{L^{\boldsymbol{W_{ln}}}} + \frac{\boldsymbol{W_w}}{W^{\boldsymbol{W_{wn}}}} + \frac{\boldsymbol{W_{wl}}}{L^{\boldsymbol{W_{ln}}} W^{\boldsymbol{W_{wn}}}} \\ + \boldsymbol{D_{wg}} V_{gsteff} + \boldsymbol{D_{wb}}\left(\sqrt{\Phi_s - V_{bseff}} - \sqrt{\Phi_s}\right)\Bigg] \end{aligned} \tag{4.9}$$

where $\boldsymbol{X_l}$ and $\boldsymbol{X_w}$ are the length, respectively width variations due to masking and etching, $\boldsymbol{L_{int}}$ and $\boldsymbol{W_{int}}$ are the lateral diffusion, respectively the width reduction for one side, $\boldsymbol{D_{wg}}$ and $\boldsymbol{D_{wb}}$ are the dependences of the effective channel width on the gate bias and body bias, respectively, and the remaining parameters describe the interdependence of the effective length and width on variations in channel dimensions.

An effective $V_{gse} - V_{th}$ voltage difference is modeled by the following equation:

$$V_{gsteff} = \frac{n v_t \ln\left\{1 + \exp\left[\frac{m(V_{gse} - V_{th})}{n v_t}\right]\right\}}{m + n C_{ox,e}\sqrt{\frac{2\Phi_s}{q\boldsymbol{N_{dep}}\varepsilon_{si}}}\exp\left[-\frac{(1-m)(V_{gse} - V_{th}) - V_{off}}{n v_t}\right]} \tag{4.10}$$

where $m = 0.5 + \frac{\arctan(\boldsymbol{M_{inv}})}{\pi}$, $\boldsymbol{M_{inv}}$ is a fitting parameter for moderate inversion, v_t is the thermal voltage, $C_{ox,e} = \boldsymbol{\varepsilon_{r,ox}}\varepsilon_0/\boldsymbol{T_{ox,e}}$, V_{off} is the threshold voltage offset, and n is the subthreshold swing computed as:

$$n = 1 + \boldsymbol{N_{fact}}\frac{C_{dep}}{C_{ox,e}} + \frac{C_{dsct} + \boldsymbol{C_{it}}}{C_{ox,e}} \tag{4.11}$$

where $\boldsymbol{N_{fact}}$ is the subthreshold swing coefficient, $C_{dep} = \varepsilon_{si}/X_{dep}$, $\boldsymbol{C_{it}}$ is an interface trap parameter, and C_{dsct} is computed from $\boldsymbol{C_{dsc}}$ (the source/drain and channel coupling capacitance), $\boldsymbol{C_{dscd}}$ (the drain-bias sensitivity of $\boldsymbol{C_{dsc}}$) and $\boldsymbol{C_{dscb}}$ (the body-bias sensitivity of $\boldsymbol{C_{dsc}}$).

Finally, the channel current is computed as:

$$\begin{aligned} I_{ds} = &\frac{I_{ds0}\boldsymbol{N_f}}{1 + \frac{R_{ds}I_{ds0}}{V_{dseff}}}\left[1 + \frac{1}{C_{clm}}\ln\left(\frac{V_{A,sat} + V_{A,CLM}}{V_{A,sat}}\right)\right]\left(1 + \frac{V_{ds} - V_{dseff}}{V_{A,DIBL}}\right) \\ &\left(1 + \frac{V_{ds} - V_{dseff}}{V_{A,DIBL}}\right)\left(1 + \frac{V_{ds} - V_{dseff}}{V_{A,DITS}}\right)\left(1 + \frac{V_{ds} - V_{dseff}}{V_{A,SCBE}}\right) \end{aligned} \tag{4.12}$$

where $\boldsymbol{N_f}$ is the number of transistor fingers, I_{ds0} expresses the linear drain current, dependent on the effective mobility model, R_{ds} embeds the bias-dependent drain-source resistance model, C_{clm} is the capacitance associated to the channel length modulation, and the effective drain to source voltage V_{dseff} is formulated as a smooth transition between V_{ds} and the saturation voltage V_{dsat}. The remaining voltage expressions in the equation (4.12) represent components of the Early voltage, which is defined for the analysis of the output resistance of the device in saturation. These include the saturation component $V_{A,sat}$, the channel length modulation component $V_{A,CLM}$, the DIBL-induced component $V_{A,DIBL}$, the component due to the substrate current-induced body effect $V_{A,SCBE}$, and the component describing the drain-induced threshold shift $V_{A,DITS}$.

Similar to the drain current, expressions for the gate-to-substrate tunneling current I_{gb}, gate-to-channel current I_{gc}, as well as gate-to-source I_{gs} and gate-to-drain I_{gd} currents are defined in the BSIM4.3 specification. The total set of bias-dependent device currents define the source current model illustrated in Fig. 4.1.

It is to be noted, that all symbols written in bold in this section represent modeling or process parameters defined in the BSIM4.3 transistor model. Variations in each of the mentioned parameters, as well as in the remaining process parameters on which the device current expressions depend[1], can be expressed as statistical distributions, as explained in the next section. Hence, the current expressions are statistically evaluated and the result is stored in pdf form.

4.1.2 Modeling Spatially-Correlated Process Parameter Variations

As explained in the previous section, parameter variations can be described for any model parameter from the underlying BSIM4 specification. Without loss of generality, the simulations employed in this work include descriptions of variations in the following process parameters:

- $\boldsymbol{X_l}$ – the length variation due to masking and etching, described by:

$$\boldsymbol{X_l} = \boldsymbol{X_{l,nom}} + \sigma_{\boldsymbol{X_l}} N(0,1) \tag{4.13}$$

- $\boldsymbol{V_{th0}}$ – the long-channel threshold voltage at zero body bias, with a variation component dependent on the square root of the channel area [71, 34]:

$$\boldsymbol{V_{th0}} = \boldsymbol{V_{th0,nom}} + \sigma_{1,\boldsymbol{V_{th0}}} N(0,1) + \frac{\sigma_{2,\boldsymbol{V_{th0}}}}{\sqrt{W \cdot L}} N(0,1) \tag{4.14}$$

- $\boldsymbol{T_{ox,e}}$ – the electrical gate equivalent oxide thickness, modeled as:

$$\boldsymbol{T_{ox,e}} = \boldsymbol{T_{ox,e}^{nom}} + \sigma_{\boldsymbol{T_{ox,e}}} N(0,1) \tag{4.15}$$

[1] For illustration purposes, this section presents only a subset of the equations implemented in the model, as well as a reduced subset of the underlying model parameters.

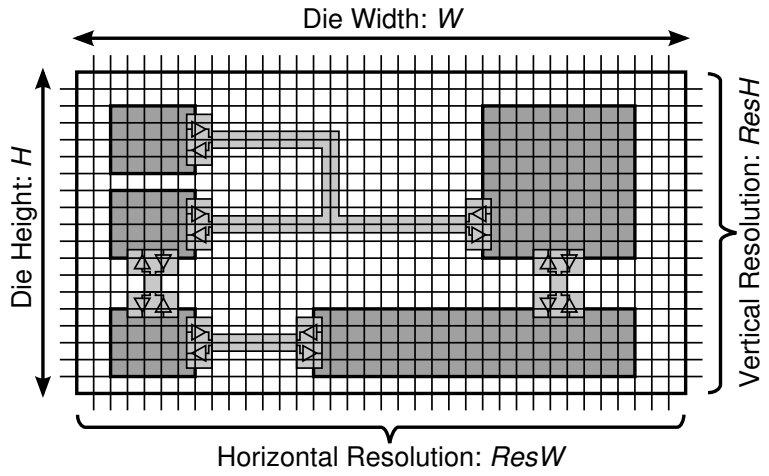

Fig. 4.2: Die grid for modeling spatially-correlated process variations.

- $\boldsymbol{T_{ox,p}}$ – the physical gate equivalent oxide thickness, described by a similar dependence:

$$\boldsymbol{T_{ox,p}} = \boldsymbol{T_{ox,p}^{nom}} + \sigma_{\boldsymbol{T_{ox,p}}} N(0,1) \tag{4.16}$$

It is to be noted, that experimental parameter distributions expressed in discretized pdf form are also supported. Further, the standard deviation values have been computed from the literature roadmap presented in Tab. 2.2. In addition to process parameters, intra-die temperature variations have been also modeled, with a strong spatial correlation.

For modeling spatially-correlated parameter variations, the die area $W \cdot H$ is divided into a grid with custom resolution $ResW \cdot ResH$, as illustrated in Fig. 4.2. Each process parameter has a spatially-correlated variation component which is distributed into $ResW \cdot ResH$ correlated random variables across the chip area. Thus, for each given process parameter P_k, a covariance matrix Σ_{P_k} is computed from the spatial correlation model and has a size of $(ResW \cdot ResH)^2$ elements. For each element in the covariance matrix $\Sigma_{P_k}(r,c)\,,\ \forall r,c = \overline{1,(ResW \cdot ResH)^2}$, the indices in the die grid are computed using the following relationships:

$$row_{index1} = r \operatorname{div} ResW \tag{4.17}$$

$$col_{index1} = r \bmod ResW \tag{4.18}$$

respectively:

$$row_{index2} = c \operatorname{div} ResW \tag{4.19}$$

$$col_{index2} = c \bmod ResW \tag{4.20}$$

where div and $\bmod$ indicate the integer division and modulo operations. Further, the center coordinates of the two grid cells corresponding to the two random variables are

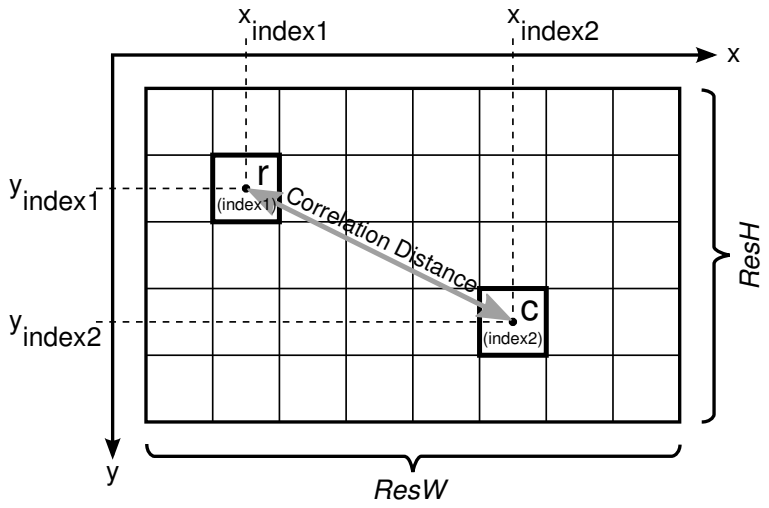

Fig. 4.3: Computed grid coordinates and correlation distance for the covariance matrix.

obtained as:

$$\begin{cases} x_{index1} = (col_{index1} + 0.5) \cdot W/ResW \\ y_{index1} = (row_{index1} + 0.5) \cdot H/ResH \end{cases} \tag{4.21}$$

$$\begin{cases} x_{index2} = (col_{index2} + 0.5) \cdot W/ResW \\ y_{index2} = (row_{index2} + 0.5) \cdot H/ResH \end{cases} \tag{4.22}$$

The correlation distance between the two variables is consequently computed from the two coordinate pairs, as shown in Fig. 4.3.

Finally, the covariance value is computed using the corresponding correlation model as:

$$\Sigma_{P_k}(r, c) = \rho_{index1,index2} \cdot \sigma_{index1}\sigma_{index2} \tag{4.23}$$

where σ_{index1} and σ_{index2} are the estimated standard deviations for the random variables at the two positions in the grid corresponding to r and c, respectively. In the case of process parameters with a spatial correlation described by a piecewise-linear (PWL) model, $\rho_{index1,index2}$ is computed as:

$$\rho_{index1,index2} = \begin{cases} 1 - \dfrac{d_\rho}{d_d}(1 - \rho_r), & d_\rho \leq d_d \\ \rho_r, & d_\rho > d_d \end{cases} \tag{4.24}$$

where d_ρ is the correlation distance computed as the euclidean distance between the centers of the two grid cells, d_d is the correlation decay distance, and ρ_r corresponds to a residual correlation which is still present for distances beyond the decay limit and is mainly the effect of die-to-die parameter variations. Alternatively, other spatial correlation models can be employed, such as a Gaussian correlation decay:

$$\rho_{index1,index2} = e^{-\left(\frac{d_\rho}{d_d}\right)^2} \tag{4.25}$$

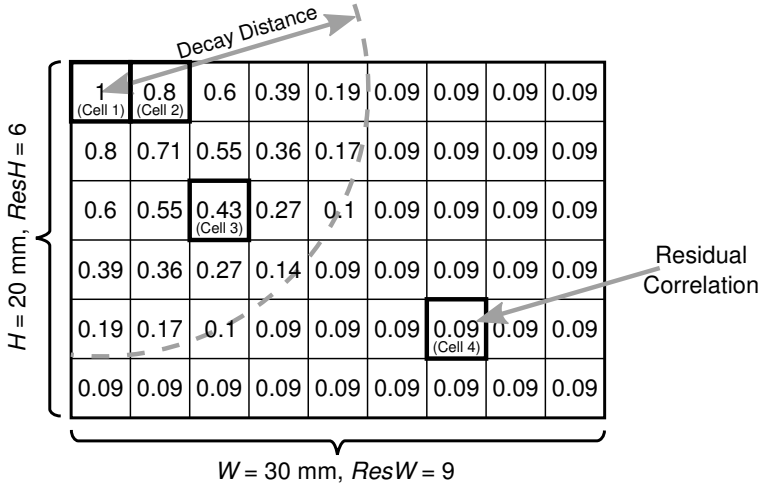

Fig. 4.4: Repartition of the correlation coefficient on a 9×6 grid, as reported to the top-left cell, for a decay distance $d_d = 15\,\text{mm}$ and a residual correlation $\rho_r = 0.09$.

This set of correlated random variables is decomposed into a set of independent RVs using principal component analysis (PCA). Assuming an initial set of spatially-correlated RVs:

$$\mathbf{P_k} = [P_{k,1}, P_{k,2}, \ldots, P_{k,n}]^T \tag{4.26}$$

where $n = ResW \cdot ResH$ is the number of grid cells, the covariance matrix Σ_{P_k} and the mean vector $\overline{\mathbf{P_k}} = [E\{P_1\}, \ldots, E\{P_n\}]^T$ are estimated. Then, a new vector of zero mean random variables is computed as the difference:

$$\mathbf{P_k^0} = \mathbf{P_k} - \overline{\mathbf{P_k}} \tag{4.27}$$

After that, the principal components of the zero mean vector are expressed as linear combinations:

$$C_{k,i} = v_{i,1}P_{k,1}^0 + v_{i,2}P_{k,2}^0 + \cdots + v_{i,n}P_{k,n} \tag{4.28}$$

where $\mathbf{V_i} = [v_{i,1}, \ldots, v_{i,n}]^T$ is the i-th eigenvector of Σ_{P_k} in decreasing order of the magnitude of the corresponding eigenvalues λ_i. Alternatively, the principal components can be expressed in matrix form as:

$$\mathbf{C_k} = [\mathbf{V_1}, \ldots, \mathbf{V_n}]^T \cdot \mathbf{P_k^0} \tag{4.29}$$

Further, the initial set of spatially-correlated random variables can be expressed in terms of a new set $\mathbf{N}$ of n uncorrelated standard normal variables, with zero mean and unit variance:

$$\mathbf{P_k} = \overline{\mathbf{P_k}} + \mathbf{D^{\frac{1}{2}}} \cdot \mathbf{V^{-1}} \cdot \mathbf{N} \tag{4.30}$$

where $\mathbf{V} = [\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_n}]^T$ is the matrix containing the eigenvectors of Σ_{P_k}, $\mathbf{D} = \text{Diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of the eigenvalues of Σ_{P_k} in decreasing order, and $\mathbf{D^{\frac{1}{2}}} = \text{Diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}\right)$.

As an example, consider a die area of 30 mm$\times$20 mm divided by a 9×6 spatial correlation grid, as depicted in Fig. 4.4. The correlation coefficient between the top-left cell (cell 1

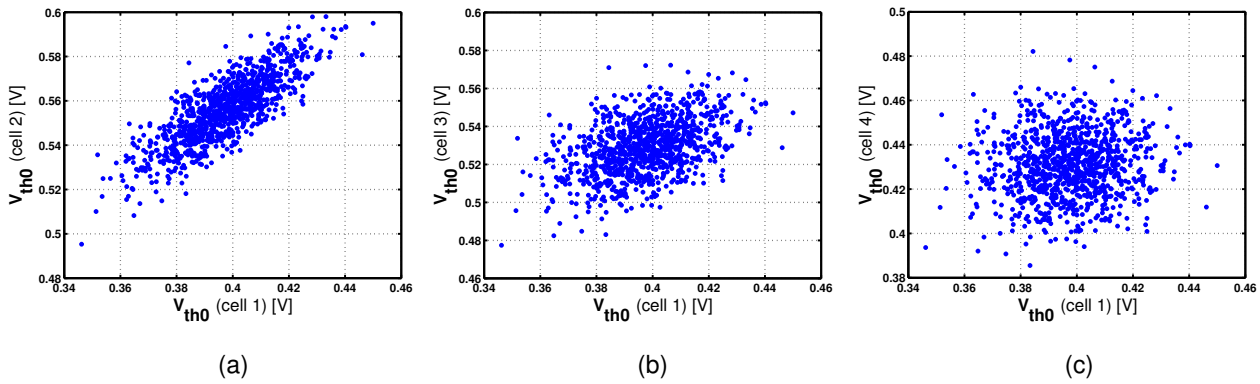

Fig. 4.5: Spatially-correlated values of the threshold voltage parameter V_{th0} from grid cells 2 (a), 3 (b), and 4 (c), plotted with respect to the values from cell 1.

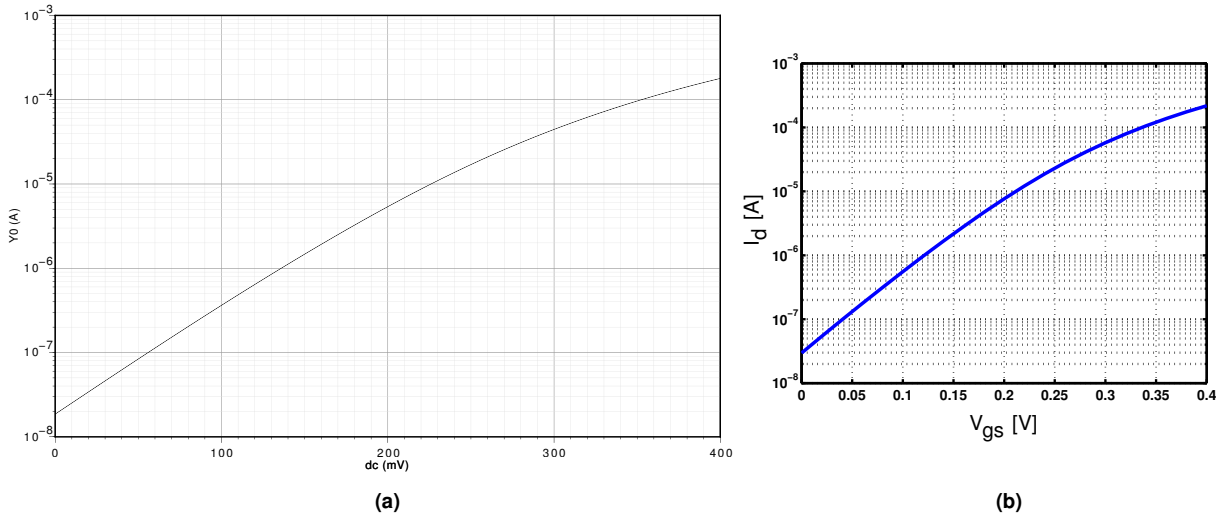

Fig. 4.6: Subthreshold current plot for an NMOS transistor with $W = 3\,\mu\text{m}$, $L = 80\,\text{nm}$ obtained with the commercial BSIM4 implementation in the Cadence Spectre circuit simulator (a) and with the derived current-source model (b).

in Fig. 4.4) and the other grid cells decays gradually according to the PWL model (4.24). The decay distance has been set to 15 mm and a residual correlation $\rho_r = 0.09$ has been assumed.

The spatially-correlated threshold voltage distribution for a 90 nm technology is shown in Fig. 4.5 for the grid cells 2, 3, and 4 from Fig. 4.4, plotted with respect to the values from cell 1. It can be seen in Fig. 4.5(a), that the threshold voltage values from the neighbor cell are highly correlated, while the values in cell 3 (Fig. 4.5(b)) still exhibit a significant amount of correlation, according to the coefficient value of 0.43. Since cell 4 (Fig. 4.5(c)) lies beyond the correlation decay distance, only a very weak residual correlation is present in the values.

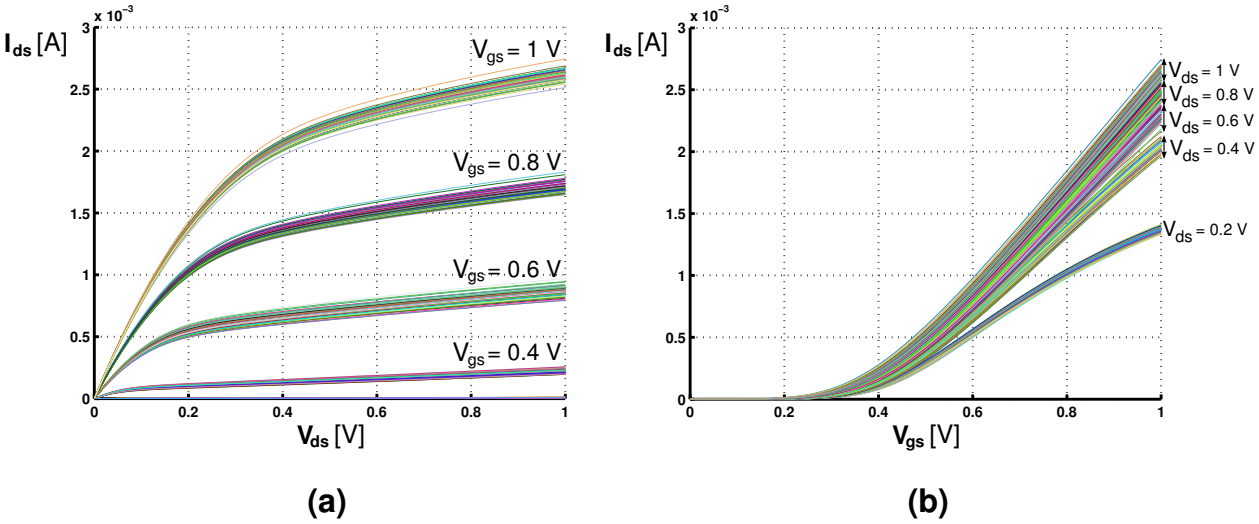

Fig. 4.7: Variations in the output and transfer characteristics obtained for an NMOS transistor with $W = 3\,\mu\text{m}$, $L = 80\,\text{nm}$ obtained from the process parameter variations described in Sec. 4.1.2.

4.1.3 Inclusion of Random Variables and Results Estimation

Due to the underlying BSIM4 equations, the derived transistor model captures well the very deep sub-micron effects. Fig. 4.6 shows a comparison between the subthreshold current simulated with the commercial implementation of the transistor model in the Cadence Spectre circuit simulator [31] and the current source model derived in Sec. 4.1.1. Further comparisons over several device sizes and bias ranges show also a good agreement with the commercial implementation.

Variations can be specified for all BSIM4 model parameters in the form of random variable descriptions, using the models developed in Sec. 3.2. In addition, spatially-correlated parameter variations can be specified using the two-dimensional die area grid and the spatial correlation models described in Sec. 4.1.2. As a consequence, the variability descriptions at process parameter level are propagated to the drain current expression using the statistical approach developed in Sec. 3.3.

An example of the variations obtained in the drain current starting from the spatially-correlated parameter variations described in Sec. 4.1.2 is shown in Fig. 4.7. It can be observed, that the drain current variability increases with the drain-to-source bias level, as illustrated in Fig. 4.7(a). At very small V_{ds} values, the drain current is almost constant with the process variations. On the contrary, Fig. 4.7(b) shows that the variability amplitude is not significantly affected by changes in V_{gs}.

This aspect is further investigated by evaluating the standard deviation of the drain current over the V_{ds} and V_{gs} voltage range. Fig. 4.8(a) shows the distribution of the drain current in the presence of process and temperature variations, whereas Fig. 4.8(b) plots the estimated standard deviation with respect to changes in V_{ds} and V_{gs}.

As previously observed, the results from Fig. 4.8(b) show a strong variation of the

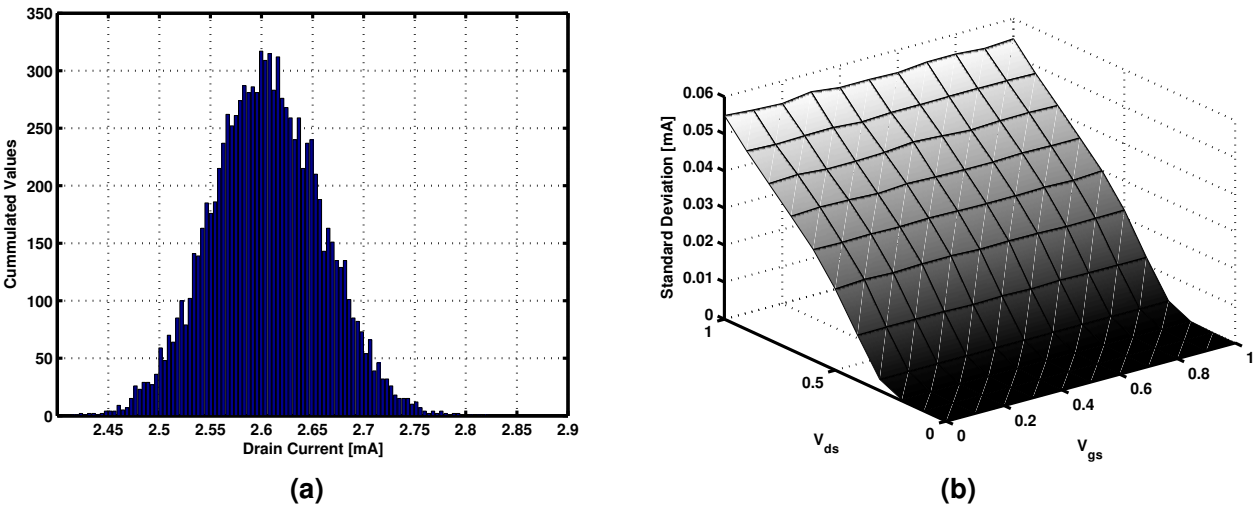

Fig. 4.8: Drain current distribution (a) and variation of the standard deviation over the bias ranges (b).

drain current variability with the drain-to-source voltage, while changes in V_{gs} do not significantly affect the variation. This result is explained by the strong dependences on V_{ds} of the factors in the drain current expression (4.12) and by the inner multiplications between V_{ds} and the variability-affected process parameters. First, the threshold voltage V_{th} has a strong dependence on $\boldsymbol{X_l}$ through L_{eff}. Similarly, the effective voltage difference V_{gsteff} and the effective mobility μ_{eff} depend strongly on the process variations through their dependence on V_{th}. In addition, the current factor I_{ds0} and the difference $V_{ds} - V_{dseff}$ have a linear dependence on V_{ds} and thus multiply the effect of process variations. Finally, it must be mentioned that the Early voltage components $V_{A,DITS}$ and $V_{A,SCBE}$ exhibit a linear dependence on the effective gate length L_{eff} and exponential dependence on V_{ds}.

4.2 Pulsed Current-Mode Signaling Model

Unlike traditional voltage-mode signaling methods, current-mode signaling techniques rely on current switching on the interconnect line to represent the "0" and "1" logic levels. As discussed in Sec. 2.6.1, the major drawback of current-mode signaling is its higher static power dissipation. To counter the effect of switching large static currents, a pulsed current-mode (PCM) signaling technique was proposed in [87], which uses sharp current pulses to modulate the signals before transmission, taking advantage at the same time of the high-frequency LC propagation through repeaterless interconnect links.

The design of the driver and receiver circuits is illustrated in Fig. 4.9. It can be observed that the PCM technique requires a larger area than a conventional voltage-mode buffer, due to a higher transistor count and to the integrated capacitances. Nevertheless, the advantage of achieving near speed-of-light propagation times across long interconnects without the use of repeaters balances this drawback. In the next section we analyze

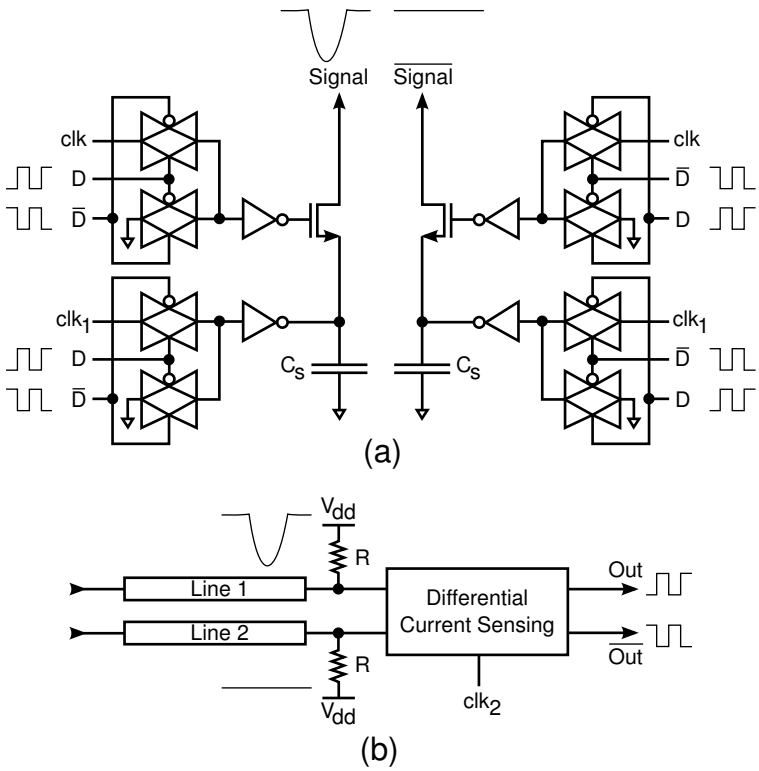

Fig. 4.9: Pulsed current-mode signaling driver (a) and receiver circuit (b).

the circuit and we identify the main switching current paths. Following, a circuit-level model for the delay and power is developed, based on the current-source transistor model from Sec. 4.1.

4.2.1 Derivation of Current Switching Paths

The input control stages of the PCM driver are implemented using dynamic logic, as shown in Fig. 4.10, to reduce the delay and area of the circuit. Each dynamic logic stage controls the duration of the current pulses and data transmission cycles by selectively connecting the data input to the inverter stages. Further, the inverter circuits control the M_1 and M_2 driving transistors, they charge and discharge the capacitances C_s, and close the current paths to the ground.

When the clock signals clk and $clk1$ are at zero, the output of the inverters is pulled down to ground, switching off the transistors M_1 and M_2, and discharging the capacitors C_s at the same time. The second clock signal $clk1$ is slightly delayed with respect to clk, such that when clk switches to "1", M_4 connects the inverter input to the drain of M_5 which is driven by the data input D, as illustrated in Fig. 4.11.

In the short time left until $clk1$ switches to "1", a sharp current pulse is driven from the V_{dd} supply, through the weak resistance R and the interconnect line, and switched to ground by the transistors M_1 and NM_1, as shown by the current path i in Fig. 4.12.

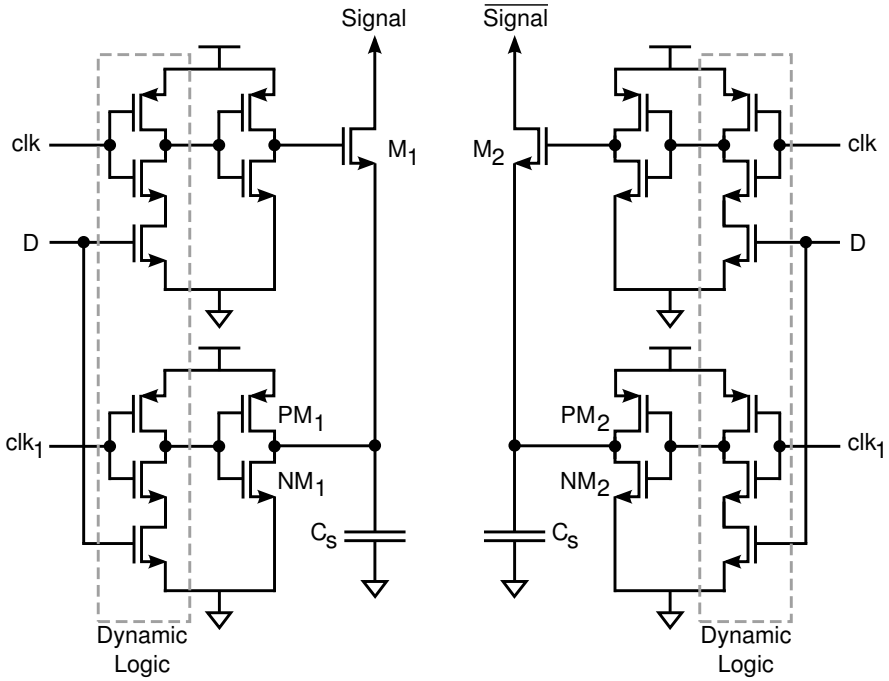

Fig. 4.10: Transistor-level circuit implementation of the PCM driver.

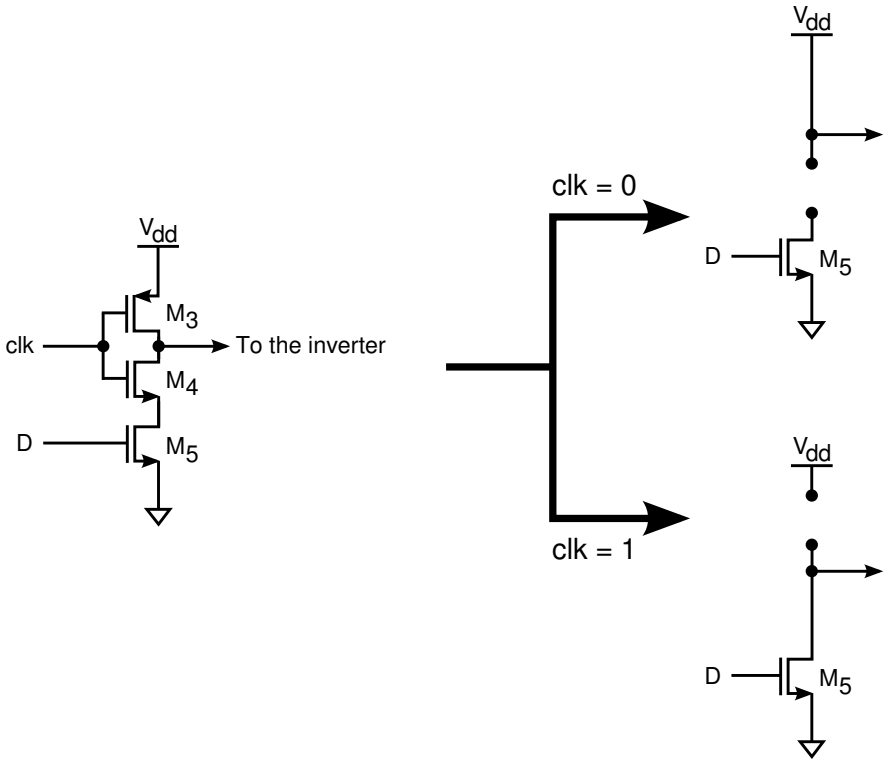

Fig. 4.11: Operation of the dynamic logic input control stage.

The skew between the clock signals clk and $clk1$ determines the duration of the current pulse. As soon as $clk1$ rises to "1", the current path is maintained only until C_s is charged to $V_{dd} - V_{th}$. At this point, the gate-to-source voltage of M_1 is equal to V_{th} and M_1 enters

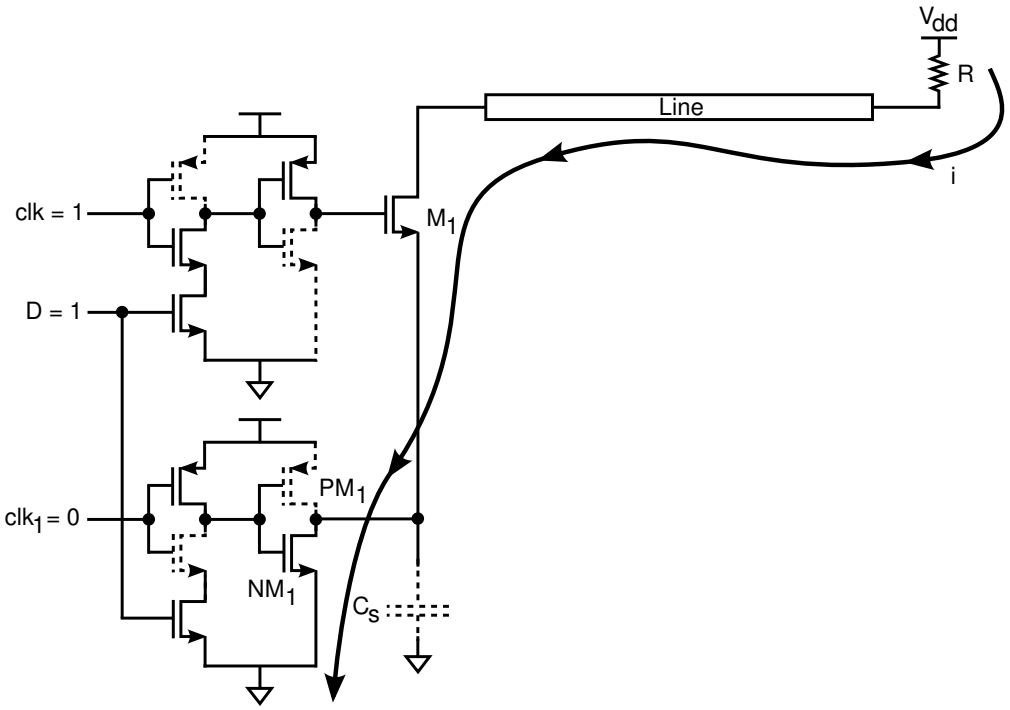

Fig. 4.12: Switched current path flowing through transistors M_1 and NM_1.

the subthreshold regime. Further, C_s is mainly charged through PM_1 up to V_{dd}, where M_1 is completely in cut-off and the current flow on the line is stopped.

The bypass capacitor C_s has a small value of only a few fF, nevertheless it has the important role of sinking a considerable amount of the high frequency current pulse. As soon as the current path is closed by M_1, the voltage on C_s increases with approx. 200 mV, thus helping decrease the current driven by transistor M_1. This reduces both the size requirements for M_1 and the overall static power dissipation.

Fig. 4.13 shows the waveforms of the two clock signals and the corresponding differential outputs. A "1" in the data signal is transmitted as a short current pulse while clk is high and $clk1$ is low. Consequently, the flowing current causes a slight voltage drop in the line from V_{dd} to the sum of the drain-to-source voltages of M_1 and NM_1. Similarly, a zero is transmitted through a pulse on the complementary line. When the clock signals switch back to "0", $clk1$ must stay low enough to ensure the complete discharging of C_s. Otherwise, a residual charge would accumulate on the capacitor over several clock cycles and would eventually prevent M_1 from switching on, as the gate-to-source voltage will be less than V_{th}. In addition, for modeling purposes, it is important to note that the main driving transistors M_1 and M_2 operate always in saturation during the current flow, and in cut-off respectively shortly in subthreshold regime when the current is interrupted.

The behavior of the PCM circuit has been analyzed in detail through several simulations under various conditions (clock frequencies, component values, interconnect length, supply voltages, and body bias values). The waveforms presented in this section have been obtained using the values from Tab. 4.1. Here, f is the clock frequency, V_b represents

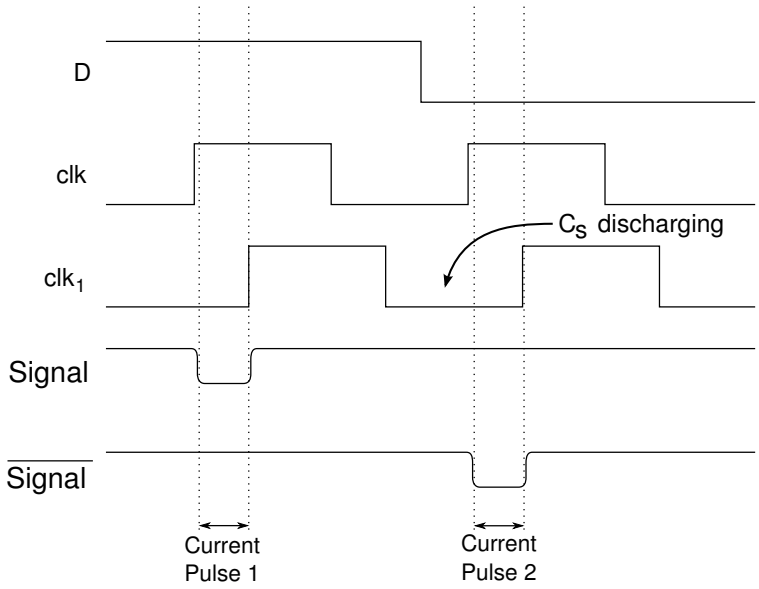

Fig. 4.13: Clock synchronization and output signals transmitting current pulses on the differential line.

V_{dd}	1.0 V
R	80 Ω
f	2.0 GHz
C_s	10 fF
$Wire\ length$	300 µm
W_{M_1}	3 µm
W_{M_2}	3 µm
W_{NM_1}	3 µm
W_{NM_2}	3 µm
L	80 nm
V_b	0 V
$Trans.\ analysis\ step$	0.2 ps

Tab. 4.1: Example values for the simulations.

the body bias, and the transient analysis step has been set to 0.2 ps.

The 1 GHz clock signals separated by a delay of 0.35 ns are plotted in Fig. 4.14. On the rising edge of clk, a current starts to flow through one of the differential interconnect lines and the voltage on the $Signal$ line drops slightly, as illustrated by the near-end and far-end waveforms in Fig. 4.14. When the data signal is on "0", a similar voltage drop occurs on the complementary line $\overline{Signal}$ between the positive edges of clk and $clk1$. It is to be noted, that the voltage at the far-end of the line drops to the following value:

$$V_{far-end} = V_{dd} - I_{pulse} \cdot R \tag{4.31}$$

This equation is verified by the results from Fig. 4.14 ($V_{far-end}$ = 921.1 mV) and by the current value from Fig. 4.15 (I_{pulse} = 985.3 µA).

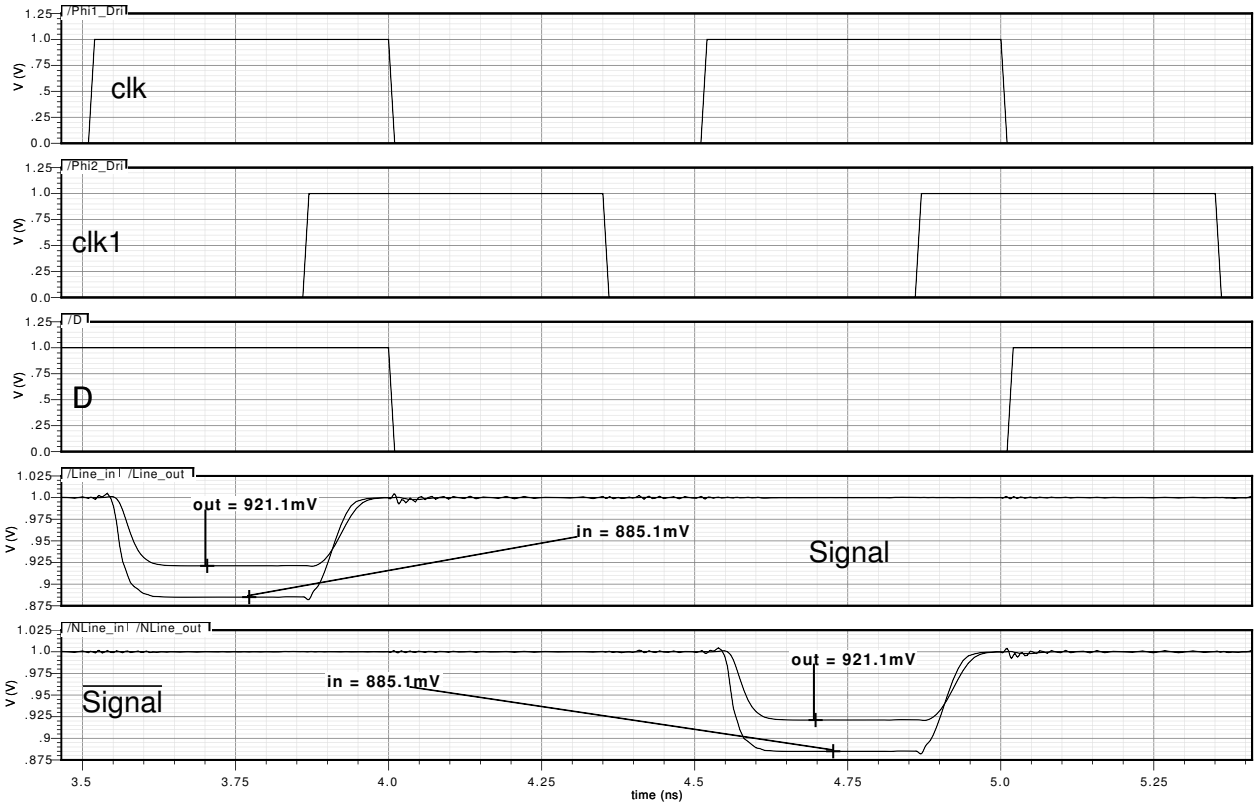

Fig. 4.14: Waveforms of the clock and data signals and the corresponding voltages at the near and far end of the differential line.

Further, Fig. 4.15 shows the current pulses issued on the two interconnect lines, which have a 0.35 ns duration, corresponding to the delay between the two clock signals. As the current starts to flow, the source voltage of M_1 (equal to the voltage on C_s) increases to approx. 200 mV, where the two transistors M_1 and NM_1 drive the total current on the line. As soon as $clk1$ switches to "1", NM_1 is turned off, and C_s is charged by both currents through M_1 and PM_1. Finally, after the falling edge of clk, M_1 is turned off and C_s is charged completely to V_{dd} by the current through PM_1. At the same time, the drain voltage of M_1 is only influenced by the current flowing through the line. Thus, the drain voltage drops between the rising edges of the two clocks to a value that is well approximated by:

$$V_{d,M_1} \approx V_{dd} - I_{pulse} \cdot R - I_{pulse} \cdot R_{line} \tag{4.32}$$

where R_{line} is the parasitic resistance of the interconnect line. For the 300 µm line in this example (in the considered 90 nm CMOS technology), a parasitic resistance of 36.67 Ω has been estimated, which together with the simulated results from Fig. 4.15 verifies the above equation. The terminal voltages of transistor M_1 during the switching period will be next utilized to compute the driving current using the transistor model from Sec. 4.1.

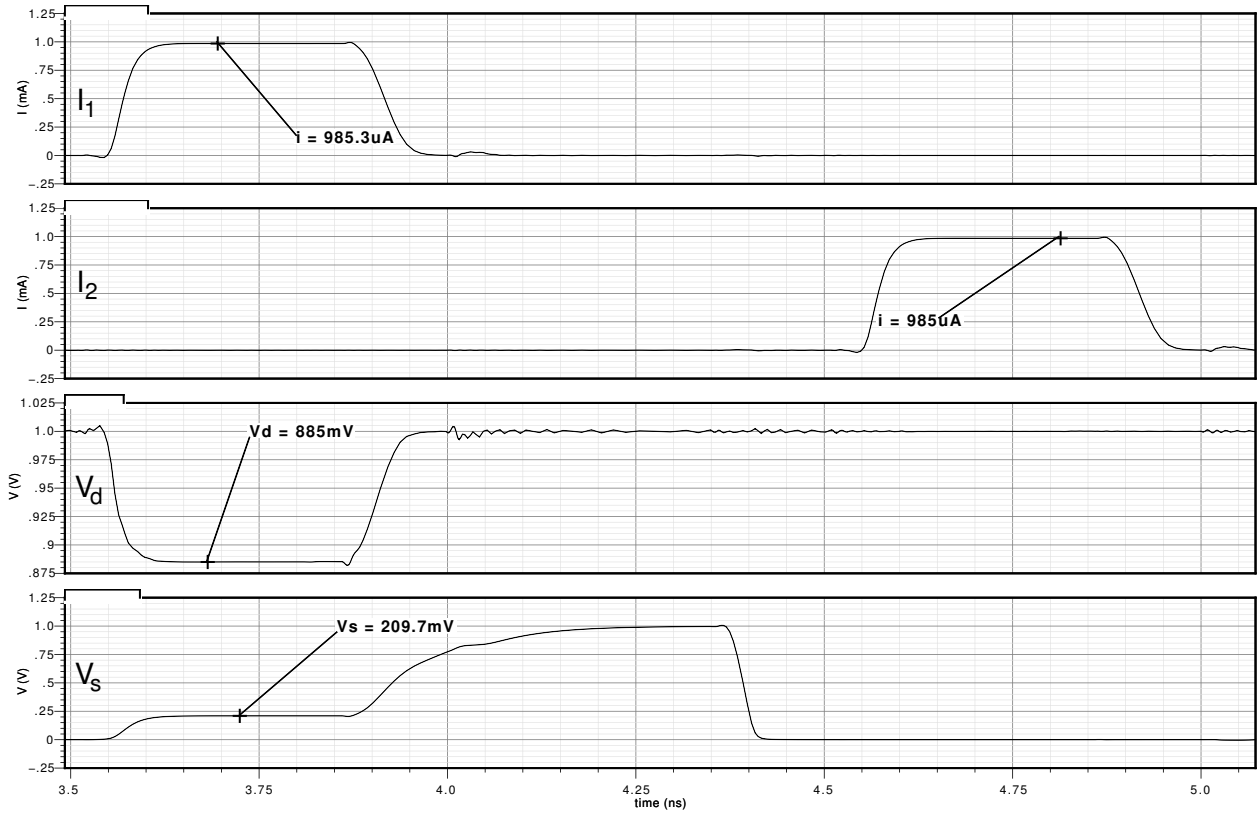

Fig. 4.15: Current pulses on the interconnect lines and the corresponding drain and source voltages for transistor M_1.

4.2.2 Equivalent Current-Source Circuit Model

Using the current-source transistor-level model developed in Sec. 4.1, the driver circuit and the current-mode line are reduced to the equivalent circuit from Fig. 4.16. The parasitic resistance and inductance of the interconnect line are lumped into the values R and L, while R_1 represents the small resistance connecting the current-mode line to the supply voltage. The first capacitance C_1 is the sum of the parasitic capacitances connected to the drain of M_1 and is evaluated as:

$$C_1 = C_{gd,eq} + C_{db} \tag{4.33}$$

where $C_{gd,eq}$ is the equivalent gate-to-drain capacitance of M_1 and C_{db} is the drain-to-bulk capacitance of the reverse-biased pn junction. In the following, we employ equations derived from the BSIM4.3 [167] capacitance models and adapted to compute the parasitic capacitances of the transistors connected to the interconnect line at the driver and receiver sides.

It is important to remember from Sec. 4.2.1, that M_1 operates only in the saturation or subthreshold/cut-off regimes. As a consequence, the only component of C_{db} is the drain-bulk overlap capacitance. Due to the Miller effect considered when connecting the series

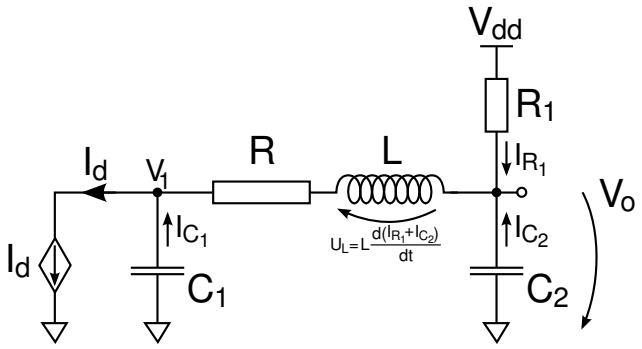

Fig. 4.16: General line model with current-mode driver.

capacitance to the ground, the equivalent gate-to-drain capacitance is computed as:

$$C_{gd,eq} = 2 \cdot C_{gd} \approx 2 \cdot \boldsymbol{C_{gdo}} \cdot W_{active} \tag{4.34}$$

where $\boldsymbol{C_{gdo}}$[2] represents the non-LDD region drain-gate overlap capacitance per unit width. Further, the active width is evaluated from the channel length L and width W using the following relationship:

$$W_{active} = W - 2 \cdot \left(\boldsymbol{D_{wc}} + \frac{\boldsymbol{W_{lc}}}{L^{\boldsymbol{W_{ln}}}} + \frac{\boldsymbol{W_{wc}}}{W^{\boldsymbol{W_{wn}}}} + \frac{\boldsymbol{W_{wlc}}}{L^{\boldsymbol{W_{ln}}} W^{\boldsymbol{W_{wn}}}} \right) \tag{4.35}$$

where $\boldsymbol{D_{wc}}$, $\boldsymbol{W_{lc}}$, $\boldsymbol{W_{wc}}$, and $\boldsymbol{W_{wlc}}$ are BSIM-model channel-width offset parameters.

The drain-to-bulk capacitance C_{db} depends on the considered voltage swing on the drain during the current pulse and on the voltage applied to the bulk. The use of averaging factors permits the computation of C_{gd} from the zero-bias junction capacitances as:

$$C_{db} = K_{eq,bottom} \cdot C_{j0,bottom} + K^{gate}_{eq,sidewall} \cdot C^{gate}_{j0,sidewall} + K^{isolation}_{eq,sidewall} \cdot C^{isolation}_{j0,sidewall} \tag{4.36}$$

Here, the bottom junction capacitance at the drain is evaluated as:

$$C_{j0,bottom} = \boldsymbol{C_{jd}} \cdot W_{active} \cdot \left(\boldsymbol{D_{mci}} + \boldsymbol{D_{mcg}} + \boldsymbol{L_{int}} + \frac{\boldsymbol{L_l}}{L^{\boldsymbol{L_{ln}}}} + \frac{\boldsymbol{L_w}}{W^{\boldsymbol{L_{wn}}}} + \frac{\boldsymbol{L_{wl}}}{L^{\boldsymbol{L_{ln}}} W^{\boldsymbol{L_{wn}}}} \right) \tag{4.37}$$

where $\boldsymbol{C_{jd}}$ is the bottom junction capacitance per unit area at the drain, $\boldsymbol{D_{mci}}$ is the distance from the drain contact center to the isolation edge, and $\boldsymbol{D_{mcg}}$ is the distance from the drain contact center to the gate edge. The sum $\boldsymbol{D_{mci}} + \boldsymbol{D_{mcg}}$ represents the exposed (uncovered) drain diffusion length, while the remaining terms estimate the lateral diffusion overlap.

Similarly, the gate and isolation sidewall capacitances are evaluated using the follow-

[2]Similar to Sec. 4.1.1, all symbols written in bold represent SPICE-model parameters dependent on the manufacturing process.

ing relationships:

$$C_{j0,sidewall}^{gate} = \boldsymbol{C_{jswgd}} \cdot W_{active} \tag{4.38}$$

$$C_{j0,sidewall}^{isolation} = \boldsymbol{C_{jswd}} \left[W_{active} + 2 \left(\boldsymbol{D_{mci}} + \boldsymbol{D_{mcg}} + \boldsymbol{L_{int}} + \frac{\boldsymbol{L_l}}{L^{\boldsymbol{L_{ln}}}} + \frac{\boldsymbol{L_w}}{W^{\boldsymbol{L_{wn}}}} + \frac{\boldsymbol{L_{wl}}}{L^{\boldsymbol{L_{ln}}} W^{\boldsymbol{L_{wn}}}} \right) \right] \tag{4.39}$$

where $\boldsymbol{C_{jswgd}}$ and $\boldsymbol{C_{jswd}}$ are the gate-edge, respectively isolation-edge sidewall junction capacitances per unit length.

The averaging constant of the bottom capacitance is computed from the following relationship:

$$K_{eq,bottom} = \frac{-\boldsymbol{P_{bd}}^{\boldsymbol{M_{jd}}}}{(V_{high} - V_{low})(1 - \boldsymbol{M_{jd}})} \left[(\boldsymbol{P_{bd}} - V_{high})^{1-\boldsymbol{M_{jd}}} - (\boldsymbol{P_{bd}} - V_{low})^{1-\boldsymbol{M_{jd}}} \right] \tag{4.40}$$

where $\boldsymbol{P_{bd}}$ is the bottom junction built-in potential at the drain and $\boldsymbol{M_{jd}}$ is the bulk junction bottom grading coefficient. The voltages V_{high} and V_{low} are defined as the maximum and minimum reverse bias voltages during the output transition for which the delay is estimated. If e.g. the output voltage swings between V_{oi} and V_{of}, then:

$$V_{high} = \begin{cases} V_b - V_{oi}, & \text{if } |V_b - V_{oi}| > |V_b - V_{of}| \\ V_b - V_{of}, & \text{otherwise} \end{cases} \tag{4.41}$$

$$V_{low} = \begin{cases} V_b - V_{oi}, & \text{if } |V_b - V_{oi}| < |V_b - V_{of}| \\ V_b - V_{of}, & \text{otherwise} \end{cases} \tag{4.42}$$

In this case, the output voltage travels during the from $V_{oi} = V_{dd}$ to the value $V_{of} = V_{d,M_1}$ given by (4.32).

Similarly, the averaging constants of the sidewall capacitances are computed using the following bias-dependent relationships:

$$K_{eq,sidewall}^{gate} = \frac{-\boldsymbol{P_{bswgd}}^{\boldsymbol{M_{jswgd}}} \left[(\boldsymbol{P_{bswgd}} - V_{high})^{1-\boldsymbol{M_{jswgd}}} - (\boldsymbol{P_{bswgd}} - V_{low})^{1-\boldsymbol{M_{jswgd}}} \right]}{(V_{high} - V_{low})(1 - \boldsymbol{M_{jswgd}})} \tag{4.43}$$

$$K_{eq,sidewall}^{isolation} = \frac{-\boldsymbol{P_{bswd}}^{\boldsymbol{M_{jswd}}} \left[(\boldsymbol{P_{bswd}} - V_{high})^{1-\boldsymbol{M_{jswd}}} - (\boldsymbol{P_{bswgd}} - V_{low})^{1-\boldsymbol{M_{jswgd}}} \right]}{(V_{high} - V_{low})(1 - \boldsymbol{M_{jswd}})} \tag{4.44}$$

where $\boldsymbol{P_{bswgd}}$ and $\boldsymbol{P_{bswd}}$ are the gate-edge and the isolation-edge sidewall junction built-in potentials, respectively, while $\boldsymbol{M_{jswgd}}$ and $\boldsymbol{M_{jswd}}$ represent the gate-edge and isolation-edge sidewall junction capacitance grading coefficients.

The second capacitance C_2 in the model from Fig. 4.16 sums the gate capacitance of the input transistor in the receiver and the parasitic capacitance of the interconnect line. The gate capacitance is estimated as:

$$C_g = C_{ovs} + C_{ovd} + C_{gc} \approx (\boldsymbol{C_{gso}} + \boldsymbol{C_{gdo}}) \cdot W_{active} + \frac{\boldsymbol{\varepsilon_{r,ox}} \varepsilon_0}{\boldsymbol{T_{ox,e}}} \cdot W_{active} \cdot L_{active} \tag{4.45}$$

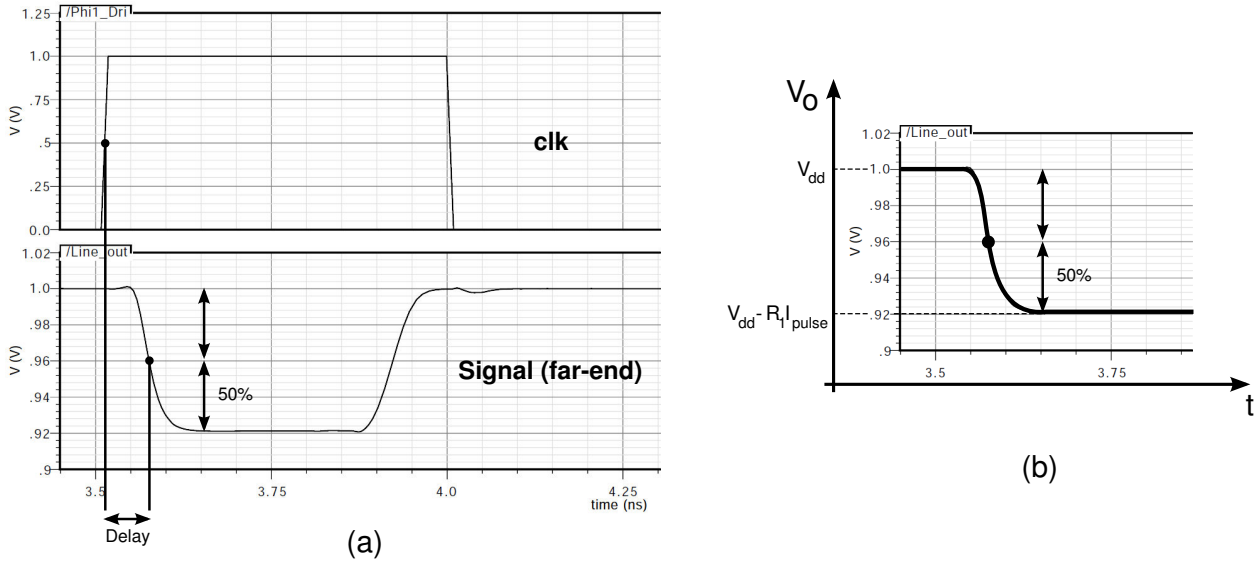

Fig. 4.17: Line delay definition at 50% swing point (a) and the output voltage of the circuit model (b) used to compute the delay.

Finally, the wire resistance, inductance, and capacitance values are estimated using the technology-dependent analytic expressions specified by the Predictive Technology Model (PTM) [116].

4.2.3 Analytic Model for Delay and Energy Consumption

The propagation time over the PCM line is defined when the far-end signal reaches 50% of its total swing, as illustrated by the waveforms in Fig. 4.17(a). Since the delay is defined on the falling edge, we are interested to model only the falling part of the characteristic and to estimate the time when the voltage reaches the 50% point, as shown by the V_o characteristic in Fig. 4.17(b). In the following, we use the circuit from Fig. 4.16 to compute the expression of $V_o\,(t)$ and then we will identify the time t_0 at which $V_o\,(t_0)$ reaches half of the swing.

The swing of the driver during the start of the current flow through the interconnect is bounded by two conditions. At the initial time, the transistor M_1 is in cut-off, thus:

$$I_d\,(0) = 0 \;\Rightarrow\; V_o\,(0) = V_{dd} \tag{4.46}$$

The drain current then raises up to its maximum value I_{pulse}, where it remains until the positive edge of $clk1$. If we assume the voltage model from Fig. 4.17(b), the second boundary condition is:

$$V_o\,(\infty) = V_{dd} - R_1 I_{pulse} \tag{4.47}$$

Note that although the voltage on the line increases back to V_{dd} at the end of the current pulse, only the falling edge is relevant for computing the delay.

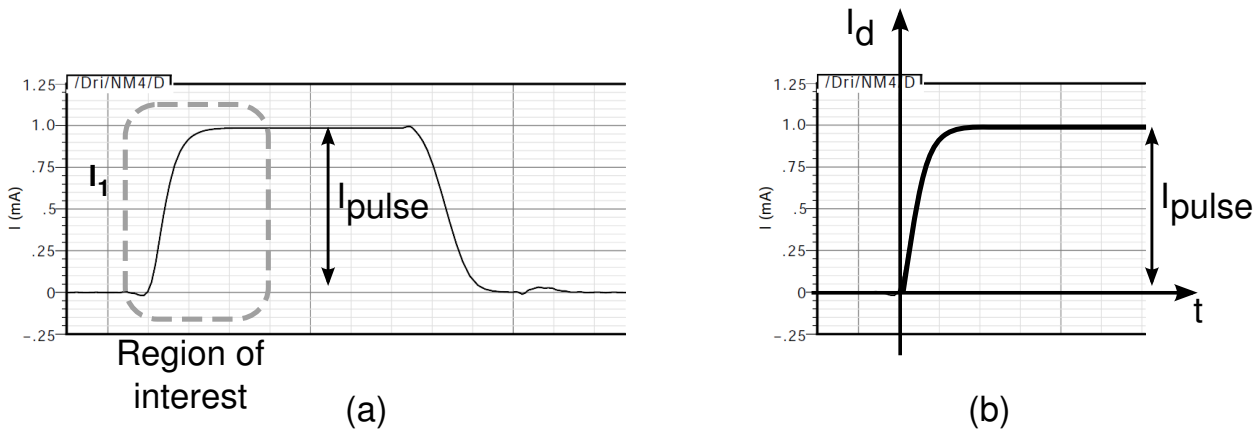

Fig. 4.18: Current pulse shape (a) and the model approximation for computing the delay (b).

Similarly, from the current pulse characteristic shown in Fig. 4.18(a), only the rising edge and the maximum value I_{pulse} contribute to the delay evaluation. Therefore, for the delay model, the current I_d can be approximated with the characteristic drawn in Fig. 4.18(b). We further assume that I_d switches from 0 to I_{pulse} according to the following characteristic:

$$I_d(t) = I_{pulse}\left(1 - e^{-\alpha t}\right) \tag{4.48}$$

which can be adjusted to fit the switching profile of the driving transistor. For instance, the parameter value $\alpha = 10^{10}$ corresponds to a current rise time $t_r \approx 50\,\text{ps}$. It is further important to notice, that since M_1 operates in saturation, the almost constant current approximation from Fig. 4.18(b) and (4.48) remains valid within the switching time frame.

The current through the resistor R_1 is given by $I_{R_1} = \frac{V_{dd} - V_o(t)}{R_1}$, while the output voltage can be expressed in terms of the charge on C_2 as:

$$V_o(t) = \frac{Q_2(t)}{C_2} = \frac{Q_2(0) - \int_0^t I_{C_2}(\tau)\, d\tau}{C_2} \tag{4.49}$$

Note that at the initial time $t = 0$, both C_1 and C_2 are charged with the non-zero charges $Q_1(0)$ and $Q_2(0)$. Since at $t = 0$ all the currents are zero, the following holds:

$$\frac{Q_1(0)}{C_1} = \frac{Q_2(0)}{C_2} = V_{dd} \tag{4.50}$$

If we neglect the inductivity effects in a first approximation, the following relationships can be written for the voltage V_1 on the capacitor C_1 (see Fig. 4.16):

$$V_1(t) = V_o(t) - R\left[I_{R_1}(t) + I_{C_2}(t)\right] \tag{4.51}$$

$$V_1(t) = \frac{Q_1(0) - \int_0^t I_{C_1}(\tau)\, d\tau}{C_1} \tag{4.52}$$

Nevertheless, the current sum must be equal to: $I_{C_1}(t) + I_{C_2}(t) + I_{R_1}(t) = I_d(t)$, such

that the following system of differential equations can be written:

$$R_1 I_{R_1}(t) = \frac{1}{C_2}\int_0^t I_{C_2}(\tau)\,d\tau \tag{4.53}$$

$$\frac{1}{C_1}\int_0^t I_{C_1}(\tau)\,d\tau = \frac{1}{C_2}\int_0^t I_{C_2}(\tau)\,d\tau + R\left[I_{R_1}(t) + I_{C_2}(t)\right] \tag{4.54}$$

$$I_{C_1}(t) = I_d(t) - I_{C_2}(t) - I_{R_1}(t) \tag{4.55}$$

After differentiating (4.53) and (4.54) and combining with (4.55), the following second-order nonhomogeneous ordinary differential equation (ODE) is obtained:

$$\frac{d^2 I_{R_1}(t)}{dt^2} + \frac{\left(R_1 + R + R_1\frac{C_2}{C_1}\right)}{RR_1C_2}\frac{dI_{R_1}(t)}{dt} + \frac{1}{RR_1C_1C_2}I_{R_1}(t) = \frac{I_d(t)}{RR_1C_1C_2} \tag{4.56}$$

Replacing $I_{R_1}(t) = e^{\lambda t}$, the associated characteristic equation becomes:

$$\lambda^2 + \frac{R_1 + R + R_1\frac{C_2}{C_1}}{RR_1C_2}\lambda + \frac{1}{RR_1C_1C_2} = 0 \tag{4.57}$$

with the solutions:

$$\lambda_{1,2} = \frac{-\left(R_1 + R + R_1\frac{C_2}{C_1}\right) \pm \sqrt{\left(R_1 + R + R_1\frac{C_2}{C_1}\right)^2 - \frac{4RR_1C_2}{C_1}}}{2RR_1C_2} \tag{4.58}$$

By applying the method of variation of constants, a particular solution can be written as $I_{R_1p}(t) = u_1(t)\,e^{\lambda_1 t} + u_2(t)\,e^{\lambda_2 t}$, from which the following equation system results:

$$u_1'(t) = -\frac{I_d(t)}{(\lambda_2 - \lambda_1)\,RR_1C_1C_2}e^{-\lambda_1 t} \tag{4.59}$$

$$u_2'(t) = \frac{I_d(t)}{(\lambda_2 - \lambda_1)\,RR_1C_1C_2}e^{-\lambda_2 t} \tag{4.60}$$

Recalling that $I_d(t) = I_{pulse}\left(1 - e^{-\alpha t}\right)$, after integration the variation functions become:

$$u_1(t) = -\frac{I_{pulse}}{(\lambda_2 - \lambda_1)\,RR_1C_1C_2}\left[-\frac{e^{-\lambda_1 t}}{\lambda_1} + \frac{e^{-(\lambda_1+\alpha)t}}{\lambda_1 + \alpha}\right] \tag{4.61}$$

$$u_2(t) = \frac{I_{pulse}}{(\lambda_2 - \lambda_1)\,RR_1C_1C_2}\left[-\frac{e^{-\lambda_2 t}}{\lambda_2} + \frac{e^{-(\lambda_2+\alpha)t}}{\lambda_2 + \alpha}\right] \tag{4.62}$$

Finally, the general solution of the inhomogeneous equation (4.56) becomes:

$$I_{R_1}(t) = K_1 e^{\lambda_1 t} + K_2 e^{\lambda_2 t} + \frac{I_{pulse}}{(\lambda_2 - \lambda_1)\,RR_1C_1C_2}\left[\frac{1}{\lambda_1} - \frac{1}{\lambda_2} + \left(\frac{1}{\lambda_2 + \alpha} - \frac{1}{\lambda_1 + \alpha}\right)e^{-\alpha t}\right] \tag{4.63}$$

From the initial condition $I_{R_1}(0) = 0$, we obtain a first relationship between K_1 and K_2:

$$K_2 = -K_1 - \frac{I_{pulse}}{RR_1C_1C_2}\left[\frac{1}{\lambda_1\lambda_2} - \frac{1}{(\lambda_1 + \alpha)(\lambda_2 + \alpha)}\right] \tag{4.64}$$

The condition $I_{R_1}(\infty) = I_{pulse}$ is verified without further constraints. Differentiating (4.53) gives $I_{C_2}(t) = R_1 C_2 \frac{dI_{R_1}(t)}{dt}$, which after replacing the derivative of (4.63) and considering the initial condition $I_{C_2} = 0$, gives the second relationship for computing K_1 and K_2:

$$R_1 C_2 K_1 \lambda_1 + R - 1C - 2K_2 \lambda_2 + \frac{\alpha I_{pulse}}{RC_1 (\lambda_1 + \alpha)(\lambda_2 + \alpha)} = 0 \tag{4.65}$$

After finding the expression of $I_{R_1}(t)$, the output voltage is obtained as follows:

$$V_o(t) = V_{dd} - R_1 K_1 e^{\lambda_1 t} - R_1 K_2 e^{\lambda - 2t} - \frac{I_{pulse}}{(\lambda_2 - \lambda_1) RC_1 C_2} \left[\frac{1}{\lambda_1} - \frac{1}{\lambda_2} + \left(\frac{1}{\lambda_2 + \alpha} - \frac{1}{\lambda_1 + \alpha} \right) e^{-\alpha t} \right] \tag{4.66}$$

which also satisfies the limit conditions $V_o(0) = V_{dd}$ and $V_o(\infty) = V_{dd} - R_1 I_{pulse}$.

An important observation has to be made. Depending on the values of the line and device parasitics, the roots of the characteristic equation can be complex:

$$\lambda_1 = \Re_\lambda + j\Im_\lambda \tag{4.67}$$

$$\lambda_2 = \Re_\lambda - j\Im_\lambda \tag{4.68}$$

As a consequence, the output voltage will also be complex. Thus, for evaluating the delay, only the real part is considered:

$$\begin{aligned} \Re_{V_o(t)} =& e^{\Re_\lambda t} \sin(\Im_\lambda t)(\Im_{K_1} - \Im_{K_2}) R_1 - e^{\Re_\lambda t} \cos(\Im_\lambda t) R_1 (\Re_{K_1} + \Re_{K_2}) - \\ & \frac{I_{pulse}}{RC_1 C_2 (\Im_\lambda^2 + \Re_\lambda^2)} + \frac{e^{-\alpha t} I_{pulse}}{RC_1 C_2 \left(\Im_\lambda^2 + (\alpha + \Re_\lambda)^2 \right)} + V_{dd} \end{aligned} \tag{4.69}$$

where the constants K_1 and K_2 are also complex quantities, computed as:

$$\Re_{K_1} = -\frac{\alpha I_{pulse} (\alpha + 2\Re_\lambda)}{2RC_1 C - 2R_1 (\Im_\lambda^2 + \Re_\lambda^2) \left(\Im_\lambda^2 + (\alpha + \Re_\lambda)^2 \right)} \tag{4.70}$$

$$\Im_{K_1} = -\frac{\alpha I_{pulse} (-\Im_\lambda^2 + \Re_\lambda (\alpha + \Re_\lambda))}{2RC_1 C_2 \Im_\lambda R_1 (\Im_\lambda^2 + \Re_\lambda^2) \left(\Im_\lambda^2 + (\alpha + \Re_\lambda)^2 \right)} \tag{4.71}$$

$$\Re_{K_2} = -\Re_{K_1} - \frac{\alpha I_{pulse} (\alpha + 2\Re_\lambda)}{RC_1 C_2 R_1 \left(\Im_\lambda^2 + (\alpha + \Re_\lambda)^2 \right)} \tag{4.72}$$

$$\Im_{K_2} = -\Im_{K_1} \tag{4.73}$$

If, in addition, the inductivity effects are important and must be considered, the equation (4.51) is replaced by the following:

$$V_1(t) = V_o(t) - L \left[\frac{dI_{R_1}(t)}{dt} + \frac{dI_{C_2}(t)}{dt} \right] - R \left[I_{R_1}(t) + I_{C_2}(t) \right] \tag{4.74}$$

This leads to the following equation:

$$\begin{aligned} & LR_1 C_2 \frac{d^3 I_{R_1}(t)}{dt^3} + (L + RR_1 C_2) \frac{d^2 I_{R_1}(t)}{dt^2} + \\ & \left(R + R_1 + \frac{R_1 C_2}{C_1} \right) \frac{dI_{R_1}(t)}{dt} + \frac{1}{C_1} I_{R_1}(t) = \frac{I_d(t)}{C_1} \end{aligned} \tag{4.75}$$

which is a third-order nonhomogeneous ODE. The solution is found in a similar way, with the only particularity of a third-order characteristic equation.

Finally, after obtaining the expression of the output voltage as in (4.69) or (4.69), the propagation delay is obtained as:

$$Delay = t_0 \Big|_{V_o(t_0)=V_{dd}-0.5\cdot R_1 I_{pulse}} \tag{4.76}$$

which corresponds to the time point at which $V_o(t)$ crosses the 50% swing point. The current I_{pulse} is evaluated using the voltage-dependent source model from Sec. 4.1. A fast solution of the transcendental equation required by (4.76) is presented in Sec. 4.4.4.

The dynamic energy consumption at the circuit level is estimated during the active communication period, where the circuit is switching. For this purpose, the total parasitic capacitance of the circuit is estimated as shown in this section. After that, the dynamic energy is computed as a function of the communication load (actually the switching activity, which is dependent on the communication load), of the total capacitance (circuit and interconnect), and of the communication segment width:

$$E_d \approx f(L_c) \frac{C_{total} V_{dd}^2}{W_s} \tag{4.77}$$

The dynamic energy of each segment is included in the overall system-level macromodel as explained in Sec. 4.4.5.

Finally, the static energy consumption is estimated from the sum of the leakage currents in the driver and receiver during the idle time and the static current flow through the line during the pulse window:

$$E_s \approx \left(I_{leakage}^{driver} + I_{leakage}^{receiver}\right) V_{dd} t_{idle} + I_{pulse} V_{dd} t_{pulse} \tag{4.78}$$

Both the leakage currents and the signaling current are estimated using the statistical transistor model. The integration in the macromodel structure is discussed in Sec. 4.4.5 as well.

4.2.4 Performance Evaluation under Voltage Scaling and Body Biasing

The delay plot in Fig. 4.19(a) shows a total delay variation of approx. 430 ps with interconnect length, for lines up to 3 cm long. This relatively small increase confirms the usage of PCM signaling circuits for global repeaterless lines. It is to be noted, that the driver itself exhibits a 70 ps delay, which is not dependent on the line length. This fixed delay represents in fact the lowest delay threshold, beyond which PCM-driven lines show their speed advantage.

A relatively higher static energy consumption is the main downside of current-driven lines, which is also illustrated by the plot from Fig. 4.19(b). Nevertheless, due to the short

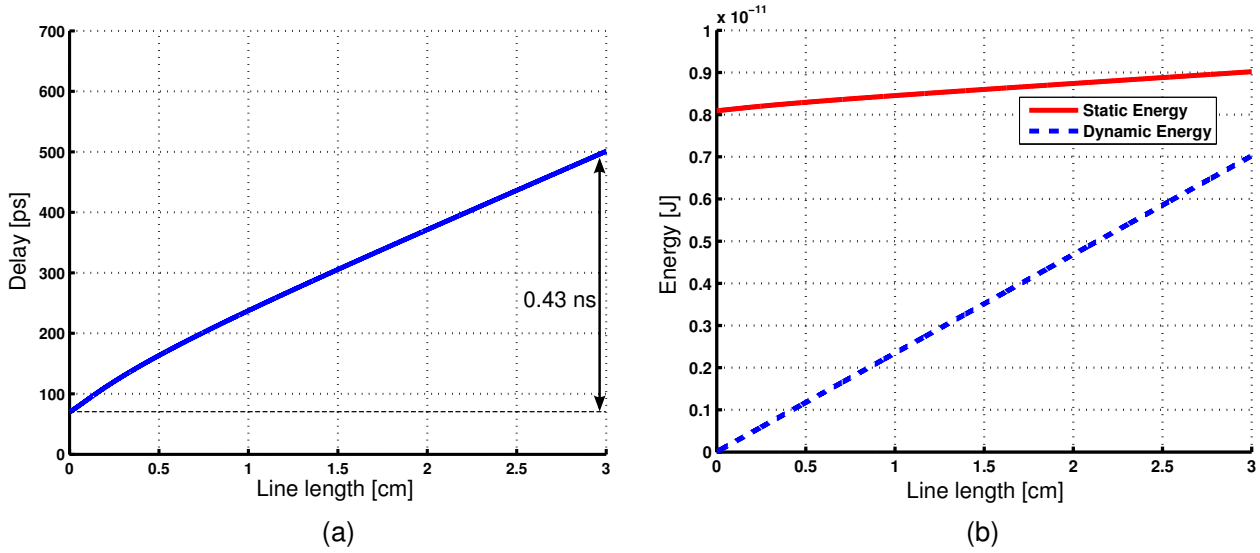

Fig. 4.19: Delay (a) and energy (b) variation with interconnect line length.

current pulses, the static energy is limited to a value around 9 pJ. For the evaluations, a pulse current duration equal to the delay on the line and an idle time of 1 ms for leakage evaluations have been assumed. This corresponds to an average power dissipation of 9 nW. It can be noticed that the relative variation with the wire length does not have a significant contribution to the total static energy value. On the other side, the dynamic energy is strongly dependent on the line capacitance, as indicated by (4.77), therefore most of its value is determined by the wire length.

The line delay is determined by the drive current which charges/discharges the parasitic capacitances. Due to the circuit structure of the PCM line, the drive current is mainly dependent on the voltage drop across the line, from the supply voltage connection at the receiver, to the drain-to-source voltage sum of the driving transistors. Consequently, a change in the body bias of the driver has only a negligible impact on the current value. Voltage scaling, on the other side, has a stronger influence on the signaling current, however it also changes the voltage from which the line capacitances are discharged. Nevertheless, due to the very small swing of the output voltage $V_{dd} - I_{pulse}R_1$ (R_1 is a small resistance, in the order of tens of Ω), the impact of voltage scaling on the overall line delay is also very small. Thus, body biasing and voltage scaling have a noticeable influence only on the small delay of the driver. Evaluations using both the previously-developed model and circuit simulations show a change in the driver delay of maximum 10 ps with voltage scaling and ± 3 ps with body biasing for the investigated range of wire lengths.

The effect of voltage scaling and body biasing on the energy consumption of the PCM communication segment has been investigated in Fig. 4.20. It can be seen that the static energy consumption is highly affected by the body bias, whereas supply voltage scaling exhibits a relatively smaller influence. As expected, reverse body biasing (negative V_{bs}) increases the threshold voltage, which leads to a significant leakage reduction. Forward body biasing (positive V_{bs}) has in the case of PCM signaling only the negative effect of

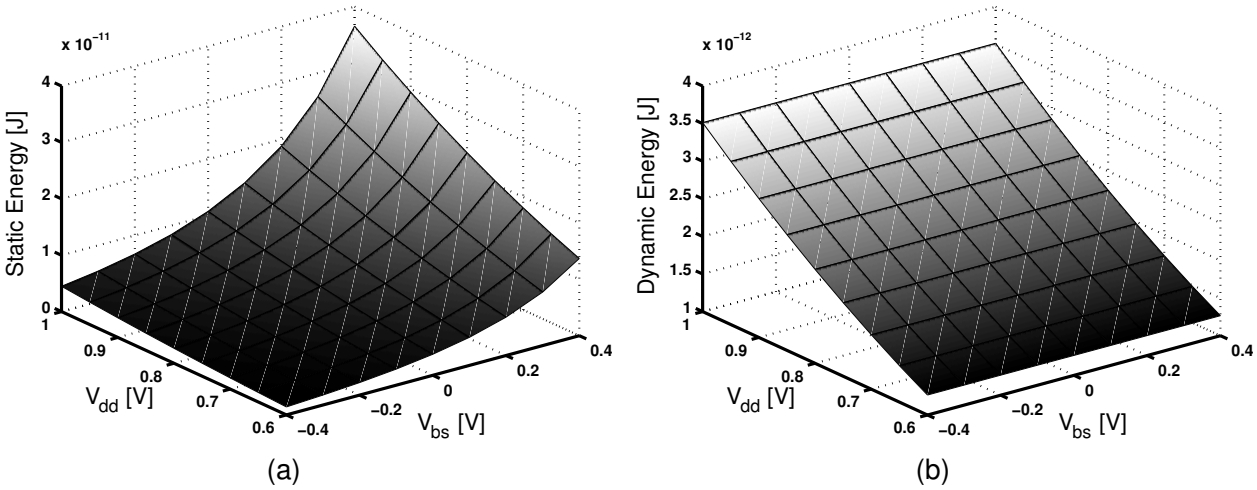

Fig. 4.20: Impact of voltage scaling and body bias on the static (a) and dynamic energy consumption (b) for a 15 mm interconnect line.

increasing the leakage, since the delay remains unaffected. It is further to be noted, that body biasing has a stronger impact on the static energy at higher supply voltages, since the static current on the line increases linearly with V_{dd}. As expected from (4.77), body biasing has no impact on dynamic energy consumption. Thus, voltage scaling is the main mechanism to control the dynamic energy at circuit level.

4.3 Voltage-Mode Signaling Model

To evaluate the advantages of the novel PCM signaling technique, a model for the classic voltage-mode signaling circuit is developed in this section. In addition, the integration of both models in the design space exploration allows a selection of the PCM circuit only when it brings a performance advantage over the classic signaling.

4.3.1 Equivalent Current-Source Circuit Model

The buffer circuit used for voltage-mode signaling is shown in Fig. 4.21. Assuming a symmetric inverter stage, the delay can be estimated e.g. from the pull-down cycle, where the NMOS transistor drives the line to ground. In this case, a short current flows through the line which discharges the gate capacitance of the receiver and the parasitic capacitance of the line. In this case, the equivalent circuit model includes the receiver capacitance, the line model, and a voltage-controlled current source together with a parasitic output capacitance for the driving stage, as illustrated in Fig. 4.21.

During the pull-down switch, the drain current I_d rises abruptly to a maximum value corresponding to the full-drive of transistor M_1 ($V_{gs} = V_{ds} = V_{dd}$), as shown by the waveforms in Fig. 4.22(a). When $V_{gs} = V_{dd}$, the PMOS at the output of the driver is turned off,

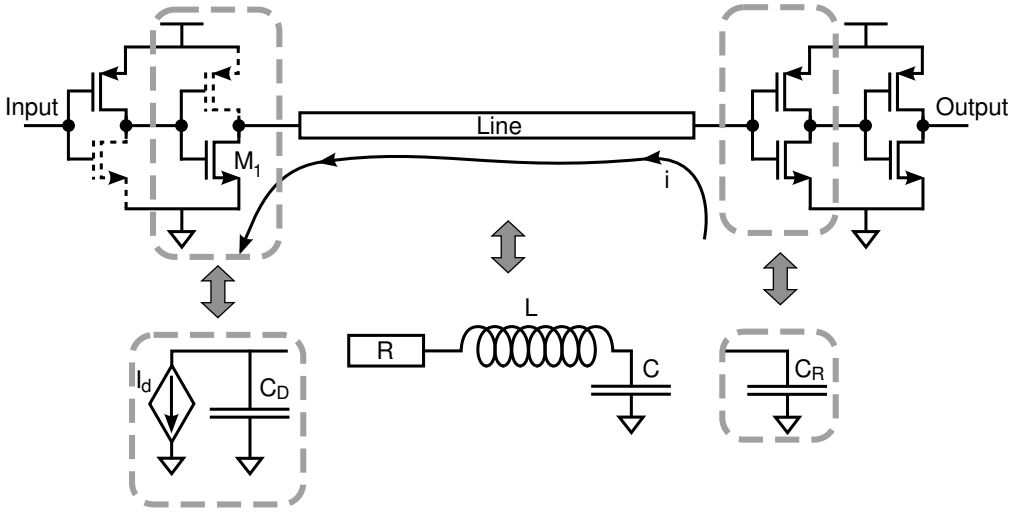

Fig. 4.21: Voltage-mode buffer circuit and equivalent circuit model.

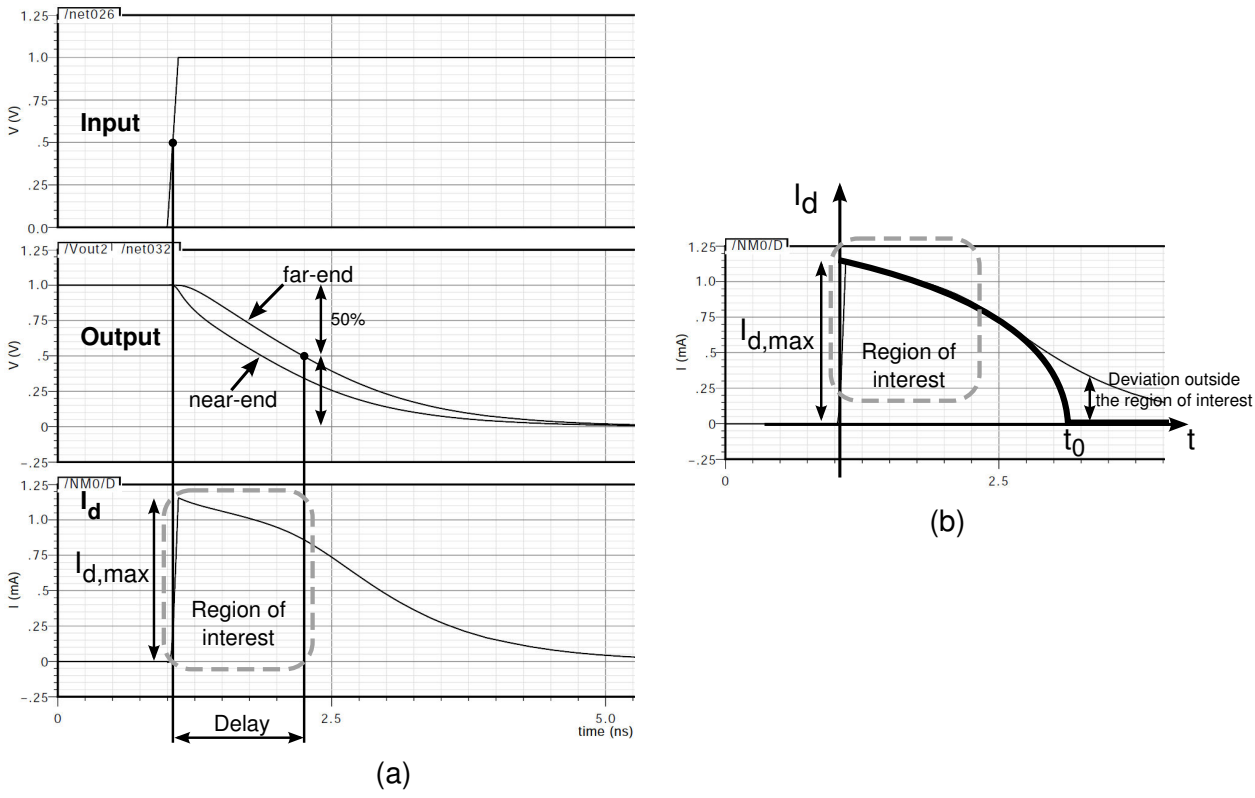

Fig. 4.22: Region of interest in the drain current characteristic for computing the delay (a) and exponential approximation for the delay model (b).

therefore the current sinked by the NMOS originates entirely from the load. After that, the capacitances on the line start to discharge and the voltage at the receiver gate starts to decrease. As discussed in Sec. 4.2.3, the propagation time is defined at 50% of the swing, which for the voltage-mode CMOS buffer occurs at $\frac{V_{dd}}{2}$. Thus, an accurate model for the drain current is required only until $V_o = \frac{V_{dd}}{2}$, as indicated in Fig. 4.22.

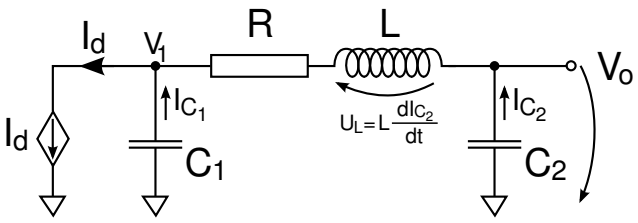

Fig. 4.23: Voltage-mode driver and line model.

Further, the identified region of interest from the drain current characteristic is approximated as shown in Fig. 4.22(b) with the following exponential function:

$$I_d(t) = \begin{cases} I_{d,max} + \gamma \left[1 - e^{\frac{t}{t_0} \ln\left(1+\frac{I_{d,max}}{\gamma}\right)}\right], & t < t_0 \\ 0, & t \geq t_0 \end{cases} \tag{4.79}$$

where $I_{d,max}$ is the maximum drain current of M_1 corresponding to $V_{gs} = V_{ds} = V_{dd}$, t_0 is the time at which $I_d = 0$ (in this partial approximation), and γ is a curvature fitting parameter. Typical values for the considered 90 nm technology are $\gamma = 0.5 \cdot 10^{-4}$ and $t_0 = 2.5$ ns for a 10 mm segment, respectively $t_0 = 1.16$ ns for a 10 µm segment.

4.3.2 Analytic Model for Delay and Energy Consumption

The equivalent circuit model derived from the observations made in the previous section is drawn in Fig. 4.23. Here, the current I_d is described by the relationship (4.79) and capacitance C_1 includes the parasitic output capacitances of the driver. The second capacitor C_2 models the sum of the line capacitance and the parasitic gate capacitances of the receiver.

Further, the transistor-level parasitic capacitances are estimated using BSIM4-derived equations with technology-dependent parameters, as described in Sec. 4.2.2. Similar to the PCM model, the line parasitics are estimated using the PTM analytic expressions [116].

In the considered case of a pull-down transition, the capacitances are charged up to the supply voltage, such that $Q_1(0) = C_1 V_{dd}$ and $Q_2(0) = C_2 V_{dd}$. Consequently, the time variation of the voltages on capacitances C_1 and C_2 can be written as:

$$V_1(t) = V_{dd} - \frac{1}{C_1} \int_0^t I_{C_1}(\tau)\, d\tau \tag{4.80}$$

$$V_o(t) = V_{dd} - \frac{1}{C_2} \int_0^t I_{C_2}(\tau)\, d\tau \tag{4.81}$$

The voltage V_1 can also be expressed in terms of the output voltage, as:

$$V_1(t) = V_o(t) - L\frac{dI_{C_2}(t)}{dt} - RI_{C_2}(t) \tag{4.82}$$

which, together with (4.80) gives:

$$\frac{1}{C_1} \int_0^t I_{C_1}(\tau)\, d\tau = \frac{1}{C_2} \int_0^t I_{C_2}(\tau)\, d\tau + L\frac{dI_{C_2}(t)}{dt} + RI_{C_2}(t) \tag{4.83}$$

By deriving (4.83) and considering that $I_{C_1}(t) = I_d(t) - I_{C_2}(t)$ we obtain the following second-order ODE:

$$\frac{d^2 I_{C_2}(t)}{dt^2} + \frac{R}{L} \cdot \frac{dI_{C_2}(t)}{dt} + \frac{C_1 + C_2}{LC_1C_2} I_{C_2} = \frac{I_d(t)}{LC_1} \tag{4.84}$$

Replacing $I_{C_2}(t) = e^{\lambda t}$ leads to the characteristic equation:

$$\lambda^2 + \frac{R}{L}\lambda + \frac{C_1 + C_2}{LC_1C_2} = 0 \tag{4.85}$$

with the solutions:

$$\lambda_{1,2} = \frac{-\frac{R}{L} \pm \sqrt{\left(\frac{R}{L}\right)^2 - \frac{4(C_1+C_2)}{LC_1C_2}}}{2} \tag{4.86}$$

A particular solution of the ODE can be found in the form $I_{C_2p}(t) = u_1(t)\, e^{\lambda_1 t} + u_2(t)\, e^{\lambda_2 t}$ with the variation functions described by:

$$u_1'(t) = -\frac{I_d(t)}{(\lambda_2 - \lambda_1) LC_1} e^{-\lambda_1 t} \tag{4.87}$$

$$u_2'(t) = \frac{I_d(t)}{(\lambda_2 - \lambda_1) LC_1} e^{-\lambda_2 t} \tag{4.88}$$

If we re-write the drain current from (4.79) as:

$$I_d(t)\Big|_{t<t_0} = I_1 - \gamma e^{\alpha t} \tag{4.89}$$

where $I_1 = I_{d,max} + \gamma$ and $\alpha = \frac{1}{t_0} \ln\left(1 + \frac{I_{d,max}}{\gamma}\right)$, integrating (4.87) and (4.88) gives:

$$u_1(t) = \frac{I_1}{(\lambda_2 - \lambda_1) LC_1\lambda_1} e^{-\lambda_1 t} + \frac{\gamma}{(\lambda_2 - \lambda_1) LC_1 (\alpha - \lambda_1)} e^{(\alpha - \lambda_1)t} \tag{4.90}$$

$$u_2(t) = -\frac{I_1}{(\lambda_2 - \lambda_1) LC_1\lambda_2} e^{-\lambda_2 t} - \frac{\gamma}{(\lambda_2 - \lambda_1) LC_1 (\alpha - \lambda_2)} e^{(\alpha - \lambda_2)t} \tag{4.91}$$

Finally, the general solution of (4.84) is obtained in the following form:

$$I_{C_2}(t) = K_1 e^{\lambda_1 t} + K_2 e^{\lambda_2 t} + \frac{I_1}{LC_1\lambda_1\lambda_2} - \frac{\gamma e^{\alpha t}}{LC_1 (\alpha - \lambda_1)(\alpha - \lambda_2)} \tag{4.92}$$

It is to be remembered, that (4.92) is valid only for $t < t_0$. Nevertheless, the model must be accurate only up to the time when V_o reaches the half of its full swing, which occurs before t_0, as shown in Fig. 4.22.

Verifying the initial condition $I_{C_2}(0) = 0$ gives the first equation for computing K_1 and K_2:

$$K_1 + K_2 + \frac{I_1}{LC_1\lambda_1\lambda_2} - \frac{\gamma}{LC_1 (\alpha - \lambda_1)(\alpha - \lambda_2)} = 0 \tag{4.93}$$

Recalling that $V_1(t) = V_o(t) - L\frac{dI_{C_2}(t)}{dt} - RI_{C_2}(t)$ and by admitting that $V_1(0) = V_o(0)$ (the initial condition before the pull-down transition), with $I_{C_2}(0) = 0$ from above, we obtain:

$$\left.\frac{dI_{C_2}(t)}{dt}\right|_{t=0} = 0 \tag{4.94}$$

which gives the second equation for finding K_1 and K_2:

$$K_1\lambda_1 + K_2\lambda_2 - \frac{\gamma\alpha}{LC_1(\alpha-\lambda_1)(\alpha-\lambda_2)} = 0 \tag{4.95}$$

with the results:

$$K_1 = \frac{1}{LC_1(\lambda_1-\lambda_2)}\left(\frac{\gamma}{\alpha-\lambda_1} + \frac{I_1}{\lambda_1}\right) \tag{4.96}$$

$$K_2 = \frac{\gamma}{LC_1(\alpha-\lambda_1)(\alpha-\lambda_2)} - \frac{I_1}{LC_1\lambda_1\lambda_2} - K_1 \tag{4.97}$$

Finally, the output voltage has the following expression:

$$\begin{aligned} V_o(t) =& V_{dd} - \frac{1}{C_2}\left[\frac{K_1}{\lambda_1}\left(e^{\lambda_1 t}-1\right) + \frac{K_2}{\lambda_2}\left(e^{\lambda_2 t}-1\right)\right. \\ &\left. - \frac{\gamma\left(e^{\alpha t}-1\right)}{\alpha LC_1(\alpha-\lambda_1)(\alpha-\lambda_2)} + \frac{I_1 t}{LC_1\lambda_1\lambda_2}\right] \end{aligned} \tag{4.98}$$

again, valid only for $t < t_0$ and in particular needed only until $V_o(t) = \frac{V_{dd}}{2}$. In addition, the expression for the case when the roots $\lambda_{1,2}$ are complex is presented in appendix A.

Once the analytic expression of the output voltage has been derived, the delay is evaluated as:

$$Delay = t_{50\%}\Big|_{V_o(t_{50\%})=0.5\cdot V_{dd}} \tag{4.99}$$

Note that, the voltage-mode buffer has a full swing from V_{dd} to ground, therefore the delay is computed at $V_{dd}/2$.

The dynamic energy consumption is a function of the switching activity, which is dictated by the communication load divided by the segment width, if we assume one buffer per segment wire. Therefore, the energy is approximated by the expression (4.77).

Further, the leakage energy consumption is estimated by adding the leakage currents of the driver and receiver and by multiplying the result with the idle time:

$$E_s \approx \left(I_{leakage}^{driver} + I_{leakage}^{receiver}\right) V_{dd} t_{idle} \tag{4.100}$$

Unlike in the case of PCM signaling, there is no significant static current flowing through the circuit. Again, the leakage currents are evaluated using the statistical transistor model.

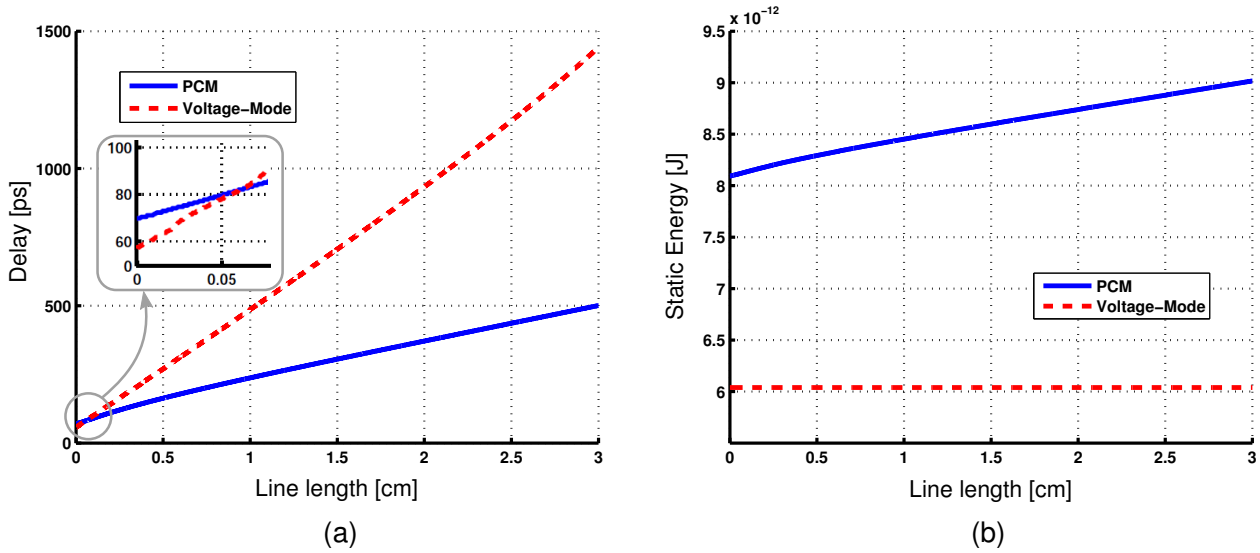

Fig. 4.24: Delay (a) and static energy (b) comparison between voltage-mode and PCM signaling.

4.3.3 Performance Evaluation under Voltage Scaling and Body Biasing

The classic voltage-mode signaling technique exhibits a stronger increase in delay with the interconnect length with respect to the previously-analyzed PCM method. Fig. 4.24(a) shows a direct comparison for interconnect lengths varying between 1 µm and 3 cm. It is also important to notice, that voltage-mode signaling actually achieves a shorter propagation delay for local interconnects. More precisely, up to a line length of approximately 600 µm the voltage-mode signaling achieves smaller delays, which is the direct consequence of the lower complexity of the two-stage buffer with respect to the PCM driver. For lines longer than 600 µm, the propagation time is dominated by the interconnect delay and the current-mode method becomes more efficient.

Fig. 4.24(b) illustrates the static energy difference between the two analyzed signaling methods. As expected, the PCM segment has a higher static energy consumption due to the signaling current. Nevertheless, the difference of up to 3 pJ (for line lengths of 3 cm) is still relatively low compared to the gain achieved in delay. Note also, that the static energy of the voltage-mode line is determined only by the leakage currents flowing through the driver and receiver buffers in idle mode. Thus, interconnect length has no impact on the static energy consumption of the voltage-mode circuit.

In a voltage-mode line the delay is primarily determined by the strength of the driver current, which charges or discharges the line and circuit capacitance. As a direct consequence, through voltage scaling and body biasing the delay can be significantly influenced, as shown by the results from Fig. 4.25(a). Here, the delay increases with approx. 700 ps for a 15 mm interconnect line when the supply voltage is decreased with 0.4 V. In addition, a ±0.4 V body bias varies the delay over a range of 98 ps at the nominal V_{dd}, while the impact increases to a variation range of 755 ps at 0.6V.

Similar to the results from the PCM circuit, body biasing has a much stronger influ-

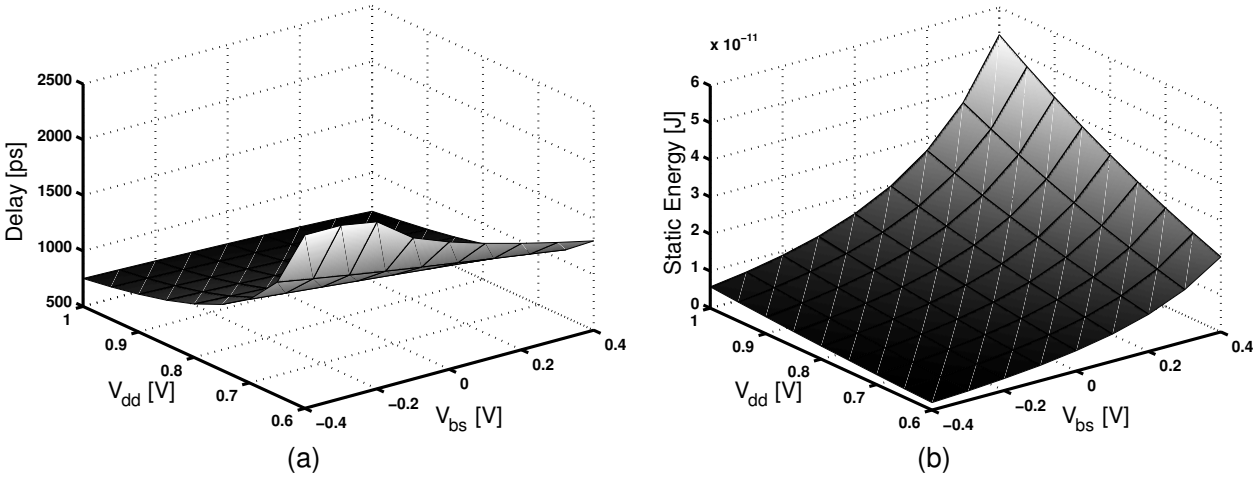

Fig. 4.25: Influence of voltage scaling and body biasing on the delay (a) and static energy consumption (b) of a 15 mm voltage-mode line.

ence on the static energy consumption than voltage scaling. Again, the effect is more significant at higher supply voltages and can vary the energy consumption by a factor of 11.

4.4 Modeling of Communication Segments

As mentioned in Sec. 2.1.1 and shown in Fig. 2.3, this work uses the concept of communication nodes (CNs) to model the on-chip communication. Chapter 3 developed a statistical methodology for representing and propagating the variability across performance macromodels and discussed in detail the representation of processing nodes (PNs). After developing in the current chapter a thorough model for the communication circuits, we are able to present the modeling of communication nodes in the context of the overall delay and energy macromodels.

The information required for building the CNs includes the following:

- Inter-task communication loads, extracted in the profiling step and specified in the input configuration file
- Task-to-resource mapping associations, created during the design exploration steps
- Resource scheduling lists, also generated during the optimization loop

Given the resource mapping configuration, the inter-task communication needs are identified: tasks connected by data dependencies and assigned to different resources require an inter-resource communication segment. Given the communication needs and the resource placement information, the synthesis algorithm allocates communication segments between the resources, using the circuit-level models developed in this chapter.

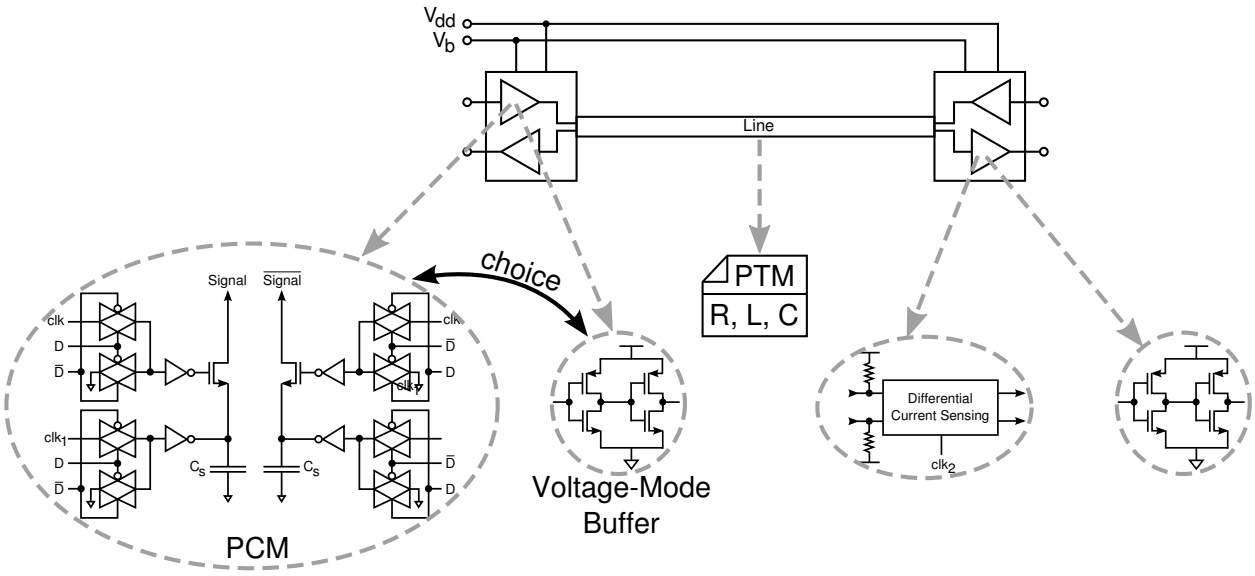

Fig. 4.26: Communication segment using the available circuit-level models.

This section presents the data structures employed to represent the inter-resource communication nodes, shows how the circuit-level models are employed in the CN structures, and discusses the integration of communication nodes into the system-level macromodels.

4.4.1 Transceiver and Interconnect Model

A communication segment is represented within the synthesis framework as an interconnect line model and a transceiver model. To achieve the fast estimations required during the optimization loop, the transceiver uses the analytic models for signaling circuits developed in Sec. 4.2.3 and 4.3.2, while the interconnect line is represented using the R, L, and C analytic expressions of the predictive technology model [116].

In this way, the optimization algorithm can select from the available signaling methods the one which has the better performance for a given segment. In addition, through voltage scaling and body biasing at the circuit level, the CN performance can be further adjusted in fine steps.

Once an optimized communication architecture is found during the synthesis, a precise validation can be performed through simulation, using the complete transistor-level circuits of the driver and receiver, and a more accurate model of the interconnect line. Such a model for the segment lines which is accurate over a wide range of frequencies is developed in chapter 5.

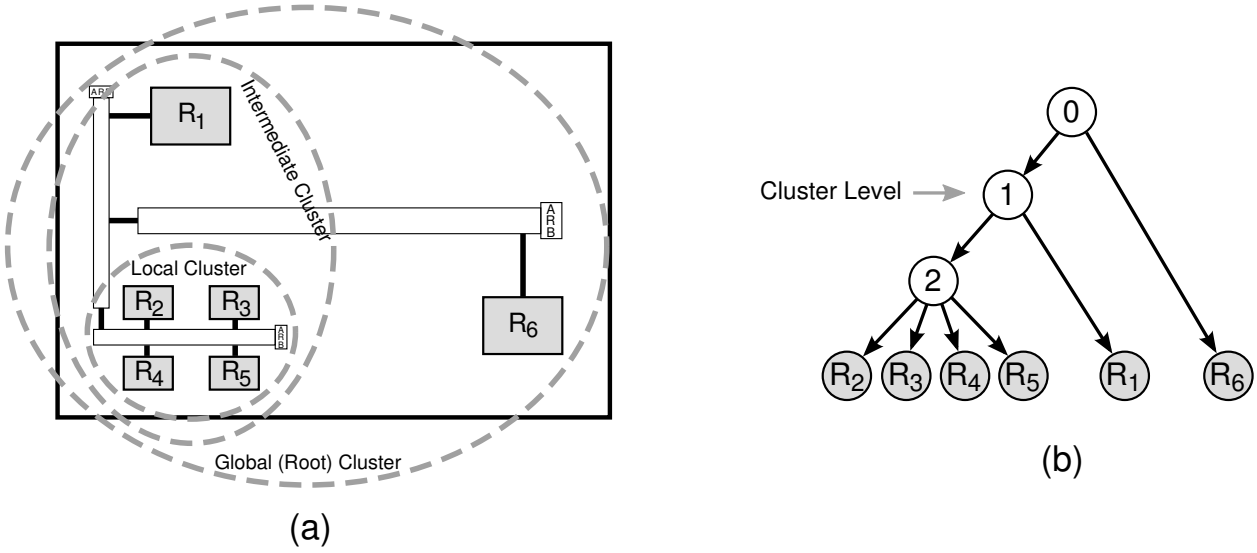

Fig. 4.27: Floorplan clusters enclosing the on-chip resources (a) and the corresponding floorplan cluster tree (FCT).

4.4.2 Floorplan Model using Clusters

Communication parameters, such as segment length, width, and the resulting speed, are strongly dependent on floorplanning and the exact placement of cores. Since the floorplanning and placement steps in the hardware/software co-synthesis are beyond the scope of this thesis, they are abstracted using the following cluster model.

A floorplan cluster encloses the resources placed on the chip within a given distance. This inter-resource distance determines the level of a cluster, such that resources placed closer to each other will belong to local (or lower-level) clusters, whereas resources placed e.g. at the opposite corners of a die will be enclosed by a global (or high-level) cluster.

The concept of floorplan clusters is illustrated in Fig. 4.27(a) for 6 resources placed on the die. Smaller clusters are associated to local communication segment, whereas the top-level cluster encloses resources communicating over global lines. Further, a cluster level is associated to each cluster size and all clusters are organized in a linked data structure called floorplan cluster tree (FCT). The FCT associated to the three clusters is represented in Fig. 4.27(b), where level 0 corresponds to the root cluster and level 2 indicates the local cluster.

Complex MPSoC architectures are described typically using several local and intermediate clusters. Moreover, the number of cluster levels is not limited, therefore the granularity of floorplan clusters can be adjusted. In addition, characteristics such as segment length and width can be associated to each cluster level according to the average distance between the enclosing resources. Thus, a communication model between two given resources can quickly estimate the segment length and width by searching their first common parent in the floorplan cluster tree. The lowest common parent node in the FCT represents the level of the smallest cluster which encloses the two resources, hence

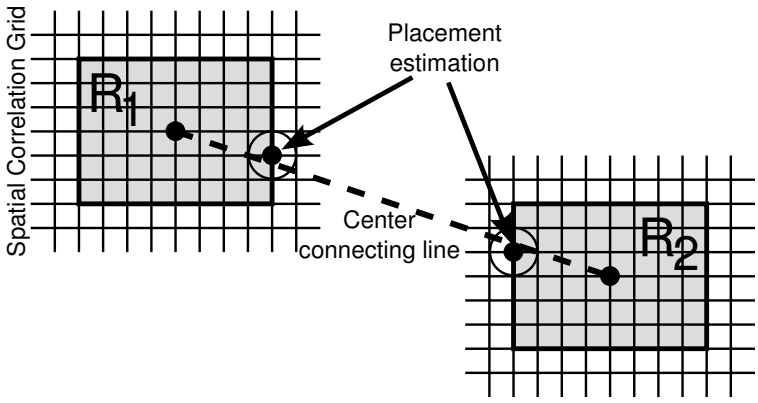

Fig. 4.28: Estimation of communication circuit location for considering spatially-correlated parameter variations.

the length and width of the shortest communication path are employed in performance estimations.

Another important aspect to be considered when using the FCT model is that a first routing estimation must be known together with the floorplan information. This is particularly important, on the one hand, for custom hierarchical bus architectures, where the main directions of the buses must be specified. On the other hand, for network-on-chip (NoC) architectures, the communication distance between cores can be estimated, due to the regular structure of a NoC. In this context, the routing algorithm determines the traveling distance, therefore the length estimation is strictly coupled with the routing mechanism. Resource clustering helps in this case, giving a first coarse estimation if an exact knowledge of the routing algorithm is not available during the synthesis.

4.4.3 Estimation of Communication Circuit Placement on Die

The position of the driver and receiver circuits on the die must be known to consider spatially-correlated process and temperature variations. Nevertheless, the exact location of communication segments is not known during the synthesis loop, as an optimized communication architecture is searched at each step in the solution space. Thus, an estimation of the communication circuit position is performed as illustrated in Fig. 4.28.

Given the floorplan specification for the processing resources R_i, the center position of each resource is determined in coordinates of the correlation grid. After that, the distance between the centers of the two communicating resources is evaluated from the center coordinates and the positions of the communication circuits are evaluated at the intersection between the line connecting the centers of the two resources with the resource boundaries. In the example from Fig. 4.28, assuming resource R_1 of size $w_1 \times h_1$ with center coordinates (x_1, y_1) and resource R_2 with size $w_2 \times h_2$ and center coordinates (x_2, y_2) where d is the center-to-center distance, the coordinates of the driver circuit at the R_1 side

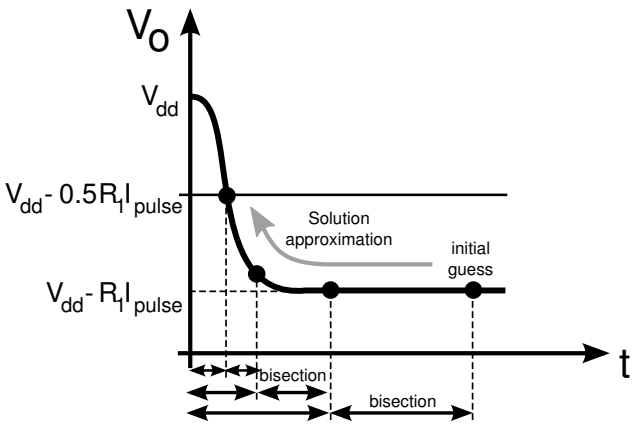

Fig. 4.29: Fast approximation of the delay solution using the bisection method (example shown for the PCM signaling circuit).

are evaluated as:

$$x_{d,1} = \begin{cases} x_1 + (x_2 - x_1)\,\dfrac{h_1}{2\,(y_2 - y_1)}, & \text{if } \dfrac{|x_2 - x_1|}{w_1} < \dfrac{d}{\sqrt{w_1^2 + h_1^2}} \\ x_1 + \dfrac{w_1}{2}\text{sign}\,(x_2 - x_1)\,, & \text{otherwise} \end{cases} \tag{4.101}$$

$$y_{d,1} = \begin{cases} y_1 + (y_2 - y_1)\,\dfrac{w_1}{2\,(x_2 - x_1)}, & \text{if } \dfrac{|y_2 - y_1|}{h_1} < \dfrac{d}{\sqrt{w_1^2 + h_1^2}} \\ y_1 + \dfrac{h_1}{2}\text{sign}\,(y_2 - y_1)\,, & \text{otherwise} \end{cases} \tag{4.102}$$

$$\tag{4.103}$$

The coordinates of the communication circuit at R_2 side are estimated similarly.

4.4.4 Quick Delay Solution

The delay of communication segments is evaluated using the analytic models of signaling circuits. It is to be noticed, that solving the equations (4.76) and (4.99) involves finding the solution of transcendental equations. Thus, this work adopts a fast numerical solving of (4.76) and (4.99) using the bisection method [38]. First, the middle of the interval $[0, t_{initial}]$ is checked, with an initial guess $t_{initial}$ large enough to cover all possible delay values. After checking the middle, the time interval is halved and the method continues as shown in Fig. 4.29 until the approximation interval is smaller than an adjustable ε. Then, the considered solution is the middle of this approximation interval, which gives an approximation error of maximum $\frac{\varepsilon}{2}$.

4.4.5 Implementation of Communication Nodes

Starting from the formulations developed in Sec. 4.2 and 4.3, a statistical model for communication circuits has been implemented, having the structure shown in Fig. 4.30. The

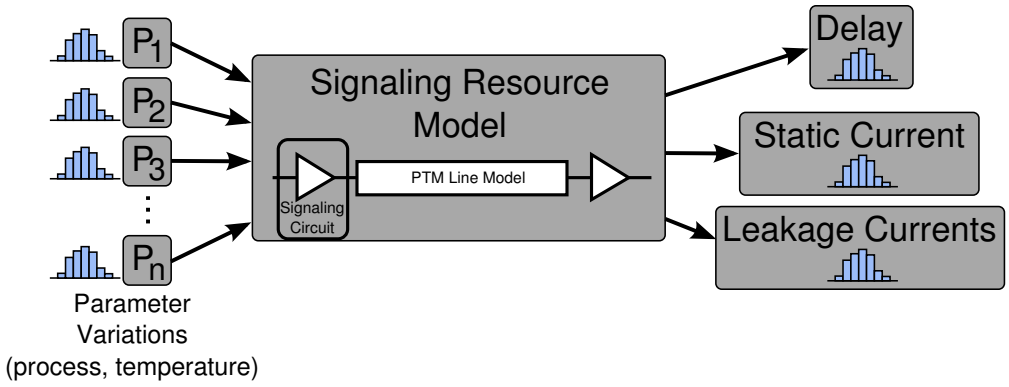

Fig. 4.30: Statistical model for signaling resources embedding the analytical formulations from Sec. 4.2 and 4.3.

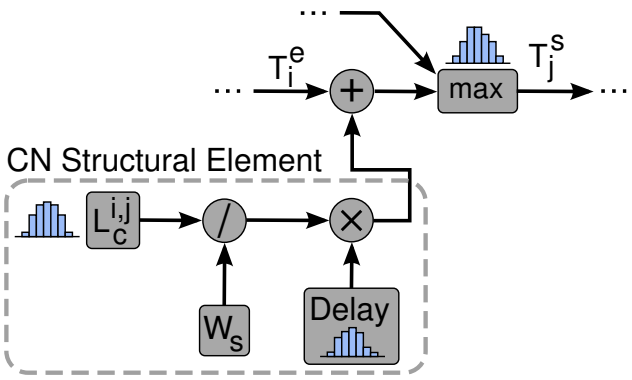

Fig. 4.31: Implementation of a communication node in the delay macromodel.

structure embeds the models of signaling circuits, as well as the interconnect line model attached to the segment. The wire attributes are read from the first common floorplan cluster of the connecting resources and the spatially-dependent process and temperature variations are read from the variations and correlations models described in Sec. 4.1.2. Additional inputs are the choice of signaling circuit, as well as the V_{dd} and V_{bs} values. According to the models, the statistical distributions for the segment delay, for the static current (in the case of PCM signaling), and for the leakage currents are evaluated and available as outputs. In addition, the total segment capacitance is also available for dynamic energy evaluations.

The delay of communication nodes is included in the overall system macromodel as shown in Fig. 4.31. Here, the total delay of the communication across a given segment is computed as the communication load $L_c^{i,j}$ (in bits) between two communicating tasks PN_i and PN_j multiplied by the delay of transmitting a single bit over the line. For communication segments with multiple parallel wires, the communication load is first divided by the segment width W_s. The communication delay computed in this way is then inserted between the end of the execution of PN_i and the start of the execution of PN_j.

Whenever a change occurs in the communication circuit, such as a change in V_{dd} or V_{bs}, or the change of signaling method, the delay random variable must be reevaluated.

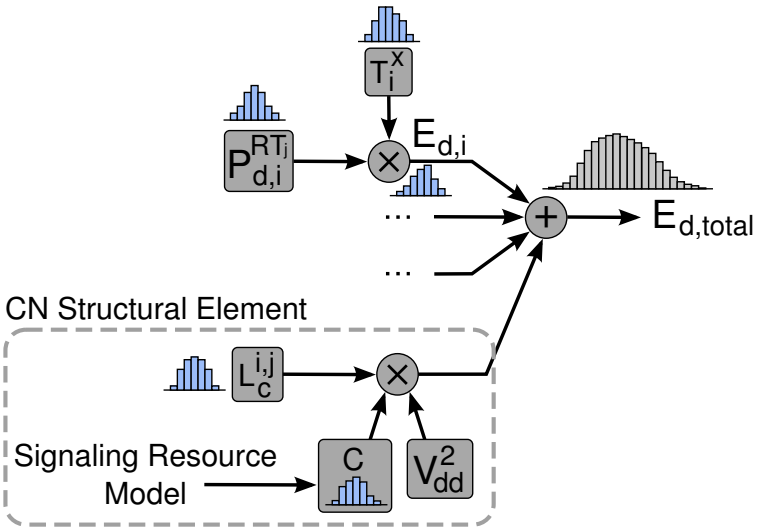

Fig. 4.32: Inclusion of communication nodes in the dynamic energy macromodel.

As a consequence, all the downstream PM nodes starting with the multiplication node shown in Fig. 4.31 must reset their evaluation and update the stored pdf to reflect the change. Further, during task re-mappings to different resources, the structural elements for the newly-required CNs must be built and inserted into the macromodel, while the CNs between the tasks which do not need inter-resource communications anymore due to the different mappings must be removed from the macromodel. Adding and removing CN structural elements requires also the propagation of updates downstream in the delay macromodel.

In the case of dynamic energy, a CN is included in the macromodel as illustrated in Fig. 4.32. Note, that the total capacitance of the segment is computed using the signaling resource model as the sum between the line capacitance, the output capacitance of the driver, and the input capacitance of the receiver. Therefore, whenever changing the signaling circuit during the optimization loop, the distribution of C is updated. This change requires an update of the evaluation for the product operator in the CN structural element and for the output sum operator of the dynamic energy macromodel. Nevertheless, a change in V_{dd} for the current segment (during voltage scaling) triggers a similar update.

Finally, communication nodes are represented within the leakage energy macromodel by structural elements similar to the one shown in Fig. 4.33. The upper branch of the structural element evaluates the leakage energy of the driver and receiver circuits using the leakage currents computed using the transistor-level model. The lower branch is only present in the structural elements representing PCM signaling circuits and evaluates the static energy consumption due to the signaling current pulses. Again, changes during the optimization loop, such as choice of circuit, voltage scaling, and body biasing require the update of statistical evaluations in the corresponding structural element and in the output node of the macromodel.

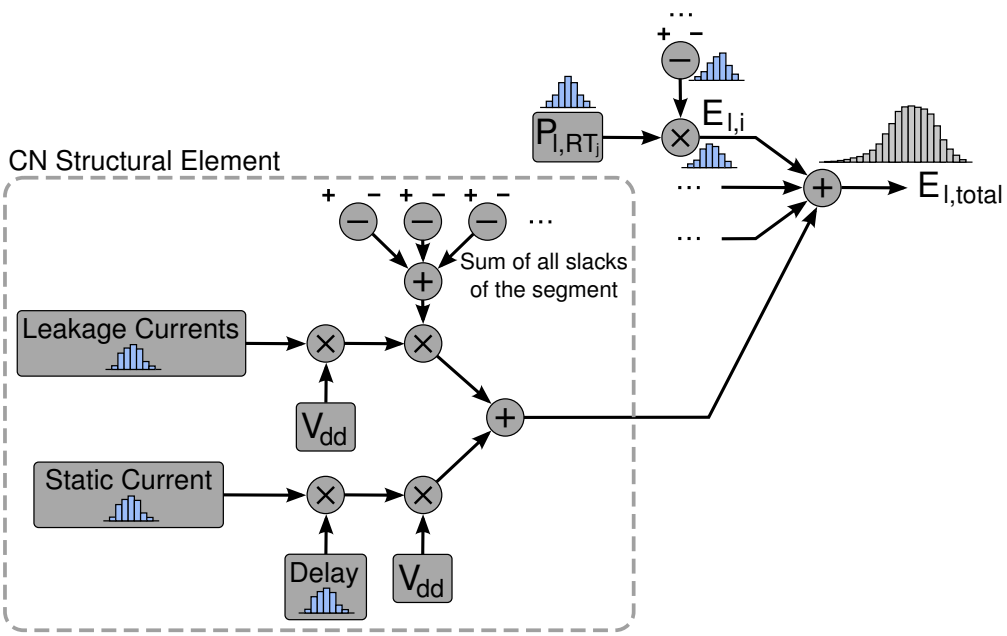

Fig. 4.33: Structural element for modeling the static energy of a communication node within the leakage energy macromodel (example shown for a PCM signaling circuit).

4.4.6 Performance Results

A three-task example with an emphasis on inter-resource communication is used to test the performance characteristics of the developed circuit-level models. Fig. 4.34(a) shows the task dependencies and resource mappings for the considered example. Two communication nodes implement the data transfers between the resources and are synthesized as two communication segments as shown in Fig. 4.34(c). It is further assumed, that resources R_3 and R_1 are placed close to each-other and thus will be connected by a short local segment, whereas R_2 is located farther away on the chip and will be connected using a global segment. The floorplanning information is stored in the two-level cluster tree depicted in Fig. 4.34(b), where the root cluster FC_0 corresponds to the long global segment connecting R_3 with R_2 and the local cluster FC_1 covers the distance between R_3 and R_1.

Tab. 4.2 shows the values used for this example. In order to emphasize the performance costs of communication activities, the execution times and power dissipation levels of the processing resources have been assumed to be relatively small. The opposite holds for the communication loads, which have been chosen to be large enough such that the total performance cost is strongly influenced by the communication synthesis. Further, variations in the communication loads, execution times, and power dissipations have been specified using normal distributions $N(\mu, \sigma)$. The communication in the local cluster is synthesized using a segment of length $L = 50\,\mu\text{m}$, for which the voltage-mode (VM) signaling achieves the smallest delay. In contrast, the root cluster requires a segment of length $L = 5\,\text{mm}$, which lies in the region where PCM signaling performs better.

The statistical evaluations in the macromodel were performed on discretized pdfs

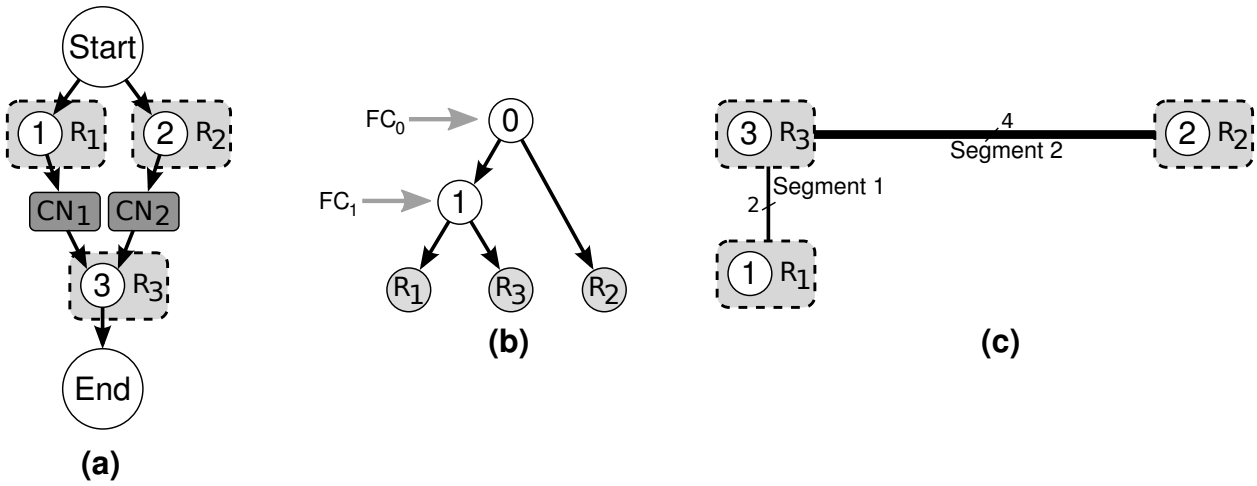

Fig. 4.34: Task graph example with emphasis on communication nodes (a), the associated floorplan cluster tree (b), and the modeled communication segments (c).

$L_c^{1,3}$	$N\,(100, 2)$ [MB]
$L_c^{2,3}$	$N\,(50, 2)$ [MB]
P_{l,RT_i}	$N\,(5, 1)$ [µW]
T^x_{i,RT_j}	$N\,(20, 2)$ [ps]
$P^{RT_j}_{d,i}$	$N\,(5, 1)$ [µW]
FC_0	$L = 5$ mm, $W_s = 4$
FC_1	$L = 50$ µm, $W_s = 2$

Tab. 4.2: Input values for the communication synthesis of the three-task example.

with N_b = 30 bins and a number of N_{sb} = 500 individual samples from a total sample count N_s = 10 000 for the numerical operators. The relevant sigma-domain for the statistical distributions has been set to $\pm 3\sigma$ and a 10×10 correlation grid was employed for parameter variations.

First, the delay of the synthesized architecture is analyzed, with respect to the different signaling choices. The results plotted in Fig. 4.35 show the delay pdfs evaluated using the delay macromodel for different signaling circuits on the two communication segments.

Using the PCM signaling method on the first segment is detrimental for the delay, since the 50 µm length of segment 1 lies below the threshold where PCM signaling starts to achieve smaller delays, as discussed in Sec. 4.3.3 and shown by Fig. 4.24(a). In addition, the high communication load $L_c^{1,3}$ amplifies the delay difference between the two signaling methods. Thus, the two configurations with PCM signaling on segment 1 achieve the worst delays, as shown by the plots in Fig. 4.35. Using voltage-mode signaling on the first segment reduces the delay mean with at least 5 ms in this example.

Further, the second segment with a large length is more appropriate for PCM signaling. Thus, the use of PCM signaling on the second segment reduces the delay on average with 6 ms and represents the optimum configuration for the given example.

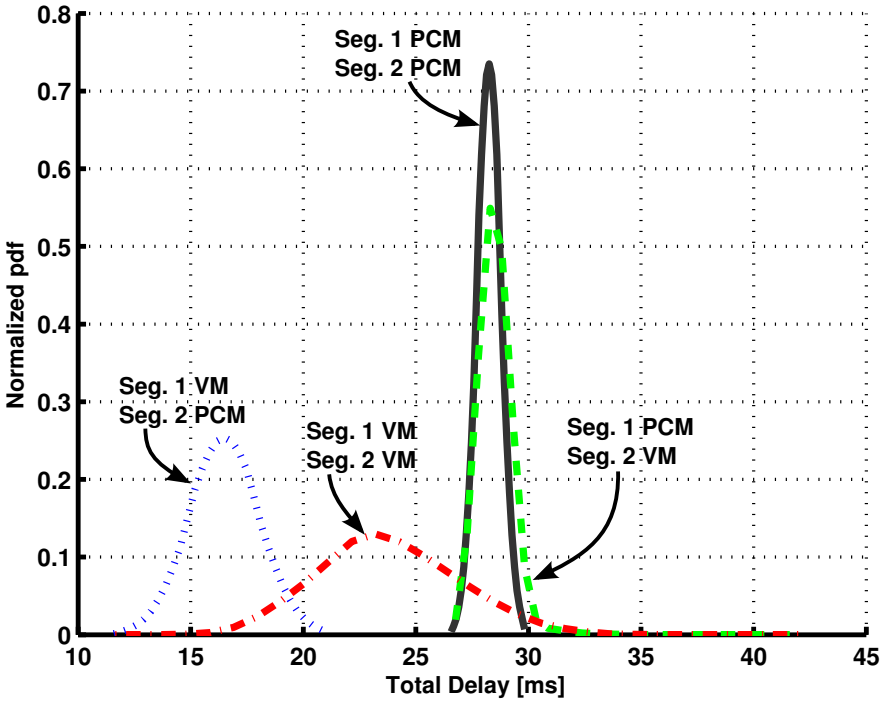

Fig. 4.35: Total delay of the synthesized structure from Fig. 4.34(c) with different signaling circuits on the two communication segments.

The impact of body biasing on the leakage power distribution is investigated in Fig. 4.36 for the minimum-delay configuration identified from Fig. 4.35. Here, the sweep of V_{bs} from -0.4 V to +0.4 V increases the leakage energy by a factor of 1.5. As discussed in Sec. 4.2.4 and 4.3.3, the leakage energy of both signaling methods is significantly influenced by body biasing.

To conclude, the modeled signaling circuits are optimally employed in different cases. For very short communication segments, voltage-mode signaling achieves the lowest delay, whereas PCM signaling has an advantage in longer communication lines. Voltage scaling has mainly an influence on the delay of voltage-mode circuits and less of an impact on the PCM lines (only the delay of the driver is affected, which represents a small fraction from the total delay of the line). Body biasing has also a negligible impact on the delay of PCM lines, while showing an influence on voltage-mode circuits (especially at lower V_{dd} values). In contrast, leakage energy is strongly dependent on the body bias, while the dynamic energy is quadratically influenced by voltage scaling.

4.5 Summary

Different signaling methods and circuit-level techniques can be used in on-chip communication to optimize the delay or energy consumption. For instance, pulsed current-mode (PCM) signaling achieves a low delay over long interconnect lines, whereas conventional voltage-mode signaling has a low static energy consumption. To embed the signaling cir-

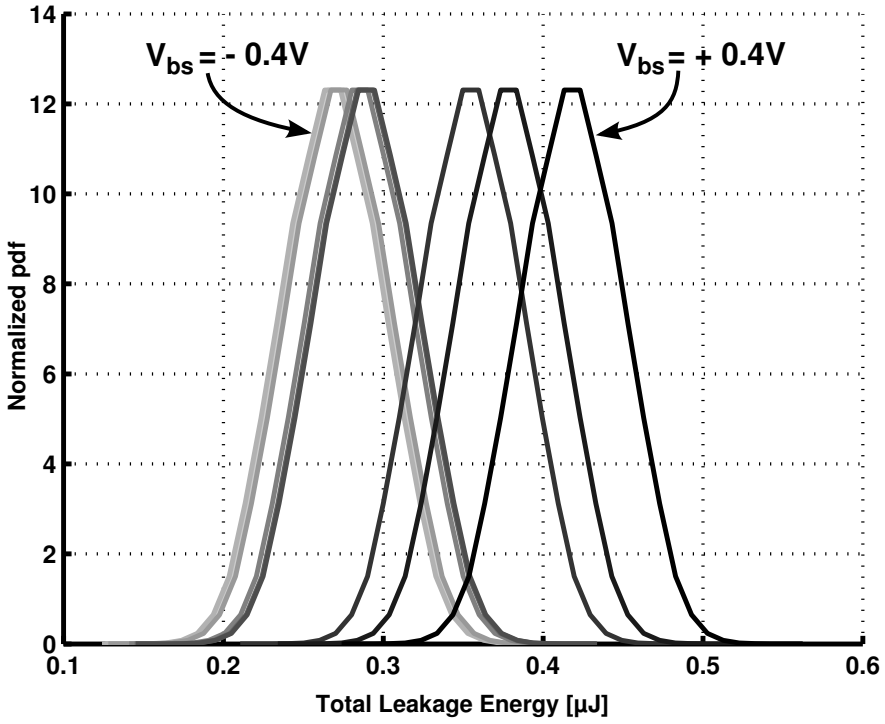

Fig. 4.36: Influence of body biasing on the leakage energy of the configuration with voltage-mode signaling on segment 1 and PCM signaling on segment 2.

cuits in the communication synthesis flow, a set of analytic models have been developed in this chapter starting from process-dependent parameters and circuit-level analysis.

First, a technology-dependent statistical transistor model has been derived from the BSIM4 model equations. The developed current-source model supports variability descriptions for all process-dependent parameters and employs the statistical operators developed in the previous chapter to propagate the parameter distributions throughout the model expressions. Spatially-correlated intra-die parameter variations are described using a two-dimensional correlation grid and a correlation decay model. Within this context, the use of principal component analysis (PCA) for the variable decomposition has also been presented.

Furthermore, the PCM and voltage-mode signaling circuits have been analyzed and modeled using equivalent circuit representations. Using the current-source transistor model, an analytic expression has been derived for the voltage at the receiver side and the delay has been computed at 50% voltage swing. In addition, static and dynamic energy models have been derived for the signaling circuits based on the current values and on the parasitic capacitances. All model evaluations are performed statistically using the underlying current-source transistor model which is dependent on process and environmental variations. The impact of voltage scaling and body biasing on the circuit performance has also been analyzed.

Finally, the circuit-level statistical models have been employed for modeling entire communication segments. The segment parameters, such as length and bitwidth are de-

rived from a floorplan model using clusters, while the location of transceiver circuits is approximated from the placement of processing resources. The inclusion of segment models into the system-level performance macromodels employed in the optimization has also been presented and the particularities of updates in the signaling method, voltage scaling, and body biasing have been discussed. The use of circuit-level models for communication segments has been demonstrated by means of a two-segment example, where the delay-optimal signaling configuration and the influence of body biasing on leakage energy consumption have been discussed.

Chapter 5

Technology-Aware Characterization Method for On-Chip Segments

Contents

Since the precision of the developed communication macromodels is directly determined by the estimation accuracy for propagation delay and energy consumption across the lines, a further refinement of the interconnect model is welcome. Particularly high-frequency quality drops such as dielectric losses, reflections, and crosstalk are becoming important at high switching speeds and require more complex interconnect models. Therefore, this chapter presents a novel wide-bandwidth interconnect modeling technique, which takes into account the complex stacked dielectric environments and tightly-coupled metal layers present in the state-of-the-art CMOS processes. Additionally, the presented method is applied for a 90-nm technology and the results are compared with a full-wave field simulator.

Sec. 5.1 enumerates briefly the main challenges of interconnect modeling and introduces our extrapolated S-parameter model. Further, Sec. 5.2 describes the simulation

framework employed for extracting the base parameter set used within the model. Next, the parameter extrapolation method is detailed in Sec. 5.3. Finally, the experimental evaluations are presented and discussed in Sec. 5.4.

5.1 Wideband Characterization Method

At gigahertz frequencies, bus data and clock signals in integrated circuits are entering the microwave-specific range and the global on-chip interconnects become a more and more critical bottleneck in the global system performance [48, 103]. Moreover, numerous vias, crossing lines, and dielectric discontinuities, as well as a high wire packing density, are common attributes of state-of-the-art CMOS processes, but constitute nevertheless a frequent cause of crosstalk and reflections [54]. In addition, important signal quality drops generated through skin effects and dielectric losses augment with the frequency and cannot be ignored anymore in the present interconnect wires. In this respect, an increasingly high percentage of the final circuit performance becomes dictated by the interconnects [48], although devices and, recently, device parameter variability continue to influence a significant performance amount. Thus, with increasingly high integration scales, the electrical performance of interconnects must be accurately characterized, modeled, and seamlessly integrated into IC design flows.

There are two common approaches in the analysis, measurement, and description of interconnect structures, namely in the time domain (using e.g. time-domain reflectometry or eye diagrams) and in the frequency domain (employing e.g. S-parameters) [103]. Although the two approaches may embed the same information in different forms, for practical circuit modeling purposes there are other factors, such as simulator support or the amount of computational overhead vs. accuracy, which decide which method becomes more appropriate. Furthermore, the analysis of on-chip interconnects is significantly burdened by challenging factors like high losses, scaled aspect ratios, increased number of wires, and strong non-uniformities in the dielectric stack [86], which contribute to the difficulty of employing standard measurement and simulation techniques.

In order to enable the accurate characterization of such parasitic effects within a practical workflow, designers need efficient performance estimation models at lower abstraction levels, which are capable to describe arbitrary interconnect structures [54] and are developed to support an integration with industry standard simulation frameworks. For instance, signal integrity analyses in digital communication structures operating at gigahertz frequencies [78] require the interconnect models to be valid over very wide frequency ranges.

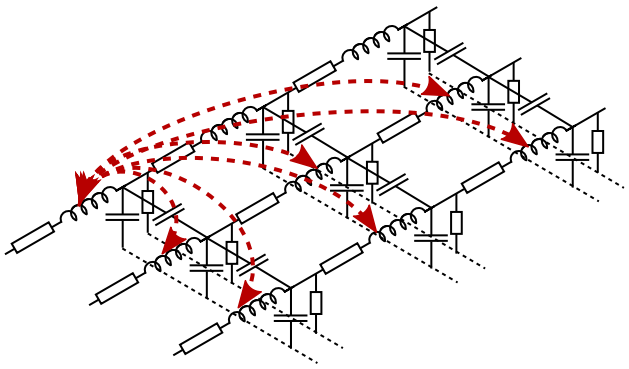

Fig. 5.1: Complexity of mutually-coupled inductances in distributed RLCG models.

5.1.1 Interconnect Modeling Challenges

The parasitic effects which affect the densely-packed interconnect wires, such as dielectric and substrate-induced dispersion, skin effects, and proximity effects, are strongly dependent on frequency [137] and need to be carefully considered by the interconnect models. Further, the evaluation of self and mutual impedances requires finding the current return paths for each individual wire, which are also frequency-dependent [109]. In addition, the actual return paths are difficult to estimate in interconnects, since there is no ground plane between the metal layers. Finally, the rapidly-switching signals exhibit very narrow rise and fall times and therefore contain significant spectral components within wide frequency ranges.

Interconnect models have traditionally evolved from simple, lumped capacitance, through lumped and distributed RC, until the state-of-the-art transmission-line distributed RLCG chains. Lumped and distributed RC models neglect inductive effects and fail to model lossy interconnect lines with propagation delays comparable or larger than the signal rise time [111]. Inductively and capacitively coupled, distributed RLCG models are today generally preferred [91, 137, 53], as they provide a good tradeoff between accuracy and model complexity. Within a distributed RLCG representation, the wires are divided into cascaded segments of circuit elements, extracted to reflect the interconnect response up to the desired significant frequency. These models are designed to enclose high-frequency effects, they achieve a perfect circuit-level compatibility with simulators, and are fast to simulate. On the other hand, they rely on coupled mutual inductances between all segments from all chains, as illustrated in Fig. 5.1 for only one inductance of a single chain segment. As a consequence, the modeling complexity increases exponentially with the number of cascaded segments and it becomes extremely hard to compute the value of each mutually coupled inductance pair.

Field solvers are commonly employed for computing accurate capacitance values [112], and for resistance and inductance [88] extractions. Nevertheless, for fast estimations of arbitrary interconnect structures needed in the early design stages, analytic expressions have been developed for capacitance [166] and self inductance [145] computation. On

the other hand, mutual inductance values can only be estimated for two parallel running lines of equal wire length [123], hence they are restricted to a small number of cases. Field solver extractions of wire parameters, while being very accurate, are not applicable for real-time estimations, due to the time overhead and complex structural setup they imply. Moreover, the extracted models exhibit a decreasing accuracy with frequency increase and their maximum frequency of validity is specified in terms of the acceptable error [111]. In addition, distributed RLCG models rely on the quasi-TEM (transverse electric and magnetic) propagation of electromagnetic waves in transmission lines [53], which does not account for radiation losses and steep discontinuities (vertical vias, wire segment bends) [137] and assumes that the cross-sectional wire dimensions are much smaller than the wavelength at the maximum frequency of interest. In such cases, more precise electromagnetic analyses might be required.

Full-wave characterization methods [137] are ideal for accurate wide frequency-range modeling purposes, as they rely on a direct discretization of Maxwell's equations and find a numerical solution at every frequency. Such methods include differential-based approaches, such as either frequency-domain finite element solvers [76] or time-domain finite difference solvers [74], and integral-based techniques, such as the method of moments [114] and the partial element equivalent circuits (PEEC) [138, 137]. The complexity implied by the discretization and numerical solving of Maxwell's equations in differential or integral form is requiring however a substantial computational overhead [137]. Hence, while holding the highest accuracy, these methods are not directly applicable for interconnect synthesis applications, unless a method exists to extract fast characterizations of arbitrary interconnect segments.

5.1.2 Multistep Extrapolated S-Parameter Model

As discussed in Sec. 5.1.1, there are two main directions in the high-frequency interconnect modeling: first, the full-wave numerical methods, with high accuracy but restricted by their large computation time, and second, the transmission-line distributed RLCG circuit models, with a good accuracy over the frequency range, but with a very complex tracking of the mutual inductive couplings between the distributed segments. In contrast, a third approach with limited complexity is proposed here, which exhibits a modeling performance close to the precision of a field simulator. This method consists of an incremental extrapolation technique for generating a set of S-parameters for an arbitrary interconnect segment in a given CMOS process, which relies on a predefined set of measured parameters obtained either with a vector network analyzer (VNA), or with a field simulator and a structural model of the silicon environment. The resulting model of the interconnect segment is an n-port with its associated S-parameter matrix.

As illustrated in Fig. 5.2, the proposed methodology starts with an initial set of extracted parameters, which samples the entire range of possible interconnect structures, as seen from a designer's point of view, in a predetermined way. This initial set ex-

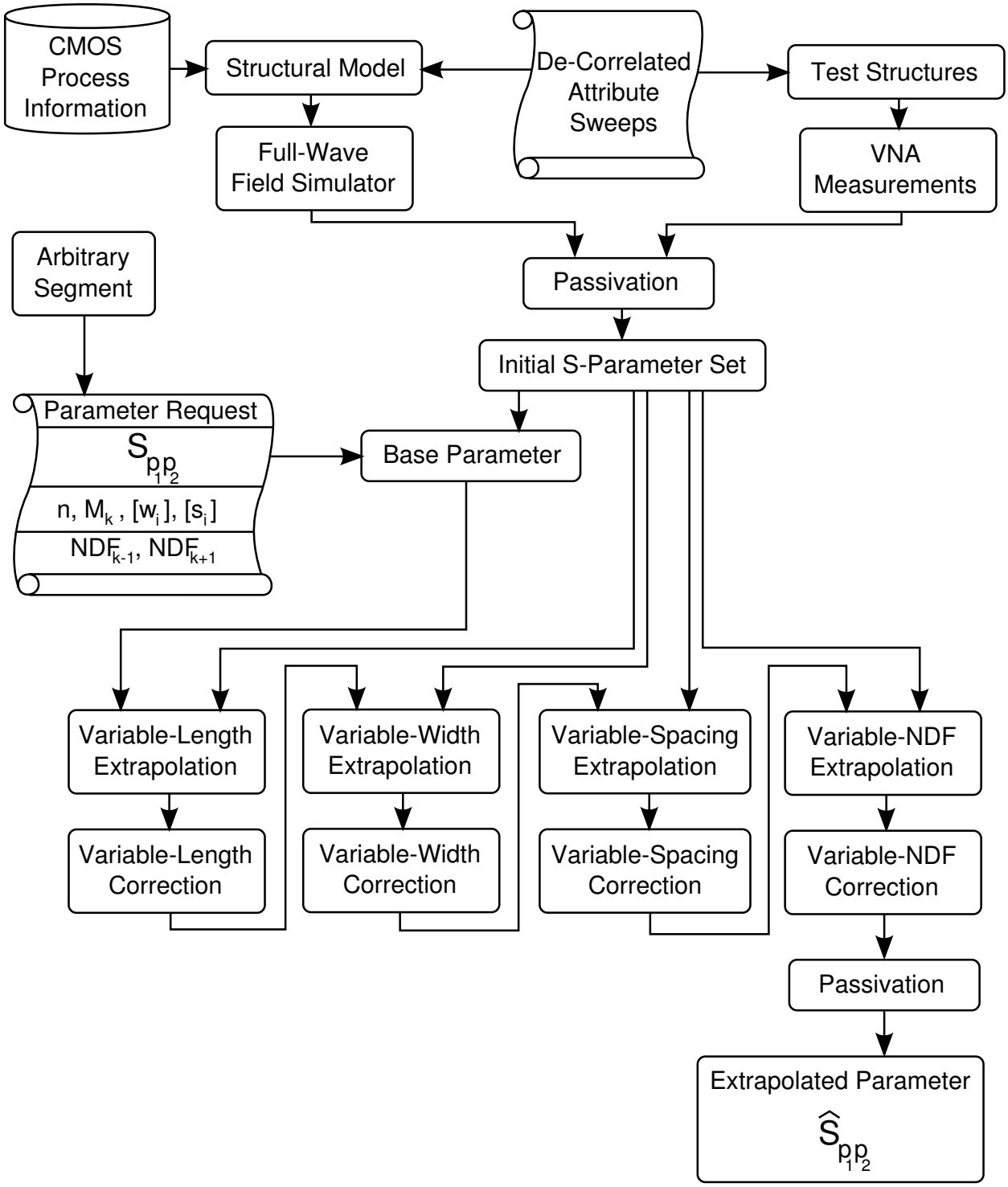

Fig. 5.2: Overview of the extrapolated S-parameter modeling workflow.

plores variations in the metal layer, wire number, wire length, as well as individual wire widths, wire spacings, and neighboring routing configurations in the adjacent metal layers. Within this process, the parameters are extracted for a wide frequency range which extends up to the bandwidth required by the target application. Furthermore, the extracted set can be obtained either from direct measurements on a test chip, or using an

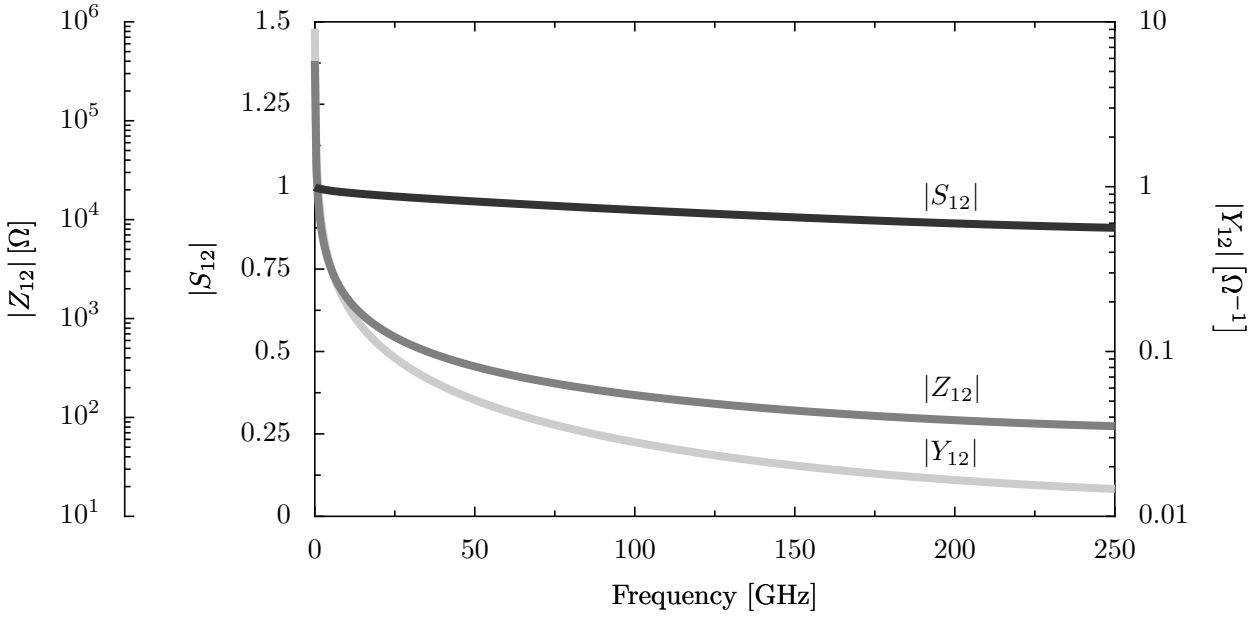

Fig. 5.3: Magnitude plot of the Z_{12}, Y_{12}, and S_{12} parameters for a single-wire segment.

accurate field simulator and a multi-layered representation of the substrate, metal, and dielectric environment for the target technology. In both cases, measurement or computational errors are likely to affect the parameter values, threatening the stability of the interconnect model. Thus, a passivity enforcement criterion is employed in the modeling flow, requiring the real part of the admittance-parameter matrix to be positive definite [158].

The initial set of parameters is then used as basis for an incremental suite of extrapolations, directed on the individual wire attributes, such as length, width, spacing, metal layer, and neighboring routing information. The individual extrapolation for each wire attribute is possible since the initial extraction of the base parameters is designed to minimize the correlation between the attributes. Furthermore, the inclusion of common design practices, such as orthogonal routing in neighboring metal layers and shielding of bus segments with V_{DD} and ground (GND), as well as the layout design rules for a specified process, limit the complexity of the initial extraction procedure to a polynomial $O(N^2)$ for a maximum of N minimum-width wires between the power grid shielding lines.

Several n-port parameter sets are available for representing the frequency behavior of interconnect segments, including impedance (Z), admittance (Y), and scattering (S) parameters. In Fig. 5.3 the magnitudes of a Z, Y, and S parameter for a 100-µm wire in a 90 nm technology (metal 5) have been plotted. The characteristics show that the Z and Y parameters vary across several orders of magnitude within the considered frequency range, whereas the S parameter value remains between 0.8 and 1. If we consider the extrapolation of parameters across the entire frequency range, then the amount of variation of each parameter within this range will directly affect the extrapolation performance. As

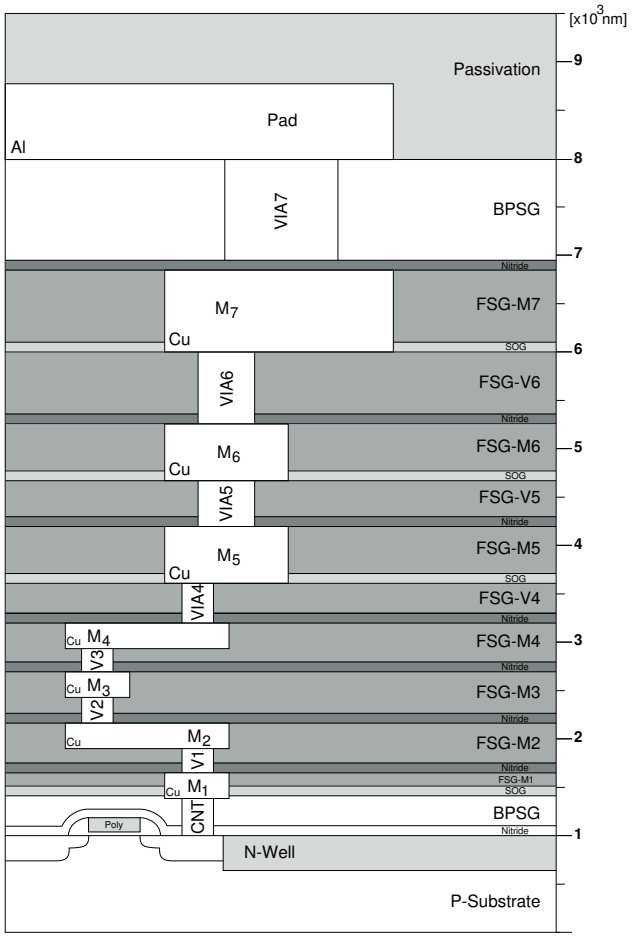

Fig. 5.4: Cross-section through the structural model of the CMOS process.

a result of this observation, the S-parameter representation has been chosen for the modeling methodology, since it exhibits the least amount of variation across the frequencies of interest.

5.2 Parameter Extraction Framework

The incremental extrapolation method relies on an initial set of S-parameters, which can be obtained either from direct VNA measurements, or with a field solver. An industry-standard 3D full-wave finite element method-based field simulator [15] has been used in this work to extract the base parameters from an interconnect structural model representing the target technology. Fig. 5.4 depicts a cross-section through the simulated 7-metal-layer (4-2-1) structure for the 90-nm, 1.0-V digital CMOS process employed within this work. The structure includes a total of 7 copper metal layers within a fluorosilicate glass (FSG) dielectric, separated by spin-on-glass (SOG) etch stop layers, silicon nitride dielectric diffusion barriers, and borophosphosilicate glass (BPSG), and finally bounded by a p-type doped silicon substrate at the bottom and an aluminum-pad grid on the top. This

stacked material structure models effectively the complex dielectric environment and the inter-wire couplings present in tightly-integrated CMOS digital circuits.

For each metal layer, all the wire structures required for the subsequent extrapolations have been simulated across a frequency range from DC up to the maximum significant frequency. Hereby, the maximum frequency for employing the model in SPICE simulations has been selected as:

$$F_{max} = F_{knee} \cdot N_{steps} \approx \frac{0.5}{t_{rise_{min}}} \cdot 5 \tag{5.1}$$

where F_{knee} denotes the knee frequency and N_{steps} is the number of required time steps per rise time (t_{rise}) during a SPICE transient simulation. A number of 5 steps together with a rounded upper bound for the knee frequency have been chosen for increased frequency validity. As a consequence, for an arbitrary minimum rise time of e.g. 10 ps, a maximum frequency of 250 GHz is obtained.

The extracted S-matrices usually contain generalized parameters, which are normalized to the impedances of each port. Since the port impedance depends on the attached load or driver, it is more practical to have all the parameters normalized to a single known impedance value. For convenience, the results have been normalized to the standard specific impedance of 50 Ω. In many cases, the solution of the solver would consider only the dominant mode. If higher-order modes are present in the structure, they should also be included. In such a case, a multi-mode analysis can be performed and the propagation constant $\gamma = \alpha + j\beta$ can be inspected for each mode. Nevertheless, each additional mode at a port adds an additional set of S-parameters. In this case however, the results show that a multi-mode analysis is not necessary, and the coupled lines can be accurately modeled with one mode per terminal.

5.3 Multistep Extrapolation Method

5.3.1 Extraction of the Base Parameter Set

The target of the developed extrapolation procedure is to compute a requested parameter $S_{p_1p_2}$ from the available set of extracted results, given the following specifications: the requested frequency f_k, the requested ports p_1 and p_2, and the structural details of the interconnect segment, such as the metal layer M_k, the wire length l, the set of wire widths $[w_i]$, the set of wire spacings $[s_i]$, and the neighboring routing configurations. In order to extrapolate the requested parameter values, we must have an initial set of extracted S-parameters for every metal layer $M_1, \ldots, M_N$, a variable number of wires, variable wire lengths, variable wire widths, variable wire spacings, and variable routings in the neighboring metal layers.

To keep the problem tractable, a set of simplifying assumptions must be considered, which actually reflect common best practices in the design of state-of-the-art high-density

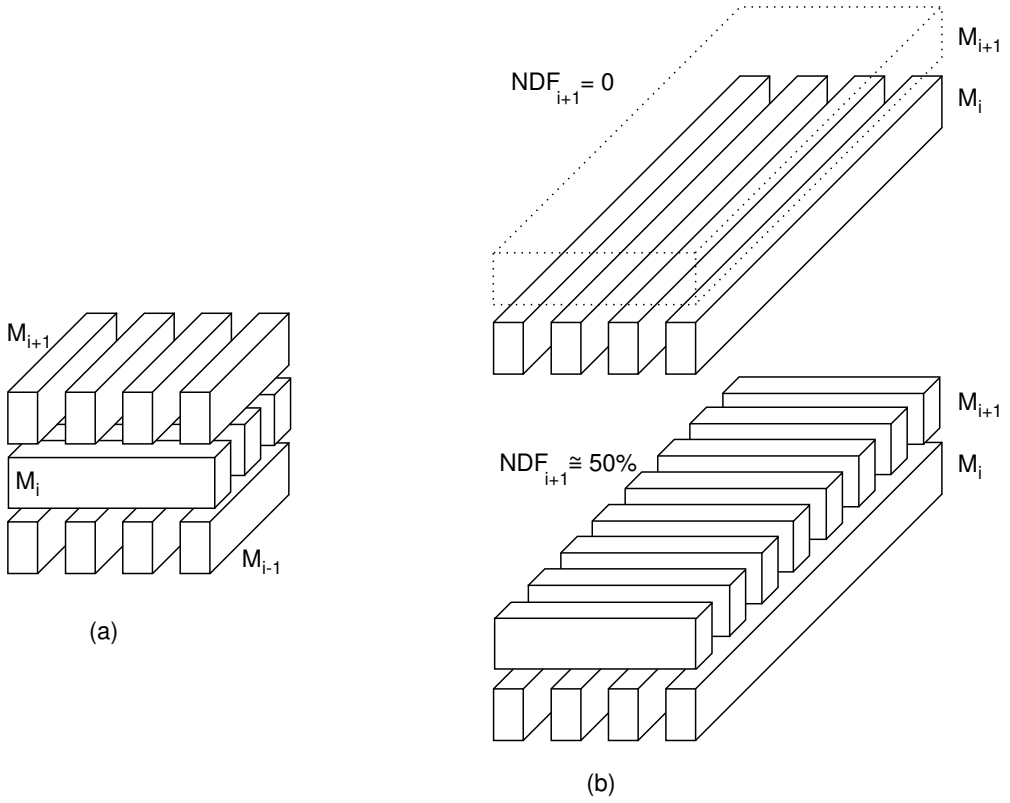

Fig. 5.5: (a) Orthogonal routing directions in adjacent metal layers. (b) NDF values of 0, respectively 50%.

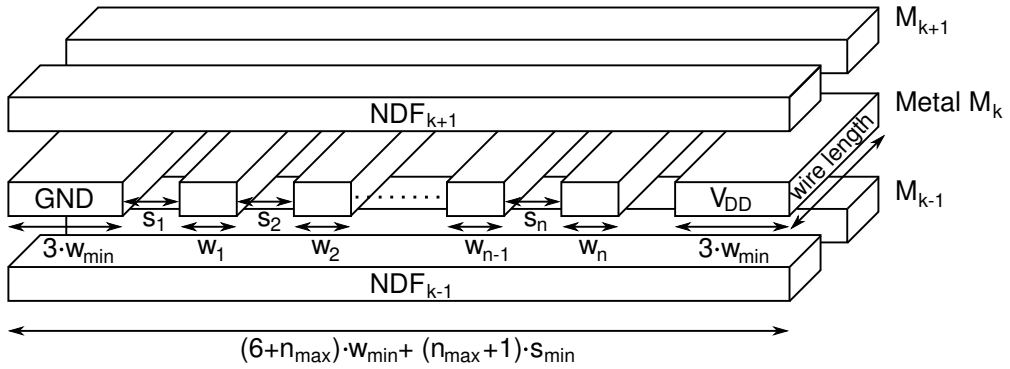

Fig. 5.6: Structural model of an n-wire interconnect segment.

digital signal processors. First, the number of parallel running wires is limited to n_{max} by introducing a power grid consisting of V_{DD} and GND shielding lines, in order to ensure a controlled low-impedance current return path and to limit the inductive-coupling effects [134, 91]. For this purpose, a maximum of six minimum-width signal wires is assumed between every two shielding lines. Finally, it is assumed that the routing in neighboring metal layers occurs only in orthogonal directions, to further minimize the inductive coupling, as shown in Fig. 5.5(a).

The influence of routed wires in the neighboring layers is considered by introducing

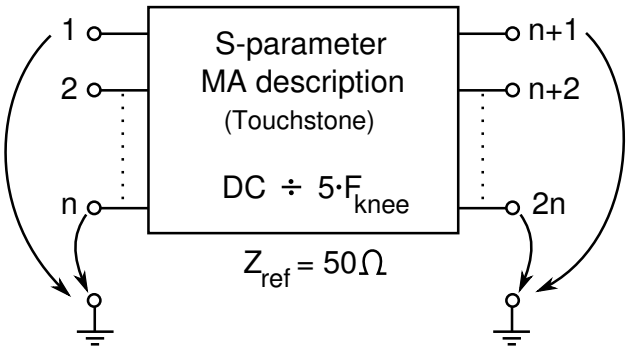

Fig. 5.7: Associated n-port model for an n-wire segment.

a *neighboring density factor* (NDF). Further, the existence of routed wires in the adjacent metal layers is modeled by considering a density factor between 100% (i.e. a metal plane or a very thick wire which covers 100% of the considered segment) and 0% (i.e. no routing in the neighboring layer, as illustrated in Fig. 5.5(b) for e.g. 0 and 50%. Given this definition, an NDF value is considered for each side of the metal, except for the lowest and the highest metal layers, which have only one neighbor. For instance, a segment placed on metal layer M_k has an NDF corresponding to the neighbors in M_{k+1} given by:

$$NDF_{k+1} = \frac{1}{l_k} \sum_{j=1}^{n_{k+1}} w_j \tag{5.2}$$

where n_{k+1} is the number of wires, including shielding lines, which cross the segment, w_j is the width of each wire, and l_k is the length of the segment under consideration.

The structural model of an n-wire interconnect segment routed on a given metal layer M_k is depicted in Fig. 5.6. All wires within the segment have the same length l, but individual widths w_i and spacings s_i. Additionally, interconnects with distinct wire lengths can be modeled by concatenating several n-port segments [53]. The associated n-port model of the segment is shown in Fig. 5.7.

A complete base of initial parameters for the subsequent extrapolation must cover all metal layers and all numbers of wires in a segment, from 1 to n_{max}. As stated before, the number of parallel-running wires is restricted by the presence of a shielding grid, thus the distance between two subsequent GND and V_{DD} lines allows for the routing of maximum n_{max} minimum-width, minimum-spacing wires. Furthermore, the minimum wire width w_{min} and wire spacing s_{min} are dictated by the layout design rules for each metal layer.

Since the model applies an incremental sequence of extrapolations for each individual wire characteristic, the initial extracted set must be chosen in such a way, as to minimize the correlation between wire attributes. More specifically, we can describe the sequence of extrapolations for a requested $S_{p_1p_2}$ as:

$$\hat{S}_{p_1p_2} = \sum_{a_i \in \{a_1, a_2, \dots\}} extrap\left(\hat{S}_i^{p_1p_2}, F_{s_i}\right) \tag{5.3}$$

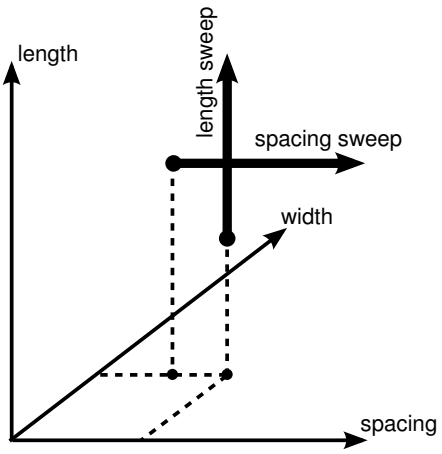

Fig. 5.8: Orthogonal sweeps of the wire attributes, illustrated here for length and spacing (NDF axis not shown).

where $\hat{S}_{p_1p_2}$ is the extrapolated parameter value, $a_1, a_2, \ldots$ are the individual wire attributes, $\hat{S}_i^{p_1p_2}$ is the extrapolated contribution of wire attribute a_i to the final parameter value, and F_{s_i} is the sweep function for wire attribute a_i. In order to apply the extrapolations individually on each attribute and sum the contributions, the sweep functions F_{s_i} must be orthogonal, i.e. they must not introduce correlations between the attributes during the sweeps.

Orthogonality between the attribute sweeps can be achieved by varying only one attribute at a time, while keeping the other attributes constant. In the attribute space, such sweeps would correspond to orthogonal lines, parallel to each of the attribute axes, as exemplified in Fig. 5.8 for wire length and spacing sweeps. An additional orthogonal NDF axis can not be displayed in Fig. 5.8, however it only adds a fourth dimension to the attribute space.

It is to be noted, that the individual wire attributes are not completely independent one from each other. For instance, the assumption of having a fixed power grid introduces a relatively strong dependence between wire width and spacing. Specifically, the width of a wire cannot be changed without affecting also the spacing to its neighbors. This generates a residual correlation between the attribute sweeps which must be taken into account and corrected afterwards. A controlled weighting of the incremental attribute correction steps is performed during the subsequent extrapolations, which are described in Sec. 5.3.2.

Given these observations, the orthogonality of the parameter sweeps is maximized by sweeping each individual wire attribute while keeping the other attributes at a neutral value (i.e. the minimum, or the average value, depending on the attribute). The algorithm applied for the extraction of the base parameter set is given in Listing 5.1. First, the length of the segment is varied across the relevant domain for metal layer M_k, i.e. from $l_{min}\,(M_k)$ to $l_{max}\,(M_k)$, with all the wires set to the minimum width and equally-spaced between the bounding power grid. From this first set of simulations we collect parameter sets which reflect only changes in wire length, while the influence of wire width and

```
EXTRACTINITIALSET()
 1  for each metal layer M_k
 2  do for n ← 1 to n_max
 3     do /* Length Sweep */
 4        NDF_{k-1} ← NDF_{k+1} ← 0;
 5        for i ← 1 to n
 6        do w_i ← w_min;
 7           s_i ← ((n_max − n)/(n+1)) w_min + ((n_max + 1)/(n+1)) s_min;
 8        for l ← l_min (M_k) to l_max (M_k)
 9        do EXTRACT-S-PARAMETERS();
10
11        /* Width Sweep */
12        l ← l_mean (M_k);
13        for i ← 1 to n
14        do for w_sweep ← w_min to (n_max − n)(w_min + s_min) + w_min
15           do w_i ← w_sweep;
16              s_i ← ((n_max − n + 1)(w_min + s_min) + s_min − w_sweep)/2;
17              for j ← 1 to n, j ≠ i
18              do w_j ← w_min;
19                 s_j ← s_min;
20              if i < n
21                then s_{i+1} ← ((n_max − n + 1)(w_min + s_min) + s_min − w_sweep)/2;
22              EXTRACT-S-PARAMETERS();
23
24        /* Spacing Sweep */
25        for i ← 1 to n
26        do w_i = w_min;
27        for i ← n downto 1
28        do for s_sweep ← s_min to (n_max − n)(w_min + s_min) + s_min
29           do s_i ← s_sweep;
30              for j ← 1 to n, j ≠ i
31              do s_j ← s_min;
32              if i < n
33                then s_{i+1} ← (n_max − n)(w_min + s_min) + 2s_min − s_sweep;
34              EXTRACT-S-PARAMETERS();
35
36        /* NDF Sweep */
37        for i ← 1 to n
38        do w_i ← w_min;
39           s_i ← ((n_max − n)/(n+1)) w_min + ((n_max + 1)/(n+1)) s_min;
40        for NDF_{k-1} ← NDF_min to NDF_max
41        do for NDF_{k+1} ← NDF_min to NDF_max
42           do EXTRACT-S-PARAMETERS();
```

Listing 5.1: Extraction of the base parameter set.

spacing is minimized. Next, the length is kept constant at an average value for the given metal layer ($l_{mean}\,(M_k)$) and the width of each wire is varied from w_{min} up to the maxi-

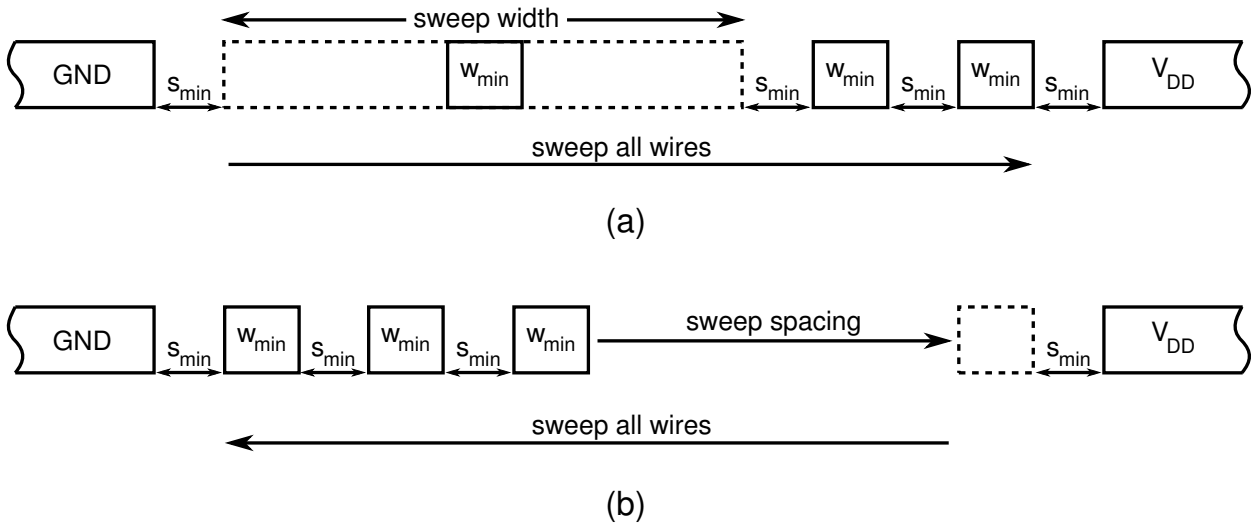

Fig. 5.9: Variable-width (a) and variable-spacing (b) sweeps during the initial parameter extraction.

mum allowed by the minimum spacing to its neighbors, with all the other wires kept to the minimum width and spacing. While doing this, the varying wire is placed exactly in the middle of the distance between its two direct neighbors. This approach minimizes the influence of wire length and spacing on the results obtained from the variable-width sweeps. An illustration of the variable-width sweep procedure is shown in Fig. 5.9(a). After that, a variable-spacing sweep is performed sequentially for every wire, as illustrated in Fig. 5.9(b), with all the wires kept at minimum width. Again, the influence of wire length and width on the results is minimized. During all the previous sweeps, the NDF of both upper and lower metal layers was set to zero, to avoid any influence of neighboring routed wires on these first results. Finally, the NDF sweeps add the information related to the presence of wires routed in the neighboring layers. During the sweeps, the density factors are varied from a minimum value NDF_{min}, which corresponds to either 0 (i.e. no routed wire), or to the value computed from the presence of only the GND and V_{DD} lines, depending on the position of the metal layer. The maximum value NDF_{max} corresponds to the maximum routing density present in the adjacent layer, including the power lines and maximum-width thick wires covering all the length of the segment. An illustration of such a maximum NDF case is shown in Fig. 5.10, where a single wire extends to the maximum width allowed by the fixed power grid. The EXTRACT-S-PARAMETERS() call designates the extraction of the S-parameter matrices for the given segment in a frequency sweep from 0 (DC) up to F_{max} with a step dictated by the application requirements.

5.3.2 Incremental Extrapolation

The input for the extrapolation procedure consists of the following:

- The request for computing a parameter $S_{p_1p_2}$ for a multi-wire interconnect segment,

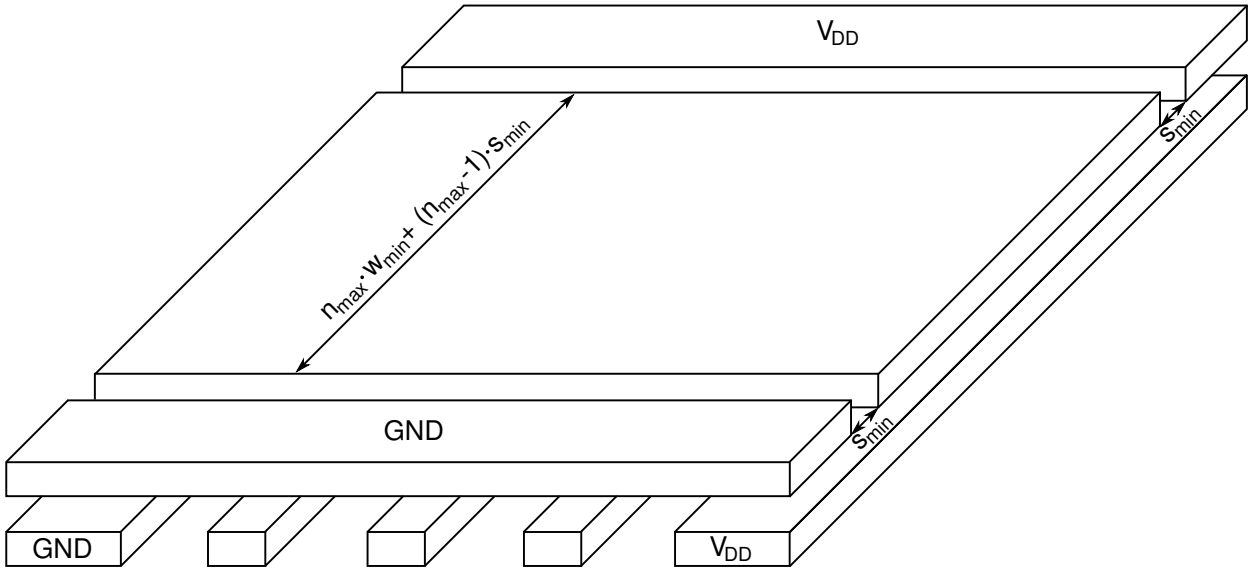

Fig. 5.10: Maximum NDF in the upper metal layer, with power grid and maximum-width signal line.

described by its length, metal layer, individual wire widths and spacings, as well as the NDF information for the adjacent metal layers;

- An initial set of extracted base parameters for the target CMOS process, built as described in Sec. 5.3.1.

The result represents the extrapolated $S_{p_1p_2}$ parameter for the specified segment, at all frequencies from DC to F_{max} (with the same step as the input data), in the standard MA-format.

First, the initial set of extracted parameters is parsed in a search for a base parameter for the extrapolation. This base parameter must be the closest-matching value for the requested parameter, i.e. a parameter describing an interconnect segment with the closest attributes to the requested one. To do this, a *matching rank* is first evaluated and the parameter with the highest matching rank will be afterwards selected. Considering a requested parameter $S_{p_1p_2}$, the factors which contribute to the matching rank and their respective weight are as follows:

- The wire length, which contributes to the wire resistance, coupling capacitance, and coupling inductance, thus having a high weight;
- The widths of the primary wires (connected to the ports p_1 or p_2), which contribute mainly to the wire resistance and coupling capacitance, having a high weight;
- The spacings of the primary wires, which mainly affect the coupling capacitance, with a medium weight;
- The widths of the secondary wires (not attached to the requested ports p_1 and p_2), which mainly influence the coupling inductance, with a relatively low weight;

- The spacings of the secondary wires, with a relatively low weight;
- The NDF, which affects only the coupling capacitance, hence with a relatively low weight.

The matching rank of an extracted parameter is computed as the sum of the individual weights for the structural details that match with the requested segment. If a closest-matching parameter is found, then it is used further as the base parameter for the extrapolation. If no structural attributes can be matched with any of the already-extracted results, then the base parameter must be extrapolated from e.g. the variable-length set. Concretely, if the wire length for the requested parameter $S_{p_1p_2}$ is l_r, then the extrapolated base parameter is computed as:

$$\begin{aligned} M_b^{p_1p_2} &= \text{extrap}\left([l_i], \left[M_{l_i}^{p_1p_2}\right], l_r, \text{'method'}\right) \\ A_b^{p_1p_2} &= \text{extrap}\left([l_i], \left[A_{l_i}^{p_1p_2}\right], l_r, \text{'method'}\right) \end{aligned} \tag{5.4}$$

where $M_b^{p_1p_2}$ and $A_b^{p_1p_2}$ are the magnitude, respective angle, of the base parameter for the extrapolation of $S_{p_1p_2}$, $[l_i]$ represents the set of wire lengths available in the initial extracted set, while $M_{l_i}^{p_1p_2}$ and $A_{l_i}^{p_1p_2}$ are the magnitude, respectively angle, of $S_{p_1p_2}$ for the segment with wire length l_i from the initial extracted set. The keyword 'method' designates the desired extrapolation function, which can be based either on linear interpolation, piece-wise cubic hermite polynomials, cubic interpolation, or cubic spline interpolation with smooth derivatives, to name only a few. The results shown in this work have been obtained with a cubic spline interpolation method, which proved to offer the best precision.

The base parameter represents the very first approximation of the requested $S_{p_1p_2}$ value. Because in the most cases the structural attributes of the requested segment do not coincide with the attributes related to the base parameter, a set of incremental corrections for each structural element must be further applied as explained in the following. Let's first assume that the wire length related to the base parameter is l_b. Then, two parameter values are extrapolated from the length-sweep results, one for l_b and one for the requested wire length l_r:

$$\begin{aligned} M_{l_b}^{p_1p_2} &= \text{extrap}\left([l_i], \left[M_{l_i}^{p_1p_2}\right], l_b, \text{'method'}\right) \\ M_{l_r}^{p_1p_2} &= \text{extrap}\left([l_i], \left[M_{l_i}^{p_1p_2}\right], l_r, \text{'method'}\right) \end{aligned} \tag{5.5}$$

The corresponding angle values $A_{l_b}^{p_1p_2}$ and $A_{l_r}^{p_1p_2}$ are computed in a similar way:

$$\begin{aligned} A_{l_b}^{p_1p_2} &= \text{extrap}\left([l_i], \left[A_{l_i}^{p_1p_2}\right], l_b, \text{'method'}\right) \\ A_{l_r}^{p_1p_2} &= \text{extrap}\left([l_i], \left[A_{l_i}^{p_1p_2}\right], l_r, \text{'method'}\right) \end{aligned} \tag{5.6}$$

Next, two variable-length *correction terms* $\Delta_l M^{p_1p_2}$, respectively $\Delta_l A^{p_1p_2}$ are computed as the following differences:

$$\begin{aligned} \Delta_l M^{p_1p_2} &= M_{l_r}^{p_1p_2} - M_{l_b}^{p_1p_2} \\ \Delta_l A^{p_1p_2} &= A_{l_r}^{p_1p_2} - A_{l_b}^{p_1p_2} \end{aligned} \tag{5.7}$$

and the *variable-length correction* is applied to the base parameter as follows:

$$\begin{aligned} M_b^{p_1p_2} &= M_b^{p_1p_2} + w_c \cdot \Delta_l M^{p_1p_2} \\ A_b^{p_1p_2} &= A_b^{p_1p_2} + w_c \cdot \Delta_l A^{p_1p_2} \end{aligned} \tag{5.8}$$

where w_c is a correction weighting factor for the parameter inter-correlations and is therefore data-dependent.

Further, to take into account the influence of wire width and spacing, for each wire in the segment the following parameter values are extrapolated:

$$\begin{aligned} M_{w_{b,i}}^{p_1p_2} &= \text{extrap}\left([w_{j,i}], \left[M_{w_{j,i}}^{p_1p_2}\right], w_{b,i}, \text{'method'}\right) \\ M_{w_{r,i}}^{p_1p_2} &= \text{extrap}\left([w_{j,i}], \left[M_{w_{j,i}}^{p_1p_2}\right], w_{r,i}, \text{'method'}\right) \end{aligned} \tag{5.9}$$

$$\begin{aligned} M_{s_{b,i}}^{p_1p_2} &= \text{extrap}\left([s_{j,i}], \left[M_{s_{j,i}}^{p_1p_2}\right], s_{b,i}, \text{'method'}\right) \\ M_{s_{r,i}}^{p_1p_2} &= \text{extrap}\left([s_{j,i}], \left[M_{s_{j,i}}^{p_1p_2}\right], s_{r,i}, \text{'method'}\right) \end{aligned} \tag{5.10}$$

where $w_{b,i}$ and $w_{r,i}$ represent the width of wire i (with i varying from 1 to n) for the base and the requested parameter, respectively, while $s_{b,i}$ and $s_{r,i}$ are the corresponding spacing values. The sets $[w_{j,i}]$ and $[s_{j,i}]$ contain the width and spacing arrays employed in the sweeps from Sec. 5.3.1 (see Listing 5.1 lines 14, respectively 28). In addition, the angle components $A_{w_{b,i}}^{p_1p_2}$, $A_{w_{r,i}}^{p_1p_2}$, $A_{s_{b,i}}^{p_1p_2}$, and $A_{s_{r,i}}^{p_1p_2}$ are obtained in a similar way.

After computing the correction terms $\Delta_{w_i} M^{p_1p_2}$, $\Delta_{w_i} A^{p_1p_2}$, $\Delta_{s_i} M^{p_1p_2}$, and $\Delta_{s_i} A^{p_1p_2}$ as the corresponding differences from the previously-extrapolated values, the *variable-width* and *variable-spacing corrections* are applied:

$$\begin{aligned} M_b^{p_1p_2} &= M_b^{p_1p_2} + w_c \sum_{i=1}^{n} (\Delta_{w_i} M^{p_1p_2} + \Delta_{s_i} M^{p_1p_2}) \\ A_b^{p_1p_2} &= A_b^{p_1p_2} + w_c \sum_{i=1}^{n} (\Delta_{w_i} A^{p_1p_2} + \Delta_{s_i} A^{p_1p_2}) \end{aligned} \tag{5.11}$$

Finally, the *variable-NDF correction* is computed and applied:

$$\begin{aligned} M_b^{p_1p_2} &= M_b^{p_1p_2} + w_c \cdot \Delta_{NDF} M^{p_1p_2} \\ A_b^{p_1p_2} &= A_b^{p_1p_2} + w_c \cdot \Delta_{NDF} A^{p_1p_2} \end{aligned} \tag{5.12}$$

where the correction terms $\Delta_{NDF} M^{p_1p_2}$ and $\Delta_{NDF} A^{p_1p_2}$ represent the differences between the extrapolated parameters for the base NDF and for the requested NDF values.

The correction steps applied to the base parameter are incremental and the influences of various wire attributes on the S-parameters are treated as independent. Although minimized, a non-zero residual correlation still exists between the individual influences, especially in the case of wire width variation, which has a significant influence on the spacing, see e.g. Fig. 5.9(a). Thus, a correction weighting factor $w_c < 1$ is employed, which accounts for the residual correlation and prevents therefore an overscaling of the final corrected value. A further correction of the extrapolated parameters is provided in Sec. 5.3.3.

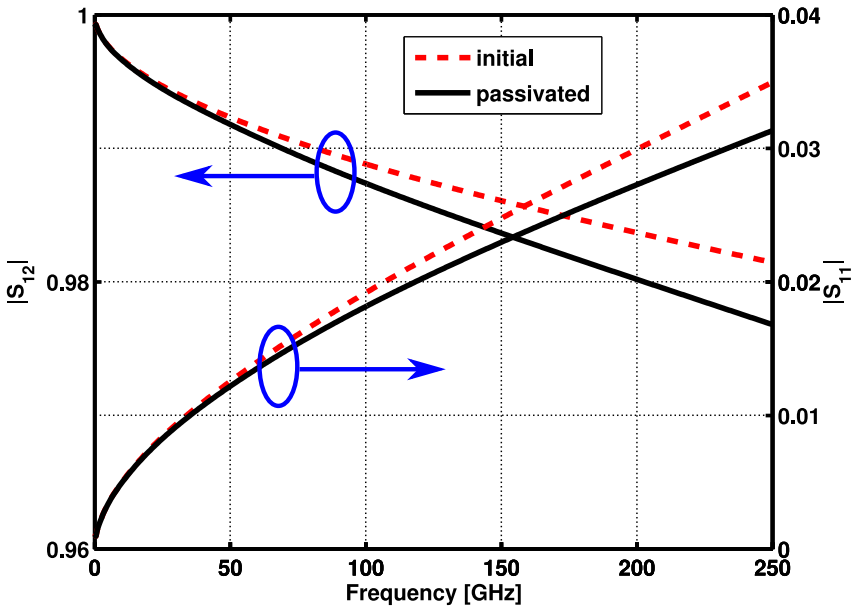

Fig. 5.11: Passivation example for a single-wire interconnect segment (metal 1, l = 10 µm, w = 400 nm, s = 810 nm).

5.3.3 Passivity Enforcement

Both measured and extrapolated S-parameters must exhibit a passive behavior, i.e. the interconnect model must dissipate active power, as opposed to generate it, at any value of the input voltage and at any frequency. Here, a passivation enforcement criterion is employed, based on the correction of the eigenvalues of the admittance matrix [72]. First, the Y-parameter matrix can be computed from the S-parameter matrix as follows [84]:

$$\mathbf{Y} = \mathbf{G}_{ref}^{-1} \cdot \mathbf{Z}_{ref}^{-1} \cdot (\mathbf{S} + \mathbf{E})^{-1} \cdot (\mathbf{E} - \mathbf{S}) \cdot \mathbf{G}_{ref} \tag{5.13}$$

where $\mathbf{Z}_{ref} = Z_{ref} \cdot \mathbf{E}$ is the reference impedance matrix, $\mathbf{G}_{ref} = \frac{1}{\sqrt{|Z_{ref}|}} \cdot \mathbf{E}$ is the reference conductance matrix, and $\mathbf{E}$ is the identity matrix. The passivity criterion requires the real part of the $\mathbf{Y}$ matrix to be positive definite [72], i.e. the eigenvalues of $Re\{\mathbf{Y}\}$ to be all positive. This relatively simple technique ensures both the passivity and the stability of the model. A more detailed discussion on passivity and stability conditions can be found in [158].

After setting the negative eigenvalues of $Re\{\mathbf{Y}\}$ to zero as in [78], the real part is recomputed as:

$$Re\{\mathbf{Y}\} = \mathbf{V} \cdot \mathbf{D}_{corr} \cdot \mathbf{V}^{-1} \tag{5.14}$$

where $\mathbf{V}$ contains the eigenvectors of $Re\{\mathbf{Y}\}$ and $\mathbf{D}_{corr}$ is a diagonal matrix with the corrected eigenvalues. The S-parameter matrix is recomposed from the corrected admittance matrix as:

$$\mathbf{S} = \mathbf{G}_{ref} \cdot (\mathbf{E} - \mathbf{Z}_{ref} \cdot \mathbf{Y}) \cdot (\mathbf{E} + \mathbf{Z}_{ref} \cdot \mathbf{Y})^{-1} \cdot \mathbf{G}_{ref}^{-1} \tag{5.15}$$

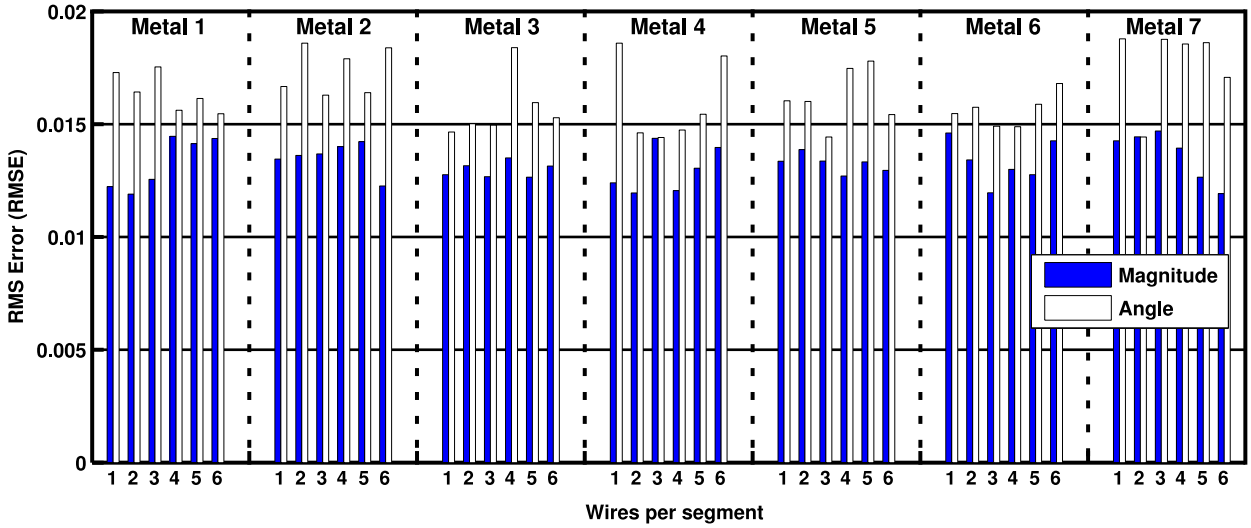

Fig. 5.12: RMS error between extrapolated and extracted results for the entire range of tested interconnect segments.

Fig. 5.11 shows two extracted S-parameters before and after the passivation, for a 10-μm single-wire interconnect segment placed on the M_1 layer. It can be observed that the passivity correction becomes more substantial as the frequency increases, which shows that measurement and numerical computation errors increase with frequency.

5.4 Experimental Validation

In order to assess the overall precision of the extrapolation method, a wide range of interconnect segments in the 90-nm technology have been tested, with up to six wires per segment and varying from metal 1 up to metal 7. In every case, the evaluation has been performed on a "difficult", non-standard segment configuration, with each wire having an individual width and spacing, randomly assigned with a uniform distribution between the minimum and maximum values allowed by the design rules.

The RMS error (RMSE) between the extrapolated parameters and the parameters obtained with the field simulator has been computed from all the $(2n)^2$ S-parameters of each segment as:

$$RMSE = \sqrt{\frac{1}{N_f \cdot 4n^2} \sum_{i,j=1}^{2n} \sum_{f_k=1}^{N_f} \left(\hat{S}_{ij}^{f_k} - S_{ij}^{f_k} \right)^2} \tag{5.16}$$

where $\hat{S}_{ij}$ and S_{ij} represent the extrapolated, respectively the extracted parameter values, and f_k is the frequency index, indicating the steps from DC up to the maximum frequency of interest. The results for all the investigated configurations are summarized in Fig. 5.12, where the angle values have been normalized to 360°. The maximum absolute errors were $1.8 \cdot 10^{-2}$ in magnitude and 6.8 degrees in angle. The main causes for the exhibited deviations are given by:

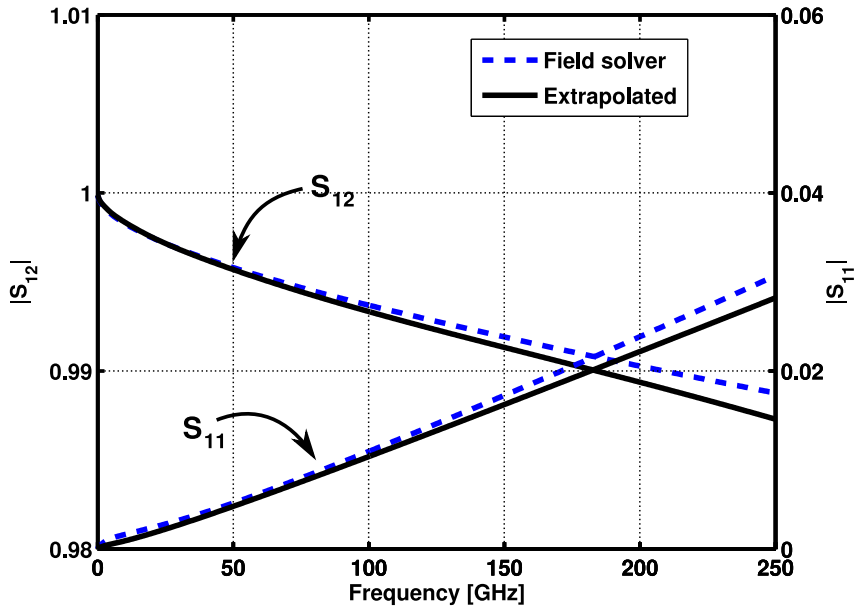

Fig. 5.13: Magnitude of extrapolated and extracted parameters for a single-wire interconnect segment.

- The residual correlations between the wire attributes, especially width and spacing;
- The non-optimal passivity correction [158, 72];
- The precision of the extrapolation method, which is limited by the number of samples available in the initial set.

A more detailed example is shown in Fig. 5.13 and 5.14 for a single-wire M_1-segment with l =10 µm, w = 580 nm, s = 420 nm, and NDF = 35%. The plots show the values for S_{11} and S_{12}, while the other two parameters, S_{22} and S_{21}, are virtually identical with S_{11}, respectively S_{12} due to the inherent symmetry of the wire. From Fig. 5.13, one can see that $|S_{12}|$, which reflects the power wave transmission from port 2 to port 1, reaches a maximum of 1 at DC and starts to drop relatively fast as the frequency increases into the multi-GHz range. This behavior shows the rate of losses in signal power with the frequency increase, for a direct transmission across the line from port 2 to port 1, and points out the expected signal integrity issues which influence the interconnect at high frequencies. The level of signal reflections at port 1 is shown by the plot of $|S_{11}|$, which indicates that reflections are essentially zero at DC, but increase with the frequency. Again, the expected signal reflections within the wire are here illustrated and quantified. The extrapolated model is overall in a good agreement with the directly-extracted parameter data.

A further example is depicted in Fig. 5.15 for a three-wire segment in metal 4, with the attributes presented in Tab. 5.1. Only six parameters have been selected for the plot from the complete set of 36, in order to maintain a reasonable amount of visible detail.

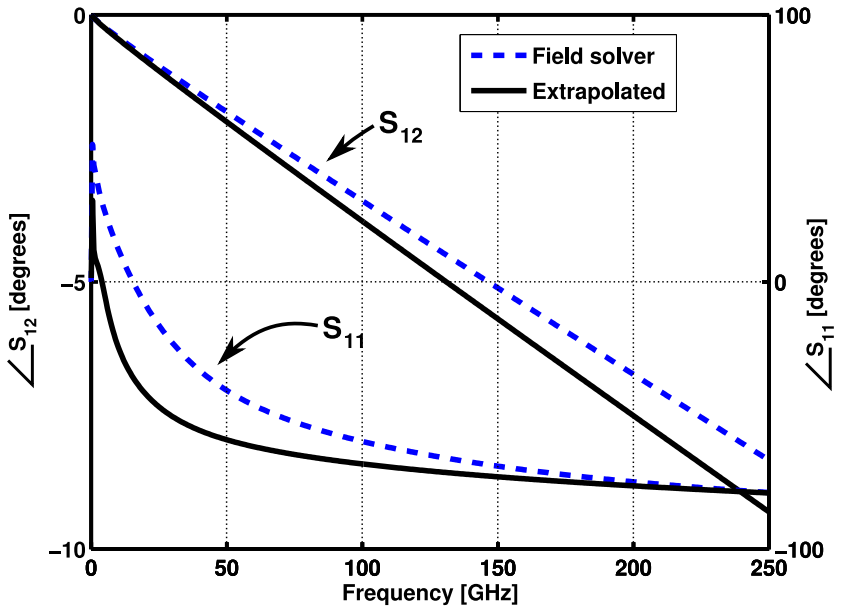

Fig. 5.14: Angle values for the extrapolated and extracted parameters of a single-wire segment.

Wire	Length	Width	Spacing	NDF_3	NDF_5
1	2.5 mm	140 nm	200 nm	70%	25%
2		300 nm	290 nm		
3		300 nm	295 nm		

Tab. 5.1: Wire attributes for a three-wire M_4-segment.

S_{14}, S_{25}, and S_{36} represent the direct signal transfer along the three wires, and show substantial losses at the maximum frequency of interest. S_{13}, S_{24}, and S_{35} reflect the crosstalk between wires 3 and 1, 1 and 2, respectively 2 and 3. Here it can be noticed that the crosstalk also increases significantly with the frequency. Thus, we can clearly observe that a wide-frequency interconnect model is extremely important to quantify the amount of performance losses at very high switching speeds. Beyond these observations, a very good agreement between the extrapolated model and the directly-extracted parameters can be noticed as well. In order to obtain a more detailed quantitative evaluation of the modeling performance, the RMS error has been computed for every parameter across the investigated frequency range. The results are displayed in Fig. 5.16 where the parameters which are identical due to the wire symmetry have been omitted, i.e. S_{44} with S_{11}, S_{41} with S_{14}, S_{55} with S_{22}, S_{52} with S_{25} etc. The measured errors are in line with the previous evaluations from Fig. 5.12.

Next, the extrapolated S-parameter models have been tested within transient circuit-level simulations. For this purpose, a SPICE-level simulator which supports the direct modeling of n-port elements using S, Y, or Z-parameter descriptions [30] has been used.

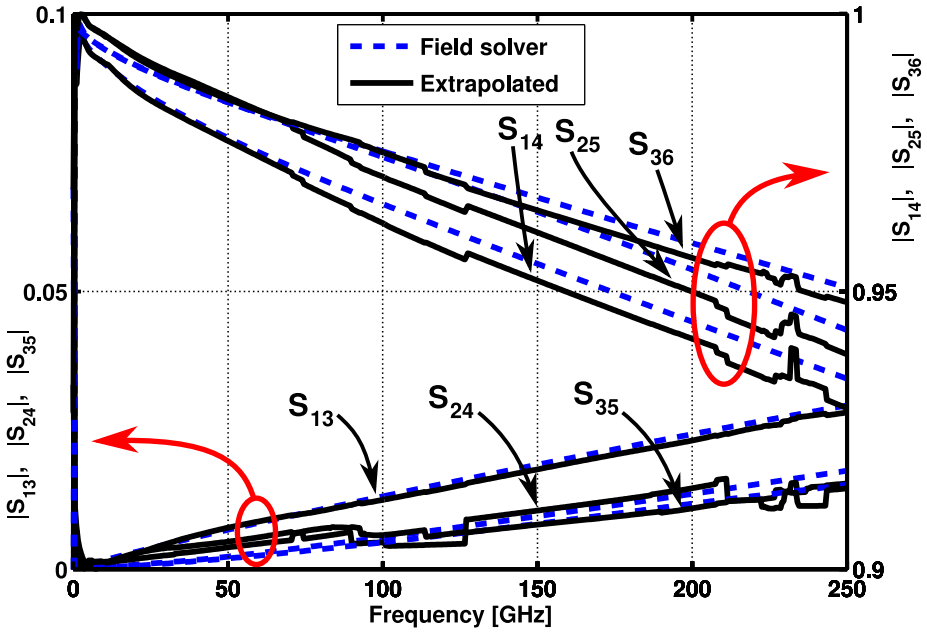

Fig. 5.15: Magnitude plot of six S-parameters for a three-wire M_4-segment.

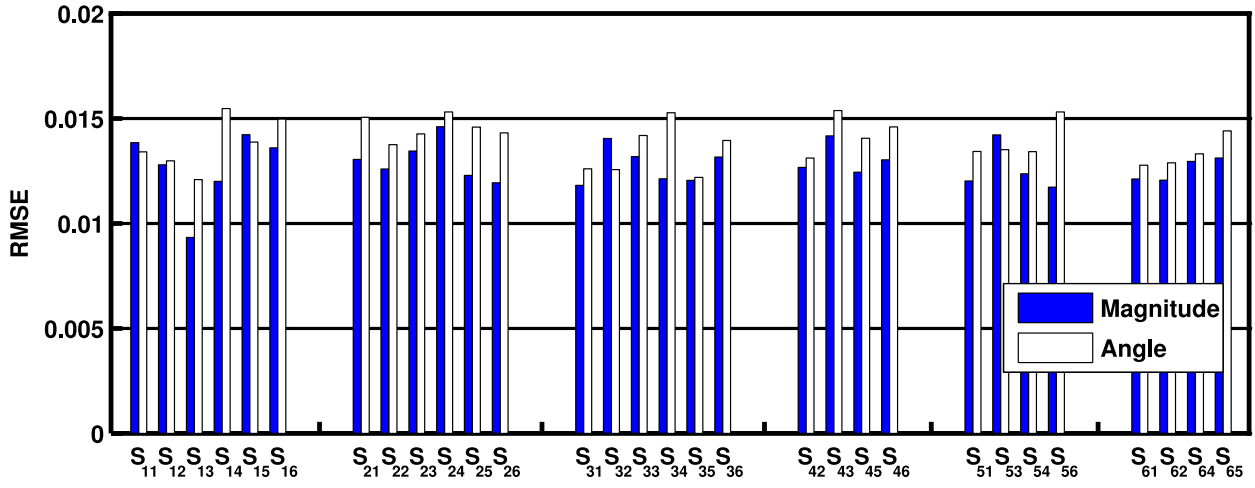

Fig. 5.16: RMS errors between extrapolated and directly-extracted parameters (three-wire M_4-segment).

Within the modeling framework, the extrapolated S-parameters are saved as standard Touchstone files which are directly supported by the simulator. The circuit configuration employed for the tests is shown in Fig. 5.17, where each wire of the interconnect model is driven and terminated independently.

The signal delay has been measured across each wire with both quiet and switching neighboring lines and the results obtained using the extrapolated S-parameters and parameters extracted with the field solver were compared. A detailed view of the results is shown in Fig. 5.18 for three-wire interconnect segments of various lengths placed on

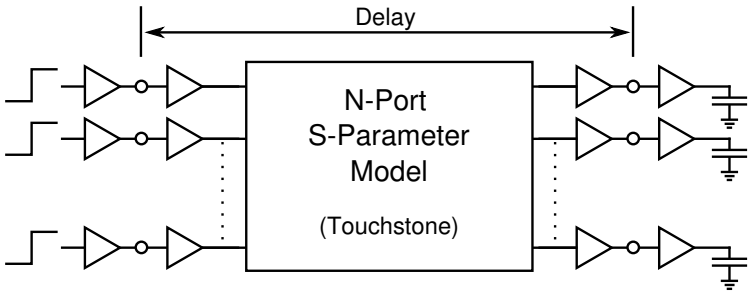

Fig. 5.17: Circuit employed for the transient simulations.

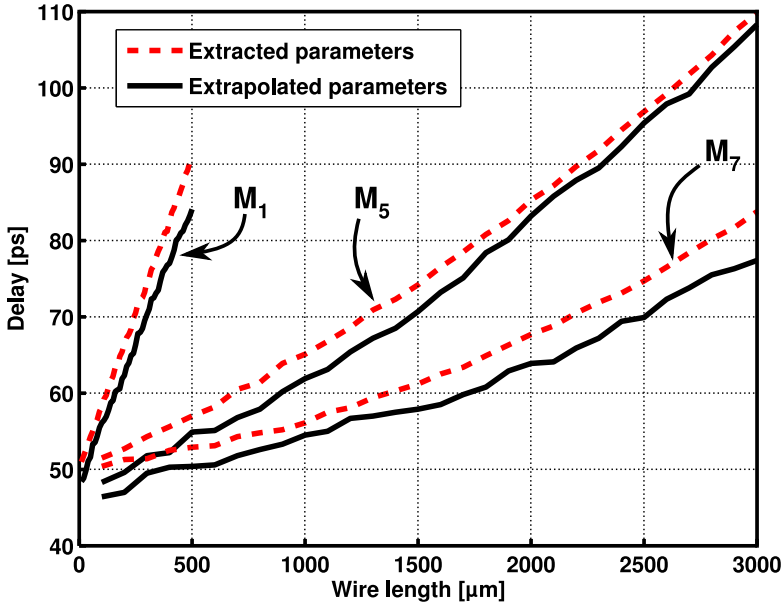

Fig. 5.18: Signal propagation delays from three-wire interconnect segments placed on three metal layers.

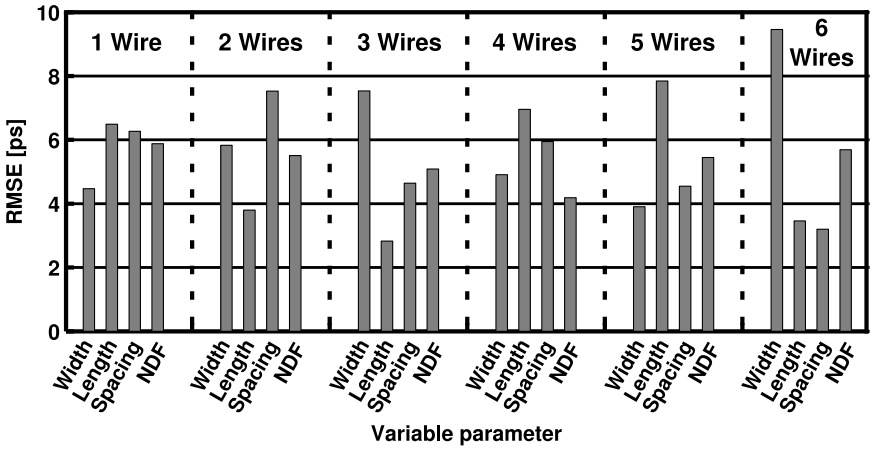

Fig. 5.19: Delay RMSE for the transient simulations of interconnect segments on metal 5.

		Number of wires per segment					
		1	2	3	4	5	6
Metal layer	1	4.28%	7.37%	7.77%	10.01%	10.47%	11.88%
	2	5.04%	6.69%	8.19%	10.39%	10.24%	11.46%
	3	5.46%	5.56%	6.94%	8.28%	9.51%	11.50%
	4	6.93%	5.34%	7.78%	8.83%	9.94%	11.74%
	5	6.79%	7.32%	6.21%	9.32%	10.39%	10.51%
	6	6.37%	5.78%	7.41%	8.60%	11.27%	**12.21%**
	7	5.56%	7.18%	8.38%	8.97%	9.29%	11.14%

Tab. 5.2: Maximum relative delay error across all considered metal layers and wires per segment.

M_1, M_5, and M_7, in each case with one neighbor switching at the same time in opposite direction. To better quantify the differences, the RMS and relative errors for every series of simulations have been measured. First, the RMS errors on each metal layer were measured for each number of wires per segment and for each sweep of the wire attributes. A detailed plot of the RMS errors in the case of metal 5 is depicted in Fig. 5.19. Each RMSE value is computed across the attribute sweep range, across the investigated frequency range, and across all S-parameters.

In addition to the RMSE values, the maximum relative delay error has been evaluated across all metal layers and numbers of wires per segment. For each metal layer and for each wire number we have varied individually the segment length, wire widths, wire spacings, and the NDF, and the maximum error was evaluated as:

$$\varepsilon_r^{max} = \max\left\{ \max_{l_i}\left\{\left|\frac{\hat{\delta}_{l_i}-\delta_{l_i}}{\delta_{l_i}}\right|\right\}, \max_{w_{j,i}}\left\{\left|\frac{\hat{\delta}_{w_{j,i}}-\delta_{w_{j,i}}}{\delta_{w_{j,i}}}\right|\right\}, \max_{s_{j,i}}\left\{\left|\frac{\hat{\delta}_{s_{j,i}}-\delta_{s_{j,i}}}{\delta_{s_{j,i}}}\right|\right\}, \max_{NDF_i}\left\{\left|\frac{\hat{\delta}_{NDF_i}-\delta_{NDF_i}}{\delta_{NDF_i}}\right|\right\}\right\} \tag{5.17}$$

where $\hat{\delta}$ is the delay measured with the extrapolated model and δ is the delay obtained using the directly-extracted parameters. It is to be noted that the attribute variations during these tests have been selected in such a way that they do not include the same values found in the initial set, for a better evaluation of the extrapolation performance. Wire length has been varied between 1 and 500 µm for the local layers (M_1 to M_4), between 100 µm and 5 mm for intermediate layers (M_5 and M_6) and from 3 mm to 5 cm for the global layer (M_7). Wire width and spacing have been varied from the minimum design rule for each metal layer up to the maximum allowed by the shielding grid and the number of wires between two successive power lines (see Listing 5.1 lines 14 and 28). Additionally, the minimum NDF was always zero, while the maximum NDF varied between 85% and 100% depending on the wire length (see Fig. 5.10).

The evaluated maximum relative errors are shown in Tab. 5.2. The maximum error generally increases with the number of wires per segment, since the total number of S-

parameters increases quadratically with the number of wires. As it can be seen, the developed method achieved during the tests a maximum delay error of 12.21% for six-wire segments placed on the 6^{th} metal layer.

5.5 Summary

Technology-accurate wide-bandwidth interconnect models are needed for the precise estimation of signal delays, crosstalk, and energy losses across the complex on-chip communication structures. Traditional transmission-line distributed models offer a good accuracy at the expense of limited frequency validity and complex mutual inductance extractions, therefore they always imply a tradeoff between precision and computational efficiency. On the other hand, full-wave interconnect analyses provide a high accuracy at all frequencies, but require extensive numerical computations which can not be performed in real time. In addition, the amount of possible wire configurations across various lengths, widths, spacings, and metal layers increases exponentially the complexity of the modeling problem.

This chapter has introduced a computationally-efficient wide-bandwidth characterization method for arbitrary interconnect structures, which is based on the incremental extrapolation of S-parameters. The method defines a set of *a priori* parameter extractions, designed to reflect the particularities of a given manufacturing process. This initial set of parameters can be extracted with high precision within an independent time frame prior to the application, and represents a data base for the subsequent computations. Further, it has been shown how the initial set can be extracted by means of a full-wave field simulator and a structural model reflecting the technological process. It has also been shown that the complexity of covering the large set of possible wire attributes can be substantially reduced by minimizing the correlations between the segments employed in the initial set. A further measure to limit the complexity is to consider a fixed power grid and orthogonal routing directions. Moreover, the presence of wire segments in the neighboring metal layers has been modeled by introducing a density factor which indicates the amount of coupling capacity between the metal layers. Next, an incremental extrapolation procedure is performed in real time for every parameter request, which includes a search for a best-matching base parameter and a suite of extrapolated corrections applied for every wire attribute. A passivation enforcement criterion has been also described, which ensures that the obtained model is stable and exhibits a passive behavior.

The model has been tested across all metal layers and up to six wires per segment and the results have been compared with an industry-standard field simulator. The results show a good agreement with the directly-extracted parameters and lie within $2{\cdot}10^{-2}$ and 7 degrees for magnitude and angle values, respectively. Another suite of extensive tests has been performed in the time domain, within circuit-level simulations. The results summarized in Tab. 5.2 show a maximum error of less than 12.5%.

Chapter 6

Methodology Binding

Contents

The previous three chapters have presented a modeling methodology focused on variability propagation and technology accuracy for the optimization of on-chip communication architectures. The optimization is guided by the top-level performance macromodels for delay and energy consumption, which have been introduced in Sec. 3.5 and 3.6 and have been extended with circuit-level models for different signaling methods in Sec. 4.4. In addition, a refinement of the interconnect models for accurate validations has been developed and presented in chapter 5. A first method for the design space exploration considering the mapping and scheduling of processing tasks on resources has been presented in Sec. 3.7 and the inclusion of communication activities in the form of communication nodes has been discussed in Sec. 4.4.5. Nevertheless, the inclusion of circuit-level models in the communication nodes enables the use of signaling choice, voltage scaling, and body biasing as additional exploration techniques in the global optimization loop.

The results of applying the developed methodology to an application scenario are presented and analyzed in this chapter. First, an application profile is derived for a video de-

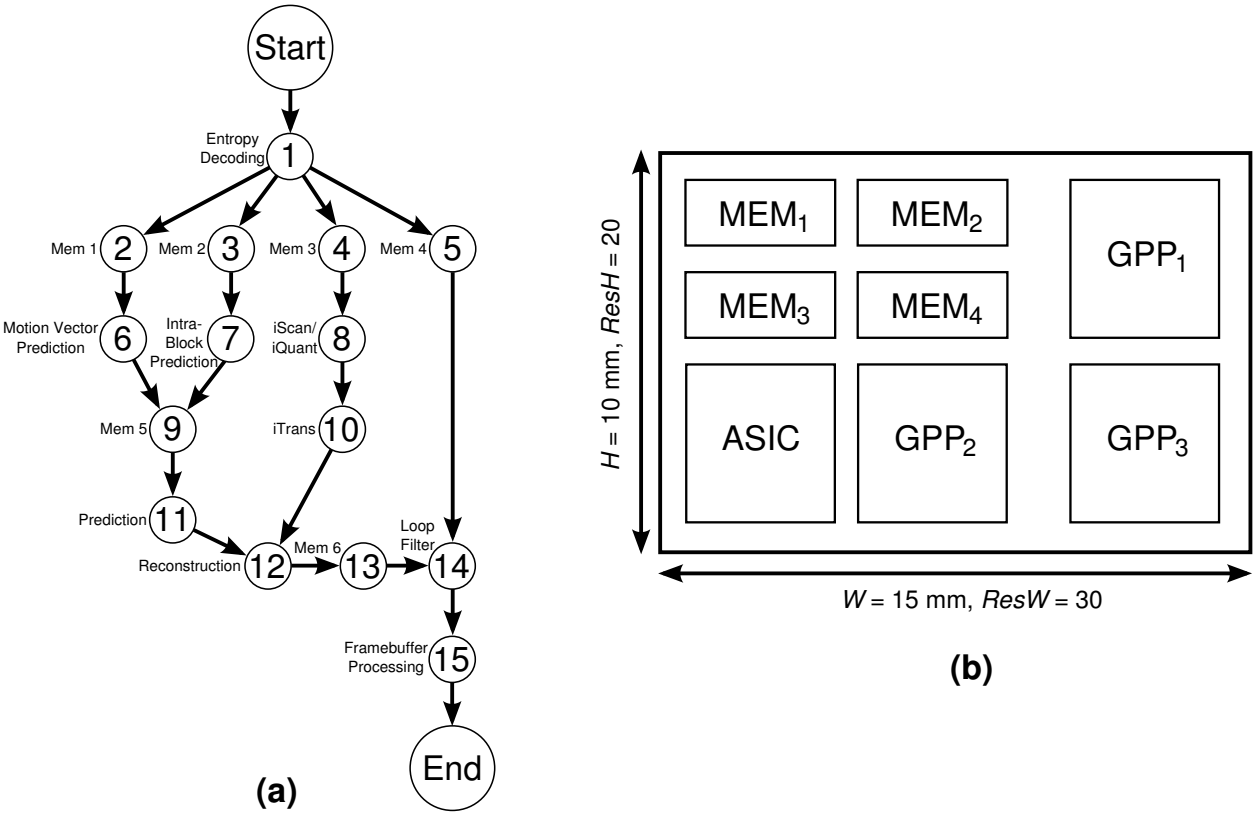

Fig. 6.1: Application task graph (a) and the considered SoC architecture (b).

coding example and the description of the target MPSoC architecture including floorplan information and process parameter variations is described. Second, a set of parameters which guide the design space exploration method toward the desired optimization criteria are presented. Within this context, the parametrization of the cost function using quantile functions and weighting factors is discussed. Furthermore, the synthesis results are presented for a delay-driven optimization and for an energy-oriented exploration and the accuracy of the modeling method for communication segments is investigated.

6.1 Application Profile Example

A video decoding application based on the H.264 standard [162] has been selected in this chapter to illustrate the results of the proposed methodology. The execution flow illustrated by the task graph from Fig. 6.1(a) contains typical decoding tasks, such as entropy decoding, motion-compensated prediction, and the integer quantization and transform operations, but also memory access operations between the decoding tasks, denoted as "Mem i".

Parameter	GPP_i		ASIC		MEM_i	
	μ	σ	μ	σ	μ	σ
P_{l,RT_i} [mW]	18	167	1.8	16.7	3.6	83.5
$P_{d,i}^{RT_j}$ [mW]	480	5	100	2	100	2
T_{1,RT_j}^{x} [µs]	1152	72.3	–	–	–	–
$T_{2,3,4,5,9,13,RT_j}^{x}$ [µs]	–	–	–	–	132	8.7
T_{6,RT_j}^{x} [µs]	72	4.5	24	1.5	–	–
T_{7,RT_j}^{x} [µs]	74	4.7	25	1.6	–	–
T_{8,RT_j}^{x} [µs]	1654	119.75	551	39.9	–	–
T_{10,RT_j}^{x} [µs]	2636	162.47	879	54.16	–	–
T_{11,RT_j}^{x} [µs]	146	11.2	49	3.73	–	–
T_{12,RT_j}^{x} [µs]	1754	126.57	584	42.19	–	–
T_{14,RT_j}^{x} [µs]	2549	154	849	51.33	–	–
T_{15,RT_j}^{x} [µs]	1367	94	–	–	–	–

Tab. 6.1: Application profile parameters used for the communication synthesis.

6.1.1 Description of the SoC Resource Set

The target MPSoC architecture for application mapping and communication synthesis is illustrated in Fig. 6.1(b) and contains a mixture of generic general-purpose processors (GPP) and memory blocks. In addition, for checking different synthesis alternatives, a high-speed ASIC block is considered as an alternative for the execution of data-intensive operations.

The estimated profile parameters for the selected application are presented in Tab. 6.1. Here, the leakage power P_{l,RT_i} of each resource type has been estimated for the considered 1.0-V, 90-nm CMOS process from a drain current value of 0.18 nA at zero V_{gs}, as indicated by Fig. 4.6. A number of approx. 10^6 leakage paths has been further assumed for the GPP cores, 10^3 paths for the ASIC, and $5 \cdot 10^4$ paths for the memory modules. It is to be noted, that the leakage power exhibits a very large variation with the process parameters and it has been modeled with a lognormal distribution (starting from a standard deviation of 0.167 µA of the leakage current).

The dynamic power consumption values $P_{d,i}^{RT_j}$ have been approximated by assuming an 800 MHz frequency for the GPP cores and an average of e.g. 0.6 mW/MHz power consumption. For the ASIC and memory blocks a lower average value of 100 mW power consumption has been assumed.

Execution times of processing tasks on the GPP cores have been estimated assuming the decoding of a high-definition resolution image of 1280×720 pixels and 4×4 integer transforms. Furthermore, single-cycle integer operations have been assumed on the GPPs and the ASIC execution has been assumed to be three times faster than the GPP execution. The execution times of the memory access operations have been averaged from the total

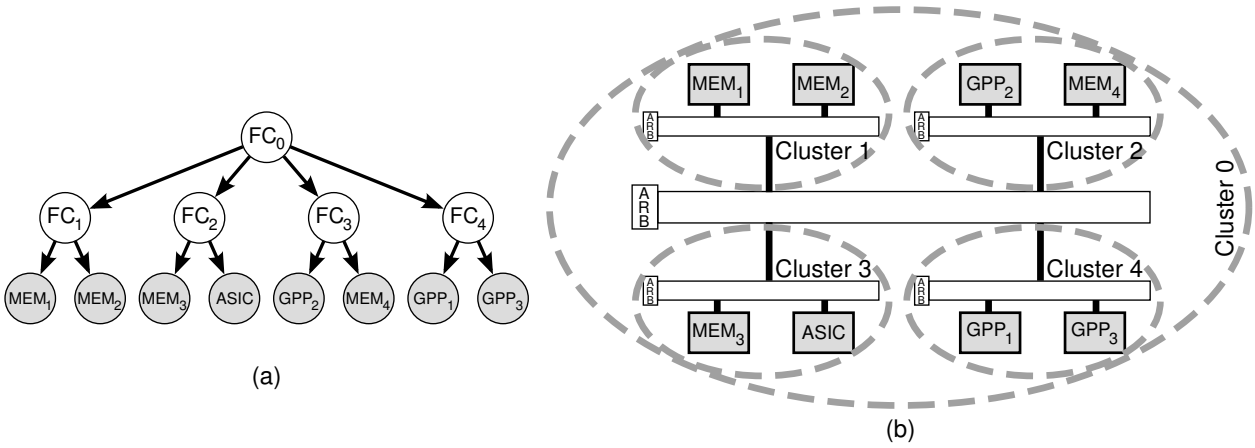

Fig. 6.2: Floorplan cluster tree for the processing resources (a) and one possible inter-resource connection in a hierarchical bus architecture (b).

Floorplan Cluster	Length	Bitwidth W_s
FC_0	20 mm	16
FC_1	1 mm	4
FC_2	100 µm	2
FC_3	100 µm	2
FC_4	1 mm	4

Tab. 6.2: Floorplan cluster parameters.

number of required load and store operations. Finally, it has been assumed, that PN_1 and PN_{15} can be only executed on the GPP cores, whereas all the intermediate tasks can be run on both GPP and the ASIC block.

A silicon die area of $15 \times 10\,\text{mm}^2$ has been further assumed, in a 1.0-V 90-nm CMOS process, with the processors cores and ASIC block occupying an area of about $4 \times 4\,\text{mm}^2$ and the memory modules around 50% of the processor core area. For spatially-correlated process parameter variations, a 30×20 grid has been used with a PWL correlation model as described by equation (4.24) with the parameters $d_d = 6\,\text{mm}$ and $\rho_r = 0.2$.

6.1.2 Floorplan Cluster Tree

Fig. 6.2 depicts the associated floorplan cluster tree for the processing resources considered in the previous section and assumes local clusters connecting MEM_3 with the ASIC block and MEM_4 with GPP_2. Further, GPP_1 and GPP_3 are grouped in a local cluster, as well as MEM_1 with MEM_2. Since the exact parameters of the communication segments can be specified only after routing, a generic cluster description is employed instead in the design space exploration. The information used to describe the five floorplan clusters is given in Tab. 6.2.

For the global cluster, which connects all resources together, an average length of 20 mm and a segment width of 16 bits have been considered. Clusters FC_2 and FC_3, which connect the ASIC and GPP_2 with a memory block have been assumed to be only 100 µm long in order to balance the choice of signaling circuits, since voltage-mode signaling is more effective for such short lengths. Furthermore, the segment bitwidth has been assumed to increase with segment length, to improve the performance of global data transfers.

6.1.3 Design Space Exploration Method

The search for an optimized communication architecture employs the macromodels for processing and communication nodes described in Sec. 3.5, 3.6, and 4.4.5 for estimating the performance metrics of possible solutions in the design space. Hereby, the exploration method is implemented as a nested optimization loop based on simulated annealing. The results presented in this chapter were obtained using an exponential cooling schedule of the form $T_{k+1} = 0.9 \cdot T_k$ in 100 steps and 10 000 local iterations for each cooling step.

As a starting point for the exploration, an initial architecture is generated by randomly assigning the processing nodes to compatible resources and creating the initial scheduling lists dictated by inter-task dependencies. Then, communication nodes are inserted in the macromodels for the inter-resource segments and the signaling circuit for each segment is selected randomly from the described PCM and voltage-mode techniques. For this initial solution, the supply voltage V_{dd} and the body bias V_{bs} are set at the nominal values of 1.0 V and 0 V, respectively.

At each local iteration, the next jump in the solution space is decided using a uniform random number generator and a set of probabilities for each jump type. In this way, the next solution is searched by applying one of the following modifications to the current architecture:

- Remap a PN from one resource to another, with a probability $P_{remap} = 0.05$;
- Change the scheduling order of two PNs on a resource, with a probability $P_{reschedule} = 0.05$;
- Change the signaling circuit on a communication segment, with a probability $P_{sig} = 0.1$;
- Change the supply voltage of a communication segment, with a probability $P_{V_{dd}} = 0.4$;
- Change the body bias on a communication segment, with a probability $P_{V_{bs}} = 0.4$.

Supply voltage scaling is performed between 1.0 V and 0.6 V in 100-mV steps, while the body bias is adjusted from -0.4 V to +0.4 V in 100-mV steps. Note, that the jump probabilities have been selected with a focus on communication architecture optimization.

Consequently, a signaling circuit change is performed twice as often as PN remapping or rescheduling. In addition, voltage scaling and body biasing are performed each four times as often as the change of signaling circuits, to provide a better coverage of these optimization resources at the circuit level.

6.1.4 Cost Function Settings

Given the global system delay T^e_{global} and the global dynamic, respectively leakage energy consumptions $E_{d,total}$ and $E_{l,total}$, which are statistically evaluated using the performance macromodels, the cost function employed for the considered example has the following form:

$$\mathcal{C}\left(T^e_{global}, E_{d,total}, E_{l,total}\right) = w_T \cdot \frac{Q\left(T^e_{global}\right)}{Q\left(T_i\right)} + w_{E_d} \cdot \frac{Q\left(E_{d,total}\right)}{Q\left(E_{d,i}\right)} + w_{E_l} \cdot \frac{Q\left(E_{l,total}\right)}{Q\left(E_{l,i}\right)} \tag{6.1}$$

where w_T, w_{E_d}, and w_{E_l} represent the weights with which the influence of the individual performance metrics are considered in the optimization. Different values of the weights are employed in this chapter to provide an emphasis to either delay minimization or low energy consumption.

It is important to note, that the distributions T^e_{global}, $E_{d,total}$, and $E_{l,total}$ are strongly application-dependent and might be separated by several orders of magnitude. To ensure that the influence of each of them is appropriately considered in the cost function, a scaling of these values is performed by normalizing them to T_i, $E_{d,i}$, and $E_{l,i}$. The latter represent the performance metrics of the initial solution, as given by the macromodels.

Finally, since the performance metrics are evaluated as statistical distributions, a quantile function $Q : \mathbb{P} \rightarrow \mathbb{R}$ is used to extract the 99% inferior quantile of the performance metric distributions as explained in Sec. 3.4.4:

$$Q\left(p_X\left(x\right)\right) = z_{99\%,inf} = F_X^{-1}\left(0.99\right) \tag{6.2}$$

6.2 Evaluation of Synthesis Results

The application profile and exploration algorithm discussed in the previous section have been employed to find a communication architecture optimized with respect to different targets. Delay-oriented and energy-oriented optimizations achieve different resource mappings, scheduling lists, and communication circuit characteristics, such as signaling method and voltage levels. To illustrate the impact of the optimization choices and to show the benefit of the developed macromodels, the synthesis results are presented and discussed in each case. In addition, an overall evaluation of the macromodel accuracy is given for the synthesized communication segments based on the comparison with circuit-level simulations.

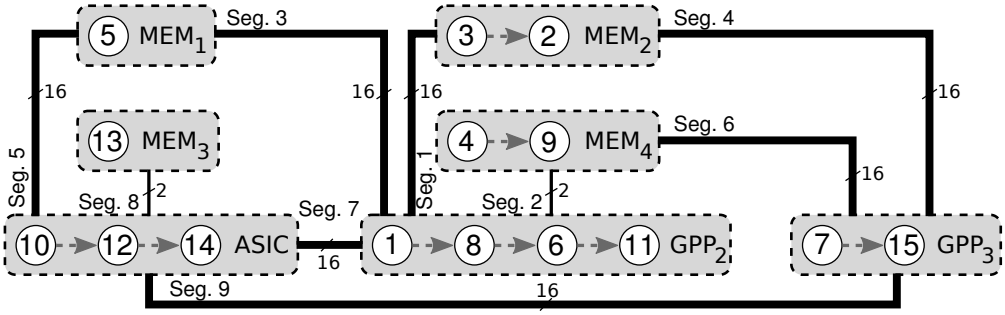

Fig. 6.3: Resource mapping configuration, scheduling sequences, and communication segments synthesized for minimum delay.

6.2.1 Delay-Optimized Architecture

To set the emphasis on delay minimization, the weighting factors in the cost function (6.1) have been set to $w_T = 0.98$ and $w_{E_d} = w_{E_l} = 0.01$. Using these settings, the optimization algorithm generated the task-resource mappings and scheduling sequences shown in Fig. 6.3 with 9 inter-resource communication segments. It is interesting to note, that the first GPP core is unused in this configuration. There is no additional parallelism which can be speculated by mapping PNs on GPP_1 and the additional inter-resource communication would only increase the overall system latency. Thus, GPP_1 can be turned off to reduce the power consumption. This configuration achieves a total system delay of 9.391 ms, a total leakage energy of 3.506 mJ, and a dynamic energy consumption of 2.305 mJ. All these performance metrics have been evaluated by extracting the 99% quantile from the statistical distributions evaluated using the developed macromodels.

The synthesis generates also the scheduling times for the processing nodes, which can be extracted from the nodes of the delay macromodel. Tab. 6.3 presents the obtained timing values for the PNs in the considered application, extracted from the statistical distributions for a desired parametric yield level of 99%. It is to be noted, that the start and end times for the PNs are computed considering also the inter-resource communication delays.

The synthesized communication segments shown in Fig. 6.3 are described in Tab. 6.4. It can be observed, that PCM signaling has been employed on the long segments connecting the ASIC, GPP_2, and GPP_3, with the memory blocks MEM_1 and MEM_2. On the other hand, the local segments connecting the ASIC with MEM_3 and GPP_2 with MEM_4 use voltage-mode signaling, which is faster than PCM at lengths of 100 µm.

The delays on the local segments 2 and 8 are relatively small and typical for the use of voltage-mode signaling at distances in the range of 100 µm. On the remaining segments, the delay values are in the range of 300 ps, as expected for the use of PCM signaling. Small delay variations from segment to segment are mainly the result of process parameter variations. Considering the leakage power results, PCM signaling circuits exhibit a significantly higher dissipation. This is mainly due to the larger number of current flow-

Processing Node PN_i	Start Time T_i^s [ms]	End Time T_i^e [ms]
PN_1	0.000	1.320
PN_2	2.021	2.157
PN_3	1.718	1.839
PN_4	1.388	1.508
PN_5	1.729	1.856
PN_6	3.158	3.255
PN_7	2.286	2.373
PN_8	1.576	2.559
PN_9	3.725	3.863
PN_{10}	2.985	3.829
PN_{11}	4.021	4.196
PN_{12}	4.700	5.244
PN_{13}	5.905	6.055
PN_{14}	6.799	7.627
PN_{15}	8.125	9.391

Tab. 6.3: Scheduled start and end times for processing nodes, evaluated as 99% inferior quantile from the statistical distributions.

Comm. Seg.	Delay [ps]	Leak. Power [nW]	Dyn. Power [nW]	Signaling Circuit	V_{dd} [V]	V_{bs} [V]
Seg. 1	380	180.345	91.358	PCM	1.0	0.0
Seg. 2	56	21.1	28.819	VM	1.0	0.4
Seg. 3	374	225.973	85.721	PCM	1.0	0.0
Seg. 4	379	137.423	79.855	PCM	1.0	0.0
Seg. 5	381	112.926	80.205	PCM	1.0	0.0
Seg. 6	376	223.036	86.331	PCM	1.0	0.0
Seg. 7	367	113.179	87.953	PCM	1.0	0.0
Seg. 8	61	23.392	28.701	VM	1.0	0.4
Seg. 9	373	112.838	91.150	PCM	1.0	0.0

Tab. 6.4: Parameters of the synthesized communication segments, evaluated as 99% inferior quantile from the statistical distributions.

ing paths, such as 8 leakage paths in the driver and 3 in the receiver, with respect to 4 leakage paths (two in the driver buffer and two in the receiver buffer) for the voltage-mode circuits. Nevertheless, the contribution of the pulse signaling current to the static power dissipation increases the difference between the two circuits. Most variations in the dynamic power values of the communication segments are due to the variations in the communication loads. Moreover, longer segments also have a relatively high dynamic power dissipation, due to the larger total line capacitance.

It is also important to note, that the V_{dd} and V_{bs} values chosen by the exploration algorithm were optimized for delay minimization. A nominal V_{dd} value ensures the mini-

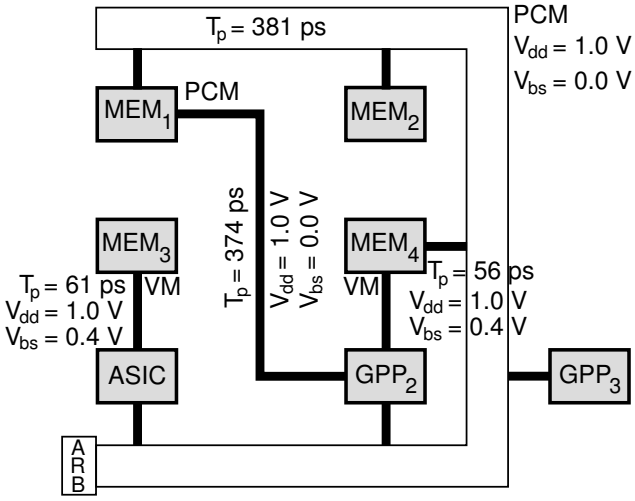

Fig. 6.4: Delay-optimized communication architecture synthesized as a shared bus and three point-to-point links.

Communication Link	Scheduled Activities
ASIC – MEM_3	$PN_{12} \rightarrow PN_{13}$, $PN_{13} \rightarrow PN_{14}$
GPP_2 – MEM_4	$PN_1 \rightarrow PN_4$, $PN_4 \rightarrow PN_8$, $PN_6 \rightarrow PN_9$, $PN_9 \rightarrow PN_{11}$
Shared Bus	$PN_1 \rightarrow PN_3$, $PN_1 \rightarrow PN_2$, $PN_3 \rightarrow PN_7$, $PN_2 \rightarrow PN_6$, $PN_7 \rightarrow PN_9$, $PN_5 \rightarrow PN_{14}$, $PN_{14} \rightarrow PN_{15}$
GPP_2 – MEM_1	$PN_1 \rightarrow PN_5$

Tab. 6.5: Scheduled communication activities on the synthesized architecture from Fig. 6.4.

mum latency, whereas the positive body bias of 0.4 V achieves a delay improvement on the voltage-mode lines, as illustrated also by the results from Fig. 4.25(b) from Sec. 4.3.3.

After identifying the required communication segments depicted in Fig. 6.3, a complete communication architecture can be synthesized e.g. as multiple bus connections. It can be noticed, that the communications $PN_1 \rightarrow PN_2$ and $PN_1 \rightarrow PN_3$ are serialized by the mapping of PN_3 and PN_2 on the same resource (see Fig. 6.3), therefore they can share the same communication link. In contrast, the communication $PN_1 \rightarrow PN_5$ should be performed in parallel with $PN_1 \rightarrow PN_3$, as indicated by the scheduling times of PN_3 and PN_5 from Tab. 6.3, thus a separate link connecting GPP_2 with MEM_1 must be provided. In addition, PN_6 starts after PN_7 finishes, as indicated by the timing values in Tab. 6.3, therefore the communications $PN_3 \rightarrow PN_7$ and $PN_2 \rightarrow PN_6$ can be serialized on the same bus shared by GPP_2, GPP_3, and MEM_2. Similarly, the communications $PN_7 \rightarrow PN_9$, $PN_5 \rightarrow PN_{14}$, and $PN_{14} \rightarrow PN_{15}$ can be serialized on a single communication link. Thus, the communication segments 1, 4, 5, 6, 7, 9, which belong to the same floorplan cluster FC_0 and exhibit therefore comparable delays, can be shared by a global bus. The local segments 2 and 8, which connect the ASIC with MEM_3 and GPP_2 with MEM_4 are to be implemented due to their small delay by individual point-to-point links. The identified

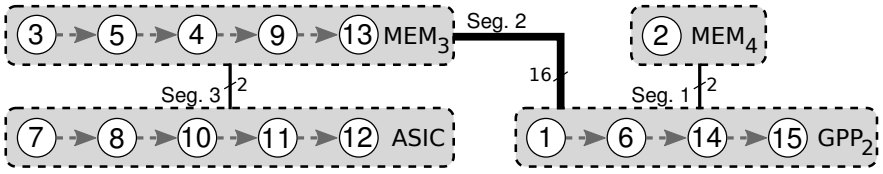

Fig. 6.5: Architecture optimized for minimum energy consumption, requiring only four resources and three communication segments.

parallel connection between GPP_2 and MEM_1 is implemented as a separate link.

The communication architecture synthesized in this way is presented in Fig. 6.4, where each communication link is denoted by its maximum delay. Note, that the association of several communication segments into a shared bus was possible due to their shared floorplan cluster and the propagation delay over the bus has been selected as the maximum delay of the unified communication segments. The corresponding scheduling lists of the communication tasks are given in Tab. 6.5.

6.2.2 Energy-Optimized Architecture

A synthesis oriented on minimizing the energy consumption with a strong emphasis on leakage energy has been performed by setting the weighting factors in the cost function to the following values:

$$w_T = 0.10 \tag{6.3}$$

$$w_{E_l} = 0.70 \tag{6.4}$$

$$w_{E_d} = 0.20 \tag{6.5}$$

This set of weights amplifies the contribution of leakage energy to the overall system cost while also considering to a small extent improvements in delay and dynamic energy. Guided by these values, the exploration algorithm found the solution shown in Fig. 6.5, which illustrates the resource mappings, scheduling lists, and the resulting communication segments. It is to be noticed that the 15 PNs have been mapped to only four of the eight available resources in this energy-optimized configuration. As a consequence, the remaining two GPP cores and the two unused memory blocks can be turned off e.g. using power gating, which reduces significantly the total energy consumption. In addition, most processing nodes are executed on the ASIC block, which has a much lower power dissipation than the GPPs (see Tab. 6.1). As a result of this energy-driven optimization, the system achieves a total leakage energy consumption of only 0.965 mJ, a moderate dynamic energy of 2.505 mJ, and a total delay of 10.191 ms (all values evaluated as the 99% inferior quantile from their respective distributions). It is to be mentioned that the leakage energy consumption is improved with respect to the previous delay-optimized architecture by a factor of 3.6, at the expense of a moderate increase in delay by a factor of approx. 1.09.

Comm. Seg.	Delay [ps]	Leak. Power [nW]	Dyn. Power [nW]	Signaling Circuit	V_{dd} [V]	V_{bs} [V]
Seg. 1	68	8.402	10.284	VM	1.0	-0.4
Seg. 2	372	45.005	92.486	PCM	0.6	-0.4
Seg. 3	67	9.241	41.165	VM	1.0	-0.4

Tab. 6.6: Parameters of the three synthesized communication segments shown in Fig. 6.5 (evaluated using the 99% inferior quantile from the statistical distributions).

The performance metrics for the synthesized inter-resource communication segments are presented in Tab. 6.6, together with the signaling circuit parameters. Voltage-mode signaling is again employed on the short segments connecting the ASIC with MEM_3 and the GPP with MEM_4, as this signaling method is both energy-efficient and also delay-efficient at very short interconnect lengths. A reverse body bias of -0.4 V decreases significantly the leakage energy on all segments, as expected from the results presented in Sec. 4.2.4 and 4.3.3. In addition, as discussed in Sec. 4.2.4, the reverse body bias on the PCM line does not significantly affect the delay. Thus, the choice of PCM signaling is optimal for the long communication segment, as it allows the leakage reduction through body biasing while keeping the delay on a low level. In contrast, voltage-mode signaling is recommended on the short segments since it exhibits a lower leakage energy, lower delay, and the delay increase due to the reverse body bias remains relatively small in the context of the overall system latency.

Scaling the supply voltage on the PCM line to the value of 0.6 V improves the dynamic energy consumption due to the large parasitic capacitance of the global segment. The leakage energy consumption is less influenced by this scaling (particularly at -0.4 V reverse body bias, see also Fig. 4.20(a)) and the delay of PCM lines, as discussed in Sec. 4.2.4, remains practically unaffected. It is important to notice, that voltage scaling on the VM lines has been rejected by the exploration algorithm, since it brings a negligible improvement on the dynamic energy due to the very small line capacitance, at the cost of a significant delay increase, since the delay of voltage-mode lines is sensitive to voltage scaling (see Fig. 4.25(a)).

6.2.3 Accuracy Evaluation

Explorations in the solution space use the developed circuit-level models and RLC line models [116] for fast estimations. The performance metrics evaluated in this way serve mainly to compare the cost functions of different solutions in the search for an optimized architecture. Nevertheless, it is useful to investigate the accuracy of this modeling approach in predicting the actual performance level of the synthesized architecture.

In order to check the modeling accuracy, two test scenarios are defined. First, the circuit-level modeling precision is tested by comparing the segment models employed in the optimization with simulations of the signaling circuits using the Cadence Spectre [31]

Comm. Seg.	RLC Line Model	Extrapolated S-Parameter Line Model
Delay-Optimized Architecture		
Seg. 1	2.882 %	10.618 %
Seg. 2	2.806 %	4.975 %
Seg. 3	2.477 %	9.894 %
Seg. 4	3.823 %	10.551 %
Seg. 5	2.872 %	10.379 %
Seg. 6	2.842 %	10.160 %
Seg. 7	2.068 %	11.082 %
Seg. 8	3.185 %	4.328 %
Seg. 9	3.912 %	10.510 %
Energy-Optimized Architecture		
Seg. 1	3.137%	5.291%
Seg. 2	3.751%	11.572%
Seg. 3	2.943%	4.345%

Tab. 6.7: Relative delay error of the communication circuit models with respect to circuit simulations.

circuit simulator. The relative delay error, evaluated as:

$$\varepsilon_{d,rel} = \frac{|T_{d,model} - T_{d,sim}|}{T_{d,sim}} \cdot 100\ [\%] \tag{6.6}$$

(where $T_{d,model}$ and $T_{d,sim}$ are the segment delays obtained using the model and circuit simulation, respectively) is given in the second column of Tab. 6.7 for every communication segment synthesized in Sec. 6.2.1 and 6.2.2. It is to be noted that this first test scenario uses the same RLC line model from [116] in the segment models and within the circuit simulations. Thus, the results from Tab. 6.7 using the RLC line model practically reflect the accuracy of the signaling circuit models. As a result of employing technology-accurate BSIM4 equations and parameters in the circuit models and due to the direct analytical solution to the equivalent circuit model, the results show a very good agreement with the circuit simulations (which employ the commercial BSIM4 implementation and the same technology parameters).

In the second test scenario, the communication segments are described using the refined interconnect models developed in chapter 5. Within this context, the wide-bandwidth interconnect models are employed in a practical application scenario to validate the synthesis results from the optimization framework. The synthesized segments are simulated first using the extrapolated S-parameter models, then the results are compared with simulations of similar S-parameter models extracted with the field solver. The relative error results are given in the last column of Tab. 6.7 and show an accuracy level similar to the results from chapter 5 for this practical example. It is important to add that shorter segments have only two parallel wires (see Fig. 6.3 and Fig. 6.5) and therefore exhibit a lower

relative error. As indicated in Sec. 5.4, the modeling error increases generally with the number of wires per segment.

6.3 Summary

This chapter has presented the results of applying the developed communication synthesis framework in the context of a practical example. An application scenario has been chosen and the description of the corresponding profile considering the target MPSoC architecture has been presented. To include the influence of process parameter variations, the chip area has been divided into a two-dimensional grid and a spatial correlation model has been applied. In addition, floorplanning information has been considered using a cluster tree model description.

Further, the exploration method for optimizing the communication architecture has been discussed and the steps performed in the solution space have been explained. An individual probability value has been assigned to each step type to increase the relative frequency of optimizations at the circuit level, such as signaling choice, voltage scaling, and body biasing. It has also been shown that the relative influence of performance metrics on the system cost function has to be normalized and balanced using weighting factors to achieve different optimization goals.

Afterwards, the synthesis results have been evaluated for two optimization scenarios. The delay-optimized architecture produced a relatively large number of communication segments at nominal supply voltage and body bias values. The choice of signaling circuits and voltages has been discussed and a communication architecture has been synthesized by unifying similar segments and scheduling communication activities. Additionally, an energy-optimized communication architecture has been synthesized by changing the cost weighting factors and the obtained parameters have been analyzed. The optimum choice of PCM and voltage-mode signaling techniques for long, respectively short communication lines has been pointed out once more and the choice of voltage values for minimizing the energy has been discussed. Finally, the modeling accuracy for the synthesized communication segments has been evaluated in comparison with circuit-level simulations using both the fast lumped RLC line models and the extrapolated interconnect model developed in this work.

Chapter 7

Conclusions

Contents

This thesis has presented a unified methodology for the modeling and optimization of on-chip communication, with an emphasis on technology accuracy and parameter variations. By developing a statistical parameter model and a method for the fast propagation of statistical distributions across algebraic operations, the development of variability-aware performance macromodels has been enabled. Consequently, optimization methods for system-level design decisions such as task mapping and scheduling have been analyzed in the context of variability and the particularities of result interpretation from statistical distributions have been explored. Technology accuracy in the communication models has been achieved by developing low-level analytic models for signaling circuits starting from an accurate statistical transistor-level model and by providing a wide-bandwidth modeling methodology for interconnection segments. Thus, the optimization of the communication architecture through circuit-level decisions such as the choice of signaling method, voltage scaling, and body biasing was enabled and analyzed.

7.1 Contributions of the Work

The proposed communication modeling and synthesis methodology has a strong focus on parameter variability. Thus, a structured method for extracting the application profile and for specifying the target MPSoC architecture considering parameter variations has been presented. Within this context, a unified application profile interface has been defined and implemented in the form of entries in a configuration file, which specify data dependencies, communication loads, SoC resource types, leakage and dynamic power consumption, task-resource compatibilities, execution times, floorplanning information,

and technology characteristics, such as process parameter values and variations. To enable the accurate description of variations in power dissipation, execution times, communication loads, as well as temperature and process parameters, a discretized distribution model with adjustable accuracy has been developed. The control of accuracy using limits of the approximation interval, number of pdf bins, and number of samples, as well as a method to extract samples from the discretized description have been also presented. An adjustable tradeoff between accuracy and speed, as well as the ability to represent non-standard distributions and profiling data are the main advantages of the proposed model.

A novel method for the propagation of complete distribution functions has been introduced, relying upon the implementation of statistical operators. Unlike previous statistical analysis approaches which derive variability dependencies as linear sensitivities, moments, or polynomial approximations, this methodology propagates the entire pdf description across the model expressions by applying statistical operators. Within this context, the complex implementation of a statistical product operator has been described in detail. In addition, a novel technique for the fast numerical implementation of statistical operators with adjustable accuracy has been developed. It has been shown that a very good accuracy can be obtained using only 0.25 % of the execution time required by Monte Carlo sampling. This technique enabled the implementation of every statistical operation required by the models, such as exponential, square root, logarithm, division etc. Based upon this methodology, statistical macromodels for delay and power have been developed and presented. The inclusion of statistical representations in the operational nodes of the macromodels has been discussed and the implied re-evaluations and updates propagated across the tree structures during the optimization-driven changes in the macromodel structure have been analyzed. In addition, the interpretation of performance costs for optimization decisions using quantile functions applied to the statistical distributions has been presented.

An accurate current-source transistor model for technology-accurate estimations has been derived from BSIM4 equations, in which the modeling expressions are performed statistically, using the developed statistical operators and distribution propagation technique. In this way, state-of-the-art CMOS process parameters and parameter variations are included in the model to achieve a high technology accuracy. Spatially-correlated intra-die parameter variations were described using two-dimensional grids and correlation decay models. In addition, spatial correlations have been modeled using the principal component analysis. Based upon the statistical transistor model, equivalent circuit models have been derived for pulsed current-mode signaling circuits and for traditional voltage-mode signaling buffers. Using analytic approximations for the signal waveforms in the regions of interest, the circuit equations have been solved analytically and models for the delay and energy consumption have been derived. Using these circuit-level models, the performance of the investigated signaling methods has been investigated under the influence of voltage scaling and body biasing. Furthermore, the circuit-level mod-

els have been employed for the representation of communication segments within the optimization framework and design decisions such as the selection of signaling method, voltage scaling, and body biasing have been included in the solution space exploration.

For accurate validations of the synthesized communication segments, a computationally-efficient wide-bandwidth characterization method based on an incremental extrapolation of S-parameters for arbitrary interconnect structures has been developed and presented. The characterization method defines a systematic set of *a priori* parameter extractions and performs on-demand multistep extrapolations for interconnect segments with specified wire length, widths, spacings, metal layer, and neighboring routing information. The experimental evaluations have shown a maximum absolute error of less than $2 \cdot 10^{-2}$ (magnitude) and 7 degrees (angle) between the developed model and an industry-standard full-wave field simulator for a 90-nm CMOS process. Circuit-level simulations with the extrapolated model have shown a maximum signal delay error of less than 12.5 % across multiple metal layers and wire configurations.

Finally, the complete developed methodology has been employed on an application example and the communication synthesis has been conducted first for speed optimization and then for minimum energy consumption. In each case, the synthesized architecture and the parameters of the communication segments have been analyzed in detail. An evaluation of the modeling accuracy has shown an error level of less than 4 % using lumped RLC line models and less than 12 % using the extrapolated interconnect model.

7.2 Directions for Future Work

The proposed performance modeling methodology represents an important backbone for a variability-aware accurate communication synthesis framework. Several optimization resources have been enabled by the developed circuit-level models. Nevertheless, several directions for future enhancements can be identified.

Additional optimization techniques at the circuit level: The described methodology can be extended to include device sizing, wire width sizing, and wire spacing as optimization techniques. Since the developed models are already dependent on these parameters, such extensions can be easily integrated in the current framework.

Complete integration with SPICE-level circuit simulators: A direct synthesis of the communication segments into SPICE-level netlists is possible, since the signaling circuits and the interconnect line parasitics are known in the macromodels. This enhancement would directly enable the automatic validation of the synthesized segments and the replacement of RLC lumped parameters with the extrapolated n-port segment models. Simulation results can be read back into the framework and additional refinements can be applied to the synthesized architecture.

Inclusion of floorplanning algorithms: The current implementation of the synthesis framework considers a limited floorplanning information in the form of cluster trees and cluster descriptions. A complete floorplanning algorithm could be included in the framework to optimize the MPSoC floorplan in conjunction with the communication synthesis.

Routing of communication segments: A routing algorithm oriented on the manufacturing process could generate a complete wiring structure for the synthesized communication architecture which can be further optimized across the metal layers and can adjust the wire thickness, wire spacing, and routing configurations in the adjacent metal layers. In order to not interfere with the existing interconnects of the resources on the chip, a three-dimensional constraint system should be implemented to delimit the allowed space for routing.

Appendix A

Complex Expression of the Output Voltage for the Voltage-Mode Signaling Circuit

Depending on device parameters and interconnect sizes, the roots of the characteristic equation (4.85) might be complex, such that:

$$\lambda_1 = \Re_\lambda + j\Im_\lambda \tag{A.1}$$

$$\lambda_2 = \Re_\lambda - j\Im_\lambda \tag{A.2}$$

As a consequence, the output voltage $V_o(t)$ has also a complex expression. In such a case, the delay model considers only the real part of V_o, which after applying the computations described in Sec. 4.3.2 results in the following form:

$$\begin{aligned}\Re_{V_o(t)} =& V_{dd} + \frac{1}{L\alpha C_1 C_2 \left(\Im_\lambda^2 + (\alpha - \Re_\lambda)^2\right)\left(\Im_\lambda^2 + \Re_\lambda^2\right)} \left\{-t\alpha I_1 \left(\Im_\lambda^2 + (\alpha - \Re_\lambda)^2\right)\right. \\ &+ \left(-1 + e^{\alpha t}\right)\gamma\left(\Im_\lambda^2 + \Re_\lambda^2\right) + L\alpha C_1\left(\Im_\lambda^2 + (\alpha - \Re_\lambda)^2\right)\left[\Im_{K_1}\Im_\lambda - \Im_{K_2}\Im_\lambda\right. \\ &+ \Re_{K_1}\Re_\lambda + \Re_{K_2}\Re_\lambda - e^{\Re_\lambda t}\sin(\Im_\lambda t)\left(\Im_\lambda\left(\Re_{K_1} + \Re_{K_2}\right) + \left(-\Im_{K_1} + \Im_{K_2}\right)\Re_\lambda\right) \\ &\left.\left.-e^{\Re_\lambda t}\cos(\Im_\lambda t)\left(\left(\Im_{K_1} - \Im_{K_2}\right)\Im_\lambda + \left(\Re_{K_1} + \Re_{K_2}\right)\Re_\lambda\right)\right]\right\}\end{aligned} \tag{A.3}$$

where the constants K_1 and K_2 have complex expressions given by:

$$\Re_{K_1} = \frac{\gamma}{2LC_1\left(\Im_\lambda^2 + (\alpha + \Re_\lambda)^2\right)} - \frac{I_1}{2LC_1\left(\Im_\lambda^2 + \Re_\lambda^2\right)} \tag{A.4}$$

$$\Im_{K_1} = -\frac{\alpha\gamma}{2LC_1\Im_\lambda\left(\Im_\lambda^2 + (\alpha + \Re_\lambda)^2\right)} + \frac{\gamma\Re_\lambda}{2LC_1\Im_\lambda\left(\Im_\lambda^2 + (\alpha - \Re_\lambda)^2\right)} - \frac{I_1\Re_\lambda}{2LC_1\Im_\lambda\left(\Im_\lambda^2\Re_\lambda^2\right)} \tag{A.5}$$

$$\Re_{K_2} = -\Re_{K_1} + \frac{\gamma\left(\alpha^2 + \Im_\lambda^2 - 2\alpha\Re_\lambda + \Re_\lambda^2\right)}{LC_1\left(\Im_\lambda^2 + (\alpha - \Re_\lambda)^2\right)^2} - \frac{I_1}{LC_1\left(\Im_\lambda^2 + \Re_\lambda^2\right)} \tag{A.6}$$

$$\Im_{K_2} = -\Im_{K_1} \tag{A.7}$$

References

[1] A. AGARWAL, D. BLAAUW, and V. ZOLOTOV. Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 900–907, 2003.

[2] A. AGARWAL, D. BLAAUW, V. ZOLOTOV, S. SUNDARESWARAN, M. ZHAO, K. GALA, and R. PANDA. Statistical Delay Computation Considering Spatial Correlations. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 271–276, 2003.

[3] A. AGARWAL, D. BLAAUW, V. ZOLOTOV, and S. VRUDHULA. Statistical Timing Analysis using Bounds and Selective Enumeration. In *IEEE/ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, pages 29–36, 2002.

[4] A. AGARWAL, D. BLAAUW, V. ZOLOTOV, and S. VRUDHULA. Computation and Refinement of Statistical Bounds on Circuit Delay. In *Design Automation Conf. (DAC)*, pages 348–353, 2003.

[5] A. AGARWAL, D. BLAAUW, V. ZOLOTOV, and S. VRUDHULA. Statistical Timing Analysis using Bounds. In *Design Automation and Test in Europe (DATE)*, pages 62–67, 2003.

[6] A. AGARWAL, V. ZOLOTOV, and D. BLAAUW. Statistical Timing Analysis Using Bounds and Selective Enumeration. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 22(9):1243–1260, Sept. 2003.

[7] K. AGARWAL and S. NASSIF. Characterizing Process Variation in Nanometer CMOS. In *Design Automation Conf. (DAC)*, pages 396–399, 2007.

[8] K. AGARWAL, R. RAO, D. SYLVESTER, and R. BROWN. Parametric Yield Analysis and Optimization in Leakage Dominated Technologies. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 15(6):613–623, June 2007.

[9] I. AHSAN, N. ZAMDMER, O. GLUSHCHENKOV, R. LOGAN, E. J. NOWAK, H. KIMURA, J. ZIMMERMAN, G. BERG, J. HERMAN, E. MACIEJEWSKI, A. CHAN, A. AZUMA, S. DESHPANDE, B. DIRAHOUI, G. FREEMAN, A. GABOR, M. GRIBELYUK, S. HUANG, M. KUMAR, K. MIYAMOTO, D. MOCUTA, A. MAHOROWALA, E. LEOBANDUNG, H. UTOMO, and B. WALSH. RTA-Driven Intra-Die Variations in Stage Delay, and Parametric Sensitivities for 65nm Technology. In *IEEE Symp. on VLSI Technology*, pages 170–171, 2006.

[10] C. S. AMIN, N. MENEZES, K. KILLPACK, F. DARTU, U. CHOUDHURY, N. HAKIM, and Y. I. ISMAIL. Statistical Static Timing Analysis: How Simple Can We Get? In *Design Automation Conf. (DAC)*, pages 652–657, 2005.

[11] H. ANANTHAN and K. ROY. A Fully Physical Model for Leakage Distribution under Process Variations in Nanoscale Double-Gate CMOS. In *Design Automation Conf. (DAC)*, pages 413–418, 2006.

[12] M. ANDERS, N. RAI, R. K. KRISHNAMURTHY, and S. BORKAR. A Transition-Encoded Dynamic Bus Technique for High-Performance Interconnects. *IEEE Journal of Solid-State Circuits*, 38(5):709–714, May 2003.

[13] A. ANDREI, M. T. SCHMITZ, P. ELES, Z. PENG, and B. M. AL HASHIMI. Quasi-Static Voltage Scaling for Energy Minimization with Time Constraints. In *Design Automation and Test in Europe (DATE)*, pages 514–519, 2005.

[14] F. ANGIOLINI, J. CENG, R. LEUPERS, F. FERRARI, C. FERRI, and L. BENINI. An Integrated Open Framework for Heterogeneous MPSoC Design Space Exploration. In *Design Automation and Test in Europe (DATE)*, pages 1145–1150, 2006.

[15] ANSOFT CORP. HFSS: 3D Full-wave Electromagnetic Field Simulation. http://www.ansoft.com/products/hf/hfss, Sept. 2008.

[16] R. BASHIRULLAH, W. LIU, R. CAVIN, and D. EDWARDS. A Hybrid Current/Voltage Mode On-Chip Signaling Scheme With Adaptive Bandwidth Capability. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 12(8):876–880, Aug. 2004.

[17] R. BASHIRULLAH, W. LIU, R. CAVIN, and D. EDWARDS. A 16 Gb/s Adaptive Bandwidth On-Chip Bus Based on Hybrid Current/Voltage Mode Signaling. *IEEE Journal of Solid-State Circuits*, 41(2):461–473, Feb. 2006.

[18] R. BASHIRULLAH, W. LIU, and R. K. CAVIN. Current-Mode Signaling in Deep Submicrometer Global Interconnects. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 11(3):406–417, June 2003.

[19] M. BERKELAAR. Statistical Delay Calculation, a Linear Time Method. In *Intl. Workshop on Timing Issues (TAU)*, pages 15–24, 1997.

[20] K. BERNSTEIN, D. J. FRANK, A. E. GATTIKER, W. HAENSCH, B. L. JI, S. R. NASSIF, E. J. NOWAK, D. J. PEARSON, and N. J. ROHRER. High-performance CMOS variability in the 65-nm regime and beyond. *IBM J. Res. & Dev.*, 50(4/5):433–449, July/Sept. 2006.

[21] S. BHARDWAJ, Y. CAO, and S. VRUDHULA. Statistical Leakage Minimization Through Joint Selection of Gate Sizes, Gate Lengths and Threshold Voltage. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 953–958, 2006.

[22] S. BHARDWAJ, S. VRUDHULA, and D. BLAAUW. τAU: Timing Analysis Under Uncertainty. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 615–620, 2003.

[23] S. BHARDWAJ, S. VRUDHULA, P. GHANTA, and Y. CAO. Modeling of Intra-Die Process Variations for Accurate Analysis and Optimization of Nano-Scale Circuits. In *Design Automation Conf. (DAC)*, pages 791–796, 2006.

[24] D. BLAAUW, K. CHOPRA, A. SRIVASTAVA, and L. SCHEFFER. Statistical Timing Analysis: From Basic Principles to State of the Art. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 27(4):589–607, Apr. 2008.

[25] T. BLICKLE, J. TEICH, and L. THIELE. System-Level Synthesis Using Evolutionary Algorithms. *J. Des. Automation Embedded Syst.*, 3(1):23–58, 1998.

[26] A. BONNOIT, S. HERBERT, and D. M. L. PILEGGI. Integrating Dynamic Voltage/Frequency Scaling and Adaptive Body Biasing using Test-Time Voltage Selection. In *Intl. Symp. on Low Power Electronics and Design (ISLPED)*, pages 207–212, 2009.

[27] S. BORKAR, T. KARNIK, S. NARENDRA, J. TSCHANZ, A. KESHAVARZI, and V. DE. Parameter Variations and Impact on Circuits and Microarchitecture. In *Design Automation Conf. (DAC)*, pages 338–342, 2003.

[28] K. A. BOWMAN, B. L. AUSTIN, J. C. EBLE, X. TANG, and J. D. MEINDL. A Physical Alpha-Power Law MOSFET Model. *IEEE Journal of Solid-State Circuits*, 34(10):1410–1414, Oct. 1999.

[29] C.-P. CHEN and D. F. WONG. Optimal Wire Sizing Function with Fringing Capacitance Consideration. In *Design Automation Conf. (DAC)*, pages 604–607, 1997.

[30] CADENCE DESIGN SYSTEMS. Virtuoso Spectre Circuit Simulator User Guide. Product Version 5.1.41, July 2004.

[31] Cadence Design Systems, Inc. *Virtuoso Spectre Circuit Simulator Reference*, Nov. 2004. Product Version 5.1.41.

[32] A. E. CALDWELL, A. B. KAHNG, S. MANTIK, I. L. MARKOV, and A. ZELIKOVSKY. On Wirelength Estimations for Row-Based Placement. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 18(9):1265–1278, 1999.

[33] K. CAO, S. DOBRE, and J. HU. Standard Cell Characterization Considering Lithography Induced Variations. In *Design Automation Conf. (DAC)*, pages 801–804, 2006.

[34] Y. CAO and L. T. CLARK. Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach. In *Design Automation Conf. (DAC)*, pages 658–663, 2005.

[35] H. CHANG and S. SAPATNEKAR. Statistical Timing Analysis Considering Spatial Correlations using a Single Pert-Like Traversal. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 621–625, 2003.

[36] H. CHANG, V. ZOLOTOV, S. NARAYAN, and C. VISWESWARIAH. Parametrized Block-Based Statistical Timing Analysis with Non-Gaussian Parameters, Nonlinear Delay Functions. In *Design Automation Conf. (DAC)*, pages 71–76, 2005.

[37] R. T. CHANG, N. TALWALKAR, C. P. YUE, and S. S. WONG. Near Speed-of-Light Signaling Over On-Chip Electrical Interconnects. *IEEE Journal of Solid-State Circuits*, 38(5):834–838, May 2003.

[38] S. C. CHAPRA and R. P. CANALE. *Numerical Methods for Engineers*. McGraw-Hill Higher Education, 2006.

[39] K. CHEN and C. HU. Performance and Vdd Scaling in Deep Submicrometer CMOS. *IEEE Journal of Solid-State Circuits*, 33(10):1586–1589, 1998.

[40] M. CHEN, W. ZHAO, F. LIU, and Y. CAO. Fast Statistical Circuit Analysis with Finite-Point Based Transistor Model. In *Design Automation and Test in Europe (DATE)*, pages 1391–1396, 2007.

[41] T. CHEN and J. GREGG. A Low Cost Individual-Well Adaptive Body Bias (IWABB) Scheme for Leakage Power Reduction and Performance Enhancement in the Presence of Intra-Die Variations. In *Design Automation and Test in Europe (DATE)*, 2004.

[42] T. CHEN and S. NAFFZIGER. Comparison of Adaptive Body Bias (ABB) and Adaptive Supply Voltage (ASV) for Improving Delay and Leakage Under the Presence of Process Variation. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 11(5):888–899, Oct. 2003.

[43] C. CHO, D. KIM, J. KIM, J.-O. PLOUCHART, and R. TRZCINSKI. Statistical Framework for Technology-Model-Product Co-Design and Convergence. In *Design Automation Conf. (DAC)*, pages 503–508, 2007.

[44] S.-W. CHOI, H.-B. LEE, and H.-J. PARK. A Three-Data Differential Signaling Over Four Conductors With Pre-Emphasis and Equalization: A CMOS Current Mode Implementation. *IEEE Journal of Solid-State Circuits*, 41(3):633–641, Mar. 2006.

[45] K. CHOPRA, S. SHAH, A. SRIVASTAVA, D. BLAAUW, and D. SYLVESTER. Parametric Yield Maximization using Gate Sizing based on Efficient Statistical Powe and Delay Gradient Computation. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 1020–1025, 2005.

[46] K. CHOPRA, N. SHENOY, and D. BLAAUW. Variogram Based Robust Extraction of Process Variation Model. In *Intl. Workshop on Timing Issues (TAU)*, 2007.

[47] B. CLINE, K. CHOPRA, D. BLAAUW, A. TORRES, and S. SUNDARESWARAN. Transistor-Specific Delay Modeling for SSTA. In *Design Automation and Test in Europe (DATE)*, pages 592–597, 2008.

[48] J. CONG. An Interconnect-Centric Design Flow for Nanometer Technologies. *Proceedings of the IEEE*, 89:505–528, Apr. 2001.

[49] J. CONG and Z. PAN. Interconnect Performance Estimation Models for Design Planning. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 20(6):739–752, 2001.

[50] B. DAVE, G. LAKSHMINARAYANA, and N. JHA. COSYN: Hardware-Software Co-Synthesis of Embedded Systems. In *Design Automation Conf. (DAC)*, pages 703–708, 1997.

[51] B. DAVE, G. LAKSHMINARAYANA, and N. JHA. COSYN: Hardware-Software Co-Synthesis of Heterogeneous Distributed Embedded Systems. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 7(1):92–104, 1999.

[52] D. J. DELEGANES, M. BARANY, G. GEANNOPOULOS, K. KREITZER, M. MORRISE, D. MILLIRON, A. P. SINGH, and S. WIJERATNE. Low-Voltage Swing Logic Circuits for a Pentium 4 Processor Integer Core. *IEEE Journal of Solid-State Circuits*, 40(1):36–43, Jan. 2005.

[53] A. DEUTSCH, P. W. COTEUS, G. V. KOPCSAY, H. H. SMITH, C. W. SUROVIC, B. L. KRAUTER, D. C. EDELSTEIN, and P. J. RESTLE. On-Chip Wiring Design Challenges for Gigahertz Operation. *Proceedings of the IEEE*, 89(4):529–555, Apr. 2001.

[54] G. V. DEVARAYANADURG and M. SOMA. An interconnect model for arbitrary terminations based on scattering parameters. *Analog Integrated Circuits and Signal Processing*, 5:31–45, Jan. 1994.

[55] A. DEVGAN and C. KASHYAP. Block-Based Static Timing Analysis with Uncertainty. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 607–614, 2003.

[56] A. DOBOLI. Integrated Hardware-Software Co-Synthesis and High-Level Synthesis for Design of Embedded Systems under Power and Latency Constraints. In *Design Automation and Test in Europe (DATE)*, pages 612–619, Mar. 2001.

[57] A. DOBOLI and P. ELES. Scheduling under Control Dependencies for Heterogeneous Architectures. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 602–608, 1998.

[58] M. EISELE, J. BERTHOLD, D. SCHMITT-LANDSIEDEL, and R. MAHNKOPF. The Impact of Intra-Die Device Parameter Variations on Path Delays and on the Design for Yield of Low Voltage Digital Circuits. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 5(4):360–368, 1997.

[59] P. ELES, A. DOBOLI, P. POP, and Z. PENG. Scheduling with Bus Access Optimization for Distributed Embedded Systems. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 8(5):472–491, 2000.

[60] W. C. ELMORE. The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers. *J. Appl. Phys.*, 19(1):55–63, 1948.

[61] R. ERNST and W. YE. Embedded Program Timing Analysis Based on Path Clustering and Architecture Classification. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 598–604, 1997.

[62] D. GAJSKI, N. DUTT, C. WU, and Y. LIN. *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1991.

[63] D. GAJSKI, F. VAHID, S. NARAYAN, and J. CONG. *Specification and Design of Embedded Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1994.

[64] D. D. GAJSKI and F. VAHID. Specification and Design of Embedded Hardware-Software Systems. *IEEE Design & Test of Computers*, 12:53–67, 1995.

[65] M. GALASSI, J. DAVIES, J. THEILER, B. GOUGH, G. JUNGMAN, M. BOOTH, and F. ROSSI. *GNU Scientific Library Reference Manual*, 1.11 edition, Feb. 2008.

[66] M. R. GAREY and D. S. JOHNSON. *Computers and Intractability: a Guide to the Theory of NP Completeness*. W. H. Freeman & Co., 1979.

[67] A. GATTIKER, S. NASSIF, R. DINAKAR, and C. LONG. Timing Yield Estimation from Static Timing Analysis. In *Intl. Symp. on Quality Electronic Design (ISQED)*, pages 437–442, 2001.

[68] P. GHANTA, S. VRUDHULA, R. PANDA, and J. WANG. Stochastic Power Grid Analysis Considering Process Variations. In *Design Automation and Test in Europe (DATE)*, pages 964–969, 2005.

[69] A. G. GLEN, L. M. LEEMIS, and J. H. DREW. Computing the distribution of the product of two continuous random variables. *Computational Statistics & Data Analysis*, 44(3):451–464, Jan. 2004.

[70] J. GONG, D. GAJSKI, and S. NARAYAN. Software Estimation from Executable Specifications. In *Proc. European Design Automation Conf. (EuroDAC)*, 1995.

[71] J. GU, S. SAPATNEKAR, and C. KIM. Width-Dependent Statistical Leakage Modeling for Random Dopant Induced Threshold Voltage Shift. In *Design Automation Conf. (DAC)*, pages 87–92, 2007.

[72] B. GUSTAVSEN and A. SEMLYEN. Enforcing Passivity for Admittance Matrices Approximated by Rational Functions. *IEEE Trans. on Power Systems*, 16(1):97–104, Feb. 2001.

[73] S. HANSON, B. ZHAI, K. BERNSTEIN, D. BLAAUW, A. BRYANT, L. CHANG, K. K. DAS, W. HAENSCH, E. J. NOWAK, and D. M. SYLVESTER. Ultralow-Voltage Minimum-Energy CMOS. *IBM J. Res. & Dev.*, 50(4/5):469–490, July/Sept. 2006.

[74] W. HEINRICH, K. BEILENHOFF, P. MEZZANOTTE, and L. ROSELLI. Optimum Mesh Grading for Finite-Difference Method. *IEEE Trans. on Microwave Theory and Techniques*, 44(9):1569–1574, Sept. 1996.

[75] J. HENKEL. A Low Power Hardware/Software Partitioning Approach for Core-based Embedded Systems. In *Design Automation Conf. (DAC)*, pages 122–127, 1999.

[76] V. HILL, O. FARLE, and R. DYCZIJ-EDLINGER. A Stabilized Multilevel Vector Finite-Element Solver for Time-Harmonic Electromagnetic Waves. *IEEE Trans. on Microwave Theory and Techniques*, 39(3):1203–1206, May 2003.

[77] C. HOARE. Communicating Sequential Processes. *Comm. ACM*, 21(8):666–677, 1978.

[78] C.-C. HUANG. Using S parameters for signal integrity analysis. eeDesign (EE Times EDA News), Feb. 2004.

[79] E. HUMENAY, D. TARJAN, and K. SKADRON. Impact of Process Variations on Multicore Performance Symmetry. In *Design Automation and Test in Europe (DATE)*, pages 1653–1658, 2007.

[80] INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2006 UPDATE. Lithography. http://www.itrs.net, Feb.

[81] INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2007 EDITION. Interconnect. http://www.itrs.net, Feb. 2008.

[82] INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2008 UPDATE. Design. http://www.itrs.net, May 2009.

[83] N. IZUMI, H. OZAKI, Y. NAKAGAWA, N. KASAI, and T. ARIKADO. Evaluation of Transistor Property Variations Within Chips on 300-nm Wafers Using a New MOSFET Array Test Structure. *IEEE Trans. on Semiconductor Manufacturing*, 17(3):248–254, Aug. 2004.

[84] S. JAHN, M. MARGRAF, V. HABCHI, and R. JACOB. The Qucs Project: Technical Papers. http://qucs.sourceforge.net/tech/technical.html, Nov. 2008.

[85] J. A. G. JESS, K. KALAFALA, S. R. NAIDU, R. H. J. M. OTTEN, and C. VISWESWARIAH. Statistical Timing for Parametric Yield Prediction of Digital Integrated Circuits. In *Design Automation Conf. (DAC)*, pages 932–937, 2003.

[86] D. JIAO, M. MAZUMDER, S. CHAKRAVARTY, C. DAI, M. KOBRINSKY, M. HARMES, and S. LIST. A Novel Technique for Full-Wave Modeling of Large-Scale Three-Dimensional High-Speed On/Off-Chip Interconnect Structures. In *International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 39–42, Sept. 2003.

[87] A. P. JOSE, G. PATOUNAKIS, and K. L. SHEPARD. Pulsed Current-Mode Signaling for Nearly Speed-of-Light Intrachip Communication. *IEEE Journal of Solid-State Circuits*, 41(4):772–780, Apr. 2006.

[88] M. KAMON, M. TSUK, and J. WHITE. FastHenry: A Multipole-Accelerated 3D Inductance Extraction Program. *IEEE Trans. on Microwave Theory and Techniques*, 42(9):1750–1758, Sept. 1994.

[89] K. KANG, B. PAUL, and K. ROY. Statistical Timing Analysis Using Levelized Covariance Propagation. In *Design Automation and Test in Europe (DATE)*, pages 764–769, 2005.

[90] J. T. KAO, M. MIYAZAKI, and A. R. CHANDRAKASAN. A 175-mV multiply-accumulate unit using an adaptive supply voltage and body bias architecture. *IEEE Journal of Solid-State Circuits*, 37:1545–1554, Nov. 2002.

[91] H. KAUL, D. SYLVESTER, and D. BLAAUW. Performance Optimization of Critical Nets Through Active Shielding. *IEEE Trans. on Circuits and Systems I: Regular Papers*, 51(12):2417–2435, Dec. 2004.

[92] V. KHANDELWAL, A. DAVOODI, and A. SRIVASTAVA. Efficient Statistical Timing Analysis Through Error Budgeting. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 473–477, 2004.

[93] J. KIM and M. ORSHANSKY. Towards Formal Probabilistic Power-Performance Design Space Exploration. In *Great Lakes Symp. on VLSI (GLSVLSI)*, pages 229–234, 2006.

[94] K. J. KUHN. Reducing Variation in Advanced Logic Technologies: Approaches to Process and Design for Manufacturability of Nanoscale CMOS. In *IEEE Intl. Electron Devices Meeting (IEDM)*, pages 471–474, Dec. 2007.

[95] M. KURIHARA, M. IZAWA, J. TANAKA, K. KAWAI, and N. FUJIWARA. Gate CD Control Considering Variation of Gate and STI Structure. *IEEE Trans. on Semiconductor Manufacturing*, 20(3):232–238, Aug. 2007.

[96] J.-Y. LAI, N. SAKA, and J.-H. CHUN. Evolution of Copper-Oxide Damascene Structures in Chemical Mechanical Polishing. *J. of the Electrochemical Society*, 149(1):G41–G50, 2002.

[97] J. LE, X. LI, and L. PILEGGI. STAC: Statistical Timing Analysis with Correlation. In *Design Automation Conf. (DAC)*, pages 343–348, 2004.

[98] Y. S. LI, S. MALIK, and A. WOLFE. Performance Estimation of Embedded Software with Instruction Cache Modeling. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 380–387, 1995.

[99] J.-J. LIOU, K.-T. CHENG, S. KUNDU, and A. KRISTIĆ. Fast Statistical Timing Analysis By Probabilistic Event Propagation. In *Design Automation Conf. (DAC)*, pages 661–666, 2001.

[100] J.-J. LIOU, A. KRISTIĆ, L.-C-WANG, and K.-T. CHENG. False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation. In *Design Automation Conf. (DAC)*, pages 566–569, 2002.

[101] F. LIU. A General Framework for Spatial Correlation Modeling in VLSI Design. In *Design Automation Conf. (DAC)*, pages 817–822, 2007.

[102] M. LOGHI, F. ANGIOLINI, D. BERTOZZI, L. BENINI, and R. ZAFALON. Analyzing On-Chip Communication in a MPSoC Environment. In *Design Automation and Test in Europe (DATE)*, pages 752–757, 2004.

[103] J. LOYER. S-parameters and digital-circuit design. EDN Magazine, Feb. 2003.

[104] J. LUO, S. SINHA, Q. SU, J.+KAWA, and C. CHIANG. An IC Manufacturing Yield Model Considering Intra-Die Variations. In *Design Automation Conf. (DAC)*, pages 749–754, 2006.

[105] M. MANI, A. DEVGAN, and M. ORSHANSKY. An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints. In *Design Automation Conf. (DAC)*, pages 309–314, 2005.

[106] G. MARSAGLIA. Ratios of Normal Variables and Ratios of Sums of Uniform Variables. *Journal of the American Statistical Association*, 60(309):193–204, Mar. 1965.

[107] H. MASUDA, S. OKAWA, and M. AOKI. Approach for Physical Design in Sub-100nm Era. In *Intl. Symp. on Circuits and Systems (ISCAS)*, volume 6, pages 5934–5937, 2005.

[108] J. W. MCPHERSON. Reliability Challenges for 45nm and Beyond. In *Design Automation Conf. (DAC)*, pages 176–181, 2006.

[109] A. V. MEZHIBA and E. G. FRIEDMAN. Properties of On-Chip Inductive Current Loops. In *Great Lakes Symp. on VLSI (GLSVLSI)*, pages 12–17, Apr. 2002.

[110] M. MIYAZAKI, G. ONO, and K. ISHIBASHI. A 1.2-GIPS/W Microprocessor using Speed-Adaptive Threshold-Voltage CMOS with Forward Bias. *IEEE Journal of Solid-State Circuits*, 37:210–217, Feb. 2002.

[111] F. MOLL and M. ROCA. *Interconnection Noise in VLSI Circuits*. Kluwer, Dordrecht, The Netherlands, 2004.

[112] K. NABORS and J. WHITE. FastCap: A Multipole-Accelerated 3D Capacitance Extraction Program. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 21(11):50–62, Nov. 1991.

[113] S. R. NAIDU. Timing Yield Calculation Using an Impulse-Train Approach. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 219–224, 2002.

[114] K. NAISHADHAM and P. MISRA. Order Recursive Method of Moments (ORMoM) for Iterative Design Applications. *IEEE Trans. on Microwave Theory and Techniques*, 44(12):2595–2604, Dec. 1996.

[115] O. S. NAKAGAWA, N. CHANG, S. LIN, and D. SYLVESTER. Circuit Impact and Skew-Corner Analysis of Stochastic Process Variation in Global Interconnect. In *IEEE Intl. Conf. on Interconnect Technology*, pages 230–232, 1999.

[116] NANOSCALE INTEGRATION AND MODELING GROUP ASU. Predictive Technology Model. http://www.eas.asu.edu˜ptm, Sept. 2008.

[117] S. NARENDRA, A. KESHAVARZI, B. A. BLOECHEL, S. BORKAR, and V. DE. Forward Body Bias for Microprocessors in 130-nm Technology Generation and Beyond. *IEEE Journal of Solid-State Circuits*, 38:696–701, May 2003.

[118] S. R. NASSIF. Design for Variability in DSM technologies. In *Proc. IEEE ISQED*, pages 451–454, Mar. 2000.

[119] M. OLIVIERI, G. SCOTTI, and A. TRIFILETTI. A Novel Yield Optimization Technique for Digital CMOS Circuits Design by Means of Process Parameters Run-Time Estimation and Body Bias Active Control. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 13:630–638, May 2005.

[120] M. ORSHANSKY, J. C. CHEN, and C. HU. Direct Sampling Methodology for Statistical Analysis of Scaled CMOS Technologies. *IEEE Trans. on Semiconductor Manufacturing*, 12(4):403–408, Nov. 1999.

[121] M. ORSHANSKY, L. MILOR, P. CHEN, K. KEUTZER, and C. HU. Impact of Spatial Intrachip Gate Length Variability on the Performance of High-Speed Digital Circuits. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 21(5):544–553, 2002.

[122] M. ORSHANSKY, L. MILOR, and C. HU. Characterization of Spatial Intrafield Gate CD Variability, Its Impact on Circuit Performance, and Spatial Mask-Level Correction. *IEEE Trans. on Semiconductor Manufacturing*, 17(1):2–11, Feb. 2004.

[123] D. PAMUNUWA. *Modelling and Analysis of Interconnects for Deep Submicron Systems-on-Chip*. PhD thesis, Royal Inst. of Technology, Stockholm, Sweden, 2003.

[124] S. PANDEY and R. DRECHSLER. Robust On-Chip Bus Architecture Synthesis for MPSoCs Under Random Tasks Arrival. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 601–606, Mar. 2008.

[125] S. PANDEY, R. DRECHSLER, T. MURGAN, and M. GLESNER. Process Variations Aware Robust On-Chip Bus Architecture Synthesis for MPSoCs. In *Intl. Symp. on Circuits and Systems (ISCAS)*, pages 2989–2992, May 2008.

[126] S. PANDEY and M. GLESNER. Statistical on-chip communication bus synthesis and voltage scaling under timing yield constraint. In *Design Automation Conf. (DAC)*, pages 663–668, 2006.

[127] S. PANDEY and M. GLESNER. Simultaneous On-Chip Bus Synthesis and Voltage Scaling Under Random On-Chip Data Traffic. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 15(10):1111–1124, Oct. 2007.

[128] S. PANDEY, N. UTLU, and M. GLESNER. Tabu Search Based On-Chip Communication Bus Synthesis for Shared Multi-Bus Based Architecture. In *IFIP Intl. Conf. on VLSI-SoC*, pages 222–227, 2006.

[129] A. S. PAPA and M. MUTYAM. Power Management of Variation Aware Chip Multiprocessors. In *Great Lakes Symp. on VLSI (GLSVLSI)*, pages 423–428, 2008.

[130] S. PASRICHA, N. DUTT, E. BOZORGZADEH, and M. BEN-ROMDHANE. Floorplan-Aware Automated Synthesis of Bus-based Communication Architectures. In *Design Automation Conf. (DAC)*, pages 565–570, June 2005.

[131] T. PHAM-GIA, N. TURKKAN, and E. MARCHAND. Density of the Ratio of Two Normal Random Variables and Applications. *Communications in Statistics - Theory and Methods*, 35(9):1569–1591, Sept. 2006.

[132] S. PRAKASH and A. PARKER. SOS: Synthesis of application-specific heterogeneous multiprocessor systems. *J. Parallel Distrib. Comput.*, 16:338–351, 1992.

[133] R. R. RAO, A. DEVGAN, D. BLAAUW, and D. SYLVESTER. Anaytical Yield Prediction Considering Leakage/Performance Correlation. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 25(9):1685–1695, Sept. 2006.

[134] J. M. RABAEY, A. CHANDRAKASAN, and B. NIKOLIĆ. *Digital Integrated Circuits. A Design Perspective*. Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 2003.

[135] R. R. RAO, D. BLAAUW, and D. SYLVESTER. Modeling and Analysis of Parametric Yield under Power and Performance Constraints. *IEEE Design & Test of Computers*, 22(4):376–385, July 2005.

[136] J. RUBINSTEIN, P. PENFIELD JR., and M. A. HOROWITZ. Signal Delay in RC Tree Networks. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 2(3):202–211, 1983.

[137] A. E. RUEHLI and A. C. CANGELLARIS. Progress in the Methodologies for the Electrical Modeling of Interconnects and Electronic Packages. *Proceedings of the IEEE*, 89(5):740–771, May 2001.

[138] A. E. RUEHLI and H. HEEB. Challenges and Advances in Electrical Interconnect Analysis. In *Design Automation Conf. (DAC)*, pages 460–465, Anaheim, California, June 1992.

[139] T. SAKURAI and A. R. NEWTON. Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, 1990.

[140] H. SATO, H. KUNITOMO, K. TSUNEMO, K. MORI, and H. MASUDA. Accurate Statistical Process Variation Analysis for 0.25-μm CMOS with Advanced TCAD Methodology. *IEEE Trans. on Semiconductor Manufacturing*, 11(4):575–582, Nov. 1998.

[141] S. SATO. Growing Importance of Fundamental Understanding of the Source of Process Variations. In *IEEE Intl. Conf. on Adv. Thermal Processing of Semiconductors (RTP)*, pages 5–9, 2006.

[142] L. SCHEFFER. Explicit Computation of Performance as a Function of Process Variation. In *IEEE/ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, pages 1–8, 2002.

[143] L. SCHEFFER. The Count of Monte Carlo. In *IEEE/ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, 2004.

[144] H. SHEN and F. PÉTROT. Novel Task Migration Framework on Configurable Heterogeneous MPSoC Platforms. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 733–738, 2009.

[145] K. L. SHEPARD and T. ZIAN. Return-Limited Inductances: A Practical Approach to On-Chip Inductance Extraction. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 19(4):425–436, Apr. 2000.

[146] Y. SHIN and K. CHOI. Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems. In *Design Automation Conf. (DAC)*, pages 134–139, 1999.

[147] A. K. SINGH, M. MANI, and M. ORSHANSKY. Statistical Technology Mapping for Parametric Yield. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 510–517, 2005.

[148] A. SRIVASTAVA, R. BAI, D. BLAAUW, and D. SYLVESTER. Modeling and Analysis of Leakage Power Considering Within-Die Process Variations. In *Intl. Symp. on Low Power Electronics and Design (ISLPED)*, pages 64–67, 2002.

[149] A. SRIVASTAVA, K. CHOPRA, S. SHAH, D. SYLVESTER, and D. BLAAUW. A Novel Approach to Perform Gate-Level Yield Analysis and Optimization Considering Correlated Variations in Power and Performance. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 27(2):272–285, 2008.

[150] A. SRIVASTAVA, S. SHAH, K. AGARWAL, D. SYLVESTER, D. BLAAUW, and S. DIRECTOR. Accurate and Efficient Gate-Level Parametric Yield Estimation Considering Correlated Variations in Leakage Power and Performance. In *Design Automation Conf. (DAC)*, pages 535–540, 2005.

[151] A. SRIVASTAVA, D. SYLVESTER, and D. BLAAUW. *Statistical Analysis and Optimization for VLSI: Timing and Power*. Springer, New York, USA, 2005.

[152] C. SVENSSON. Optimum Voltage Swing on On-Chip and Off-Chip Interconnect. *IEEE Journal of Solid-State Circuits*, 36(7):1108–1112, July 2001.

[153] S. TASIRAN and A. DEMIR. Smart Monte Carlo for Yield Estimation. In *Intl. Workshop on Timing Issues (TAU)*, 2006.

[154] R. TEODORESCU, J. NAKANO, A. TIWARI, and J. TORRELLAS. Mitigating Parameter Variation with Dynamic Fine-Grain Body Biasing. In *Intl. Symp. on Microarchitecture (MICRO)*, pages 27–39, 2007.

[155] N. THEPAYASUWAN and A. DOBOLI. Hardware-Software Co-Design of Resource Constrained Systems on a Chip. In *Intl. Conf. on Distributed Computing Systems Workshops (ICDCSW)*, pages 818–823, Mar. 2004.

[156] N. THEPAYASUWAN and A. DOBOLI. Layout Conscious Approach and Bus Architecture Synthesis for Hardware/Software Codesign of Systems on Chip Optimized for Speed. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 13(5):525–538, May 2005.

[157] R. O. TOPALOGLU and A. ORAILOGLU. Forward Discrete Probability Propagation Method for Device Performance Characterization under Process Variations. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 220–223, 2005.

[158] P. TRIVERIO, S. GRIVET-TALOCIA, M. S. NAKHALA, F. G. CANAVERO, and R. ACHAR. Stability, Causality, and Passivity in Electrical Interconnect Models. *IEEE Trans. on Advanced Packaging*, 30(4):795–808, Nov. 2007.

[159] J. W. TSCHANZ, J. T. KAO, S. T. NARENDRA, R. NAIR, D. A. ANTONIADIS, A. P. CHANDRAKASAN, and V. DE. Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage. *IEEE Journal of Solid-State Circuits*, 37(11):1396–1402, Nov. 2002.

[160] S. TSUKIYAMA, M. TANAKA, and M. FUKUI. A Statistical Static Timing Analysis Considering Correlations Between Delays. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, pages 353–358, 2001.

[161] N. TZARTZANIS and W. W. WALKER. Differential Current-Mode Sensing for Efficient On-Chip Global Signaling. *IEEE Journal of Solid-State Circuits*, 40(11):2141–2147, Nov. 2005.

[162] E. B. VAN DER TOL, E. G. JASPERS, and R. H. GELDERBLOM. Mapping of H.264 Decoding on a Multiprocessor Architecture. In *Image and Video Communications and Processing*, pages 707–718, May 2003.

[163] J. WANG, P. GHANTA, and S. VRUDHULA. Stochastic Analysis of Interconnect Performance in the Presence of Process Variations. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 880–886, 2004.

[164] W.-S. WANG and M. ORSHANSKY. Robust Estimation of Parametric Yield under Limited Descriptions of Uncertainty. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 884–890, 2006.

[165] J. WATTS, N. LU, C. BITTNER, S. GRUNDON, and J. OPPOLD. Modeling FET Variation within a Chip as a Function of Circuit Design and Layout Choices. In *Nanotech Workshop on Compact Modeling*, pages 87–92, 2005.

[166] S.-C. WONG, T. G.-Y. LEE, D.-J. MA, and C.-J. CHAO. An Empirical Three-Dimensional Crossover Capacitance Model for Multilevel Interconnect VLSI Circuits. *IEEE Trans. on Semiconductor Manufacturing*, 13(2):219–227, May 2000.

[167] X. XI, M. DUNGA, J. HE, W. LIU, K. M. CAO, X. JIN, J. J. OU, M. CHAN, A. M. NIKNEJAD, and C. HU. *BSIM4.3.0 MOSFET Model - User's Manual*. University of California, Berkeley, 2003.

[168] J. XIONG, K. TAM, and L. HE. Buffer Insertion Considering Process Variation. In *Design Automation and Test in Europe (DATE)*, pages 970–975, 2005.

[169] J. XIONG, V. ZOLOTOV, and L. HE. Robust Extraction of Spatial Correlation. *IEEE Trans. on Computer-Aided Design (CAD) of Integrated Circuits and Systems*, 26(4):619–631, Apr. 2007.

[170] L. YAN, J. LUO, and N. K. JHA. Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-Time Embedded Systems. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 30–37, 2003.

[171] X. YE, P. LI, and F. LIU. Practical Variation-Aware Interconnect Delay and Slew Analysis for Statistical Timing Verification. In *Intl. Conf. on Computer-Aided Design (ICCAD)*, 2006.

[172] G. YU, W. DONG, Z. FENG, and P. LI. A Framework for Accounting for Process Model Uncertainty in Statistical Static Timing Analysis. In *Design Automation Conf. (DAC)*, pages 829–834, 2007.

[173] P. YU, S. X. SHI, and D. Z. PAN. Process Variation Aware OPC with Variational Lithography Modeling. In *Design Automation Conf. (DAC)*, pages 785–790, 2006.

[174] L. ZHANG, J. SHAO, and C. C. CHEN. Non-Gaussian Statistical Parameter Modeling for SSTA with Confidence Interval Analysis. In *Intl. Symp. on Physical Design (ISPD)*, pages 33–38, 2006.

[175] M. ZHANG, M. OLBRICH, H. KINZELBACH, D. SEIDER, and E. BARKE. A Fast and Accurate Monte Carlo Method for Interconnect Variation. In *IEEE Intl. Conf. on Integrated Circuit Design and Technology (ICICDT)*, 2006.

[176] M. ZHANG, M. OLBRICH, D. SEIDER, M. FRERICHS, H. KINZELBACH, and E. BARKE. CMCal: An Accurate Analytical Approach for the Analysis of Process Variations with Non-Gaussian Parameters and Nonlinear Functions. In *Design Automation and Test in Europe (DATE)*, pages 243–248, 2007.

List of Publications

[177] P. B. BACINSCHI and M. GLESNER. Modeling and Design of Organic Transistor Circuits. Workshop of Analog Integrated Circuits, Kaiserslautern, Germany, Mar. 2006.

[178] P. B. BACINSCHI and M. GLESNER. A Multistep Extrapolated S-Parameter Model for Arbitrary On-Chip Interconnect Structures. In *IFIP/IEEE Intl. Conf. on VLSI-SoC*, Florianópolis, Brazil, Oct. 2009. Extended version accepted as book chapter to be published by Springer.

[179] P. B. BACINSCHI and M. GLESNER. Technology-Accurate Variability-Aware Performance Macromodels for On-Chip Communication Synthesis. In *IFIP/IEEE Intl. Conf. on VLSI-SoC, PhD Forum*, Florianópolis, Brazil, Oct. 2009.

[180] P. B. BACINSCHI and M. GLESNER. Variability-Aware Synthesis of On-Chip Communication Architectures. Workshop "Design of Future Reliable Systems from Unreliable Fabrics", Munich, Germany, July 2009.

[181] P. B. BACINSCHI, T. MURGAN, K. KOCH, and M. GLESNER. Process Variations Robust Design of Buffers and Schmitt Triggers for a Hierarchical DLL Timing Measurement Framework. In *edaWorkshop*, pages 47–52, Hannover, Germany, June 2007.

[182] P. B. BACINSCHI, T. MURGAN, K. KOCH, and M. GLESNER. Variability-Aware Design of CMOS Schmitt Triggers for On-Chip Timing Measurement Frameworks. Workshop "Analogschaltungen", Freiburg, Germany, Mar. 2007.

[183] P. B. BACINSCHI, T. MURGAN, K. KOCH, and M. GLESNER. An Analog On-Chip Adaptive Body Bias Calibration for Reducing Mismatches in Transistor Pairs. In *Design Automation and Test in Europe (DATE)*, pages 698–703, Munich, Germany, Mar. 2008.

[184] M. MOMENI, P. B. BACINSCHI, and M. GLESNER. Comparison of Opamp-Based and Comparator-Based Delta-Sigma Modulation. In *Design Automation and Test in Europe (DATE)*, pages 688–693, Munich, Germany, Mar. 2008.

[185] T. MURGAN, P. B. BACINSCHI, A. GARCÍA ORTIZ, and M. GLESNER. Partial Bus-Invert Bus Encoding Schemes for Low-Power DSP Systems Considering Inter-Wire Capacitance. In *Intl. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 169–180, Montpellier, France, Sept. 2006.

[186] T. MURGAN, P. B. BACINSCHI, and M. GLESNER. Exploiting the Bit-Level Correlation of DSP Signals for Low Power Coding Schemes Construction in Capacitively Coupled Buses. In *edaWorkshop*, pages 15–20, Hannover, Germany, June 2007.

[187] T. MURGAN, P. B. BACINSCHI, S. PANDEY, A. GARCÍA ORTIZ, and M. GLESNER. On the Necessity of Combining Coding with Spacing and Shielding for Improving Performance and Power in Very Deep Sub-Micron Interconnects. In *Intl. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 242–254, Göteborg, Sweden, Sept. 2007.

[188] T. MURGAN, A. GUNTORO, H. HINKELMANN, P. B. BACINSCHI, and M. GLESNER. Low-Complexity Adaptive Encoding Schemes Based on Partial Bus-Invert for Power Reduction in Buses Exhibiting Capacitive Coupling. In *Intl. Workshop on Reconfigurable Communication Centric System-on-Chips (ReCoSoC)*, pages 7–14, Montpellier, France, June 2007.

[189] T. MURGAN, O. MITEA, S. PANDEY, P. B. BACINSCHI, and M. GLESNER. Simultaneous Placement and Buffer Planning for Reduction of Power Consumption in Interconnects and Repeaters. In *IFIP Intl. Conf. on VLSI-SoC*, pages 302–307, Nice, France, Oct. 2006.

Supervised Theses

[190] P. BEDNAROVSCHI. Prognostics and Health Management Systems for Electronics. Studienarbeit, Technische Universität Darmstadt, May 2010. *Co-advised with Andre Guntoro*.

[191] E. CHENG. Performance Modeling of Pulsed Current-Mode Signaling Circuits. Studienarbeit, Technische Universität Darmstadt, July 2008.

[192] M. HOEHL. Scheduling and Mapping of Data Processing Tasks for a MPEG-4 SoC. Studienarbeit, Technische Universität Darmstadt, Sept. 2008.

[193] Y. HU. Equivalent Cell Models. Master's thesis, Technische Universität Darmstadt, Oct. 2007. *Co-advised with Klaus Koch and Tudor Murgan*.

[194] M. K. MEBRAHTU. Accurate In-Situ Measurement of Crosstalk-induced Delay Change and Waveform Reconstruction. Master's thesis, Technische Universität Darmstadt, Oct. 2006. *Co-advised with Tudor Murgan*.

[195] T. VOLLBERG. Präzise On-Chip Messung der durch Übersprechen erzeugten Spannungsspitzen. Master's thesis, Technische Universität Darmstadt, Feb. 2005. *Co-advised with Tudor Murgan*.

Lebenslauf

Petru Bogdan BACINSCHI

Zur Person:

Geburtsdatum:	24. Juli 1981
Geburtsort:	Luduş, Rumänien

Ausbildung:

1996 bis 2000	Gymnasium “Cantemir Vodă” in Bukarest Abschluss: Abitur
2000 bis 2005	Student an der Fakultät für Elektronik, Nachrichtentechnik und Informationstechnik, Universität ‘Politehnica’ Bukarest Abschluss: Diplomingenieur
2005 bis 2010	Doktorand am Fachgebiet Mikroelektronische Systeme der Technischen Universität Darmstadt

Beruflicher Werdegang:

2001 bis 2005	Softwareingenieur an der Firma Crystal Interactive Systems S.R.L. in Bukarest
2005 bis 2010	Wissenschaftlicher Mitarbeiter am Fachgebiet Mikroelektronische Systeme der Technischen Universität Darmstadt
seit dem 1.8.2010	Hardwarearchitekt bei Infineon Technologies AG in Neubiberg