Mourad Fakhfakh
Esteban Tlelo-Cuautle
Patrick Siarry *Editors*

Computational Intelligence in Digital and Network Designs and Applications

Springer

Computational Intelligence in Digital and Network Designs and Applications

Mourad Fakhfakh · Esteban Tlelo-Cuautle
Patrick Siarry
Editors

Computational Intelligence in Digital and Network Designs and Applications

Springer

Editors
Mourad Fakhfakh
Department of Electronics
ENET'Com, University of Sfax
Sfax
Tunisia

Patrick Siarry
Laboratoire LiSSi (EA 3956)
Université Paris-Est Créteil
Vitry sur Seine
France

Esteban Tlelo-Cuautle
Department of Electronics
INAOE
Tonantzintla
Mexico

ISBN 978-3-319-20070-5 ISBN 978-3-319-20071-2 (eBook)
DOI 10.1007/978-3-319-20071-2

Library of Congress Control Number: 2015942618

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

Preface

Computational intelligence has been an astounding success in the engineering domain, particularly in electronic design. Over the past two decades, improved techniques have raised the productivity of designers to a remarkable degree. Indeed, in the areas of digital, analog, radio-frequency, and mixed-signal engineering, there is a focused effort on trying to automate all levels of the design flow of electronic circuits, a field where it was long assumed that progress demanded a skilled designer's expertise. Thus, new computational-based modeling, synthesis and design methodologies, and applications of optimization algorithms have been proposed for assisting the designer's task.

This book offers the reader a collection of recent advances in computational intelligence—algorithms, design methodologies, and synthesis techniques—applied to the design of integrated circuits and systems. It highlights new biasing and sizing approaches and optimization techniques and their application to the design of high-performance digital, VLSI, radio-frequency, and mixed-signal circuits and systems.

As editors we invited experts from related design disciplines to contribute overviews of their particular fields, and we grouped these into the following:

- Volume 1, *Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design*, contains 17 chapters, divided into two parts: Analog and Mixed-Signal Applications (Chaps. 1–8); and Radio-Frequency Design (Chaps. 9–17).
- Volume 2, *Computational Intelligence in Digital and Network Designs and Applications*, contains 12 chapters, divided into three parts: Digital Circuit Design (Chaps. 1–6); Network Optimization (Chaps. 7–8); and Applications (Chaps. 9–12).

Here we present detailed descriptions of the chapters in both volumes.

Volume 1—Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design

Part I—Analog and Mixed-Signal Applications

Chapter 1, "I-Flows: A Novel Approach to Computational Intelligence for Analog Circuit Design Automation Through Symbolic Data Mining and Knowledge-Intensive Reasoning," was written by Fanshu Jiao, Sergio Montano, Cristian Ferent, and Alex Doboli. It presents an overview of the authors' ongoing work toward devising a new approach to analog circuit synthesis. The approach computationally implements some of the facets of knowledge-intensive reasoning that humans perform when tackling new design problems. This is achieved through a synthesis flow that mimics reasoning using a domain-specific knowledge structure with two components: an associative part and a causal reasoning part. The associative part groups known circuit schematics into abstractions based on the similarities and differences of their structural features. The causal reasoning component includes the starting ideas as well as the design sequences that create the existing circuits.

Chapter 2, "Automatic Synthesis of Analog Integrated Circuits Including Efficient Yield Optimization," was written by Lucas C. Severo, Fabio N. Kepler, and Alessandro G. Girardi. Here the authors show the main aspects and implications of automatic sizing, including yield. Different strategies for accelerating performance estimation and design space search are addressed. The analog sizing problem is converted into a nonlinear optimization problem, and the design space is explored using metaheuristics based on genetic algorithms. Circuit performance is estimated by electrical simulations and the generated optimal solution includes yield prediction as a design constraint. The method was applied for the automatic design of a 12-free-variables two-stage amplifier. The resulting sized circuit presented 100 % yield within a 99 % confidence interval, while achieving all the performance specifications in a reasonable processing time. The authors implemented an efficient yield-oriented sizing tool which generates robust solutions, thus increasing the number of first-time-right analog integrated circuit designs.

Chapter 3, "Application of Computational Intelligence Techniques to Maximize Unpredictability in Multiscroll Chaotic Oscillators," was written by Victor Hugo Carbajal-Gómez, Esteban Tlelo-Cuautle, and Francisco V. Fernández. It applies and compares three computational intelligence algorithms—the genetic algorithm (GA), differential evolution (DE), and particle swarm optimization (PSO)—to maximize the positive Lyapunov exponent in a multiscroll chaotic oscillator based on a saturated nonlinear function series based on the modification of the standard settings of the coefficient values of the mathematical description, and taking into account the correct distribution of the scrolls drawing the phase-space diagram. The experimental results show that the DE and PSO algorithms help to maximize the positive Lyapunov exponent of truncated coefficients over the continuous spaces.

Chapter 4, “Optimization and Cosimulation of an Implantable Telemetric System by Linking System Models to Nonlinear Circuits,” was written by Yao Li, Hao Zou, Yasser Moursy, Ramy Iskander, Robert Sobot, and Marie-Minerve Louërat. It presents a platform for modeling, design, optimization, and cosimulation of mixed-signal systems using the SystemC-AMS standard. The platform is based on a bottom-up design and top-down simulation methodologies. In the bottom-up design methodology, an optimizer is inserted to perform a knowledge-aware optimization loop. During the process, a Peano trajectory is applied for global exploration and the Nelder–Mead Simplex optimization method is applied for local refinement. The authors introduce an interface between system-level models and their circuit-level realizations in the proposed platform. Moreover, a transient simulation scheme is proposed to simulate nonlinear dynamic behavior of complete mixed signal systems. The platform is used to design and verify a low-power CMOS voltage regulator for an implantable telemetry system.

Chapter 5, “Framework for Formally Verifying Analog and Mixed Signal Designs,” was written by Mohamed H. Zaki, Osman Hasan, Sofiène Tahar, and Ghiath Al-Sammane. It proposes a complementary formal-based solution to the verification of analog and mixed-signal (AMS) designs. The authors use symbolic computation to model and verify AMS designs through the application of induction-based model checking. They also propose the use of higher-order-logic theorem proving to formally verify continuous models of analog circuits. To test and validate the proposed approaches they developed prototype implementations in Mathematica and HOL and target analog and mixed-signal systems such as delta sigma modulators.

Chapter 6, “Automatic Layout Optimizations for Integrated MOSFET Power Stages,” was written by David Guilherme, Jorge Guilherme, and Nuno Horta. It presents a design automation approach that generates automatically error free, area and parasitic optimized layout views of output power stages consisting of multiple power MOSFETs. The tool combines a multitude of constraints associated with DRC, DFM, ESD rules, current density limits, heat distribution, and placement. It uses several optimization steps based on evolutionary computation techniques that precede a bottom-up layout construction of each power MOSFET, its optimization for area and parasitic minimization, and its optimal placement within the output stage power topology network.

Chapter 7, “Optimizing Model Precision in High Temperatures for Efficient Analog and Mixed-Signal Circuit Design Using Modern Behavioral Modeling Techniques: an Industrial Case Study,” was written by Sahbi Baccar, Timothée Levi, Dominique Dallet, and François Barbara. It deals with the description of a modeling methodology dedicated to simulation of AMS circuits in high temperatures (HT). A behavioral model of an op-amp is developed using VHDL-AMS in order to remedy the inaccuracy of the SPICE model. The precision of the model simulation in HT was improved thanks to the VHDL-AMS model. Almost all known op-amp parameters were inserted into the model which was developed manually. The future work can automate the generation of such a behavioral model to describe the interdependency between different parameters. This is possible by using modern computational intelligence techniques, such as genetic algorithms, or other techniques such as Petri nets or model order reduction.

Chapter 8, "Nonlinearities Behavioral Modeling and Analysis of Pipelined ADC Building Blocks," was written by Carlos Silva, Philippe Ayzac, Nuno Horta, and Jorge Guilherme. It presents a high-speed simulation tool for the design and analysis of pipelined analog-to-digital converters implemented using the Python programming language. The development of an ADC simulator requires behavior modeling of the basic building blocks and their possible interconnections to form the final converter. This chapter presents a Pipeline ADC simulator tool which allows topology selection and digital calibration of the frontend blocks. Several block nonlinearities are included in the simulation, such as thermal noise, capacitor mismatch, gain and offset errors, parasitic capacitances, settling errors, and other error sources.

Part II—Radio-Frequency Design

Chapter 9, "SMAS: A Generalized and Efficient Framework for Computationally Expensive Electronic Design Optimization Problems," was written by Bo Liu, Francisco V. Fernández, Georges Gielen, Ammar Karkar, Alex Yakovlev, and Vic Grout. Many electronic design automation (EDA) problems encounter computationally expensive simulations, making simulation-based optimization impractical for many popular synthesis methods. Not only are they computationally expensive, but some EDA problems also have dozens of design variables, tight constraints, and discrete landscapes. Few available computational intelligence methods can solve them effectively and efficiently. This chapter introduces a surrogate model-aware evolutionary search (SMAS) framework, which is able to use much fewer expensive exact evaluations with comparable or better solution quality. SMAS-based methods for mm-wave integrated circuit synthesis and network-on-chip parameter design optimization are proposed, and are tested on several practical problems. Experimental results show that the developed EDA methods can obtain highly optimized designs within practical time limitations.

Chapter 10, "Computational Intelligence Techniques for Determining Optimal Performance Tradeoffs for RF Inductors," was written by Elisenda Roca, Rafael Castro-López, Francisco V. Fernández, Reinier González-Echevarría, Javier Sieiro, Neus Vidal, and José M. López-Villegas. The automatic synthesis of integrated inductors for radio-frequency (RF) integrated circuits is one of the most challenging problems that RF designers have to face. In this chapter, computational intelligence techniques are applied to automatically obtain the optimal performance tradeoffs of integrated inductors. A methodology is presented that combines a multiobjective evolutionary algorithm with electromagnetic simulation to get highly accurate results. A set of sized inductors is obtained showing the best performance tradeoffs for a given technology. The methodology is illustrated with a complete set of examples where different inductor tradeoffs are obtained.

Chapter 11, "RF IC Performance Optimization by Synthesizing Optimum Inductors," was written by Mladen Božanić and Saurabh Sinha. It reviews inductor theory and describes various integrated inductor options. It also explains why integrated planar spiral inductors are so useful when it comes to integrated RF circuits. Furthermore, the chapter discusses the theory of spiral inductor design, inductor modeling, and how this theory can be used in inductor synthesis. In the central part of the chapter the authors present a methodology for synthesis of planar spiral inductors, where numerous geometries are searched through in order to fit various initial conditions.

Chapter 12, "Optimization of RF On-Chip Inductors Using Genetic Algorithms," was written by Eman Omar Farhat, Kristian Zarb Adami, Owen Casha, and John Abela. It discusses the optimization of the geometry of RF on-chip inductors by means of a genetic algorithm in order to achieve adequate performance. Necessary background theory together with the modeling of these inductors is included in order to aid the discussion. A set of guidelines for the design of such inductors with a good quality factor in a standard CMOS process is also provided. The optimization process is initialized by using a set of empirical formulae in order to estimate the physical parameters of the required structure as constrained by the technology. Then automated design optimization is executed to further improve its performance by means of dedicated software packages. The authors explain how to use state-of-the-art computer-aided design tools in the optimization process and how to efficiently simulate the inductor performance using electromagnetic simulators.

Chapter 13, "Automated System-Level Design for Reliability: RF Front-End Application," was written by Pietro Maris Ferreira, Jack Ou, Christophe Gaquière, and Philippe Benabes. Reliability is an important issue for circuits in critical applications such as military, aerospace, energy, and biomedical engineering. With the rise in the failure rate in nanometer CMOS, reliability has become critical in recent years. Existing design methodologies consider classical criteria such as area, speed, and power consumption. They are often implemented using postsynthesis reliability analysis and simulation tools. This chapter proposes an automated system design for reliability methodology. While accounting for a circuit's reliability in the early design stages, the proposed methodology is capable of identifying an RF front-end optimal design considering reliability as a criterion.

Chapter 14, "The Backtracking Search for the Optimal Design of Low-Noise Amplifiers," was written by Amel Garbaya, Mouna Kotti, Mourad Fakhfakh, and Patrick Siarry. The backtracking search algorithm (BSA) was recently developed. It is an evolutionary algorithm for real-valued optimization problems. The main feature of BSA vis-à-vis other known evolutionary algorithms is that it has a single control parameter. It has also been shown that it has better convergence behavior. In this chapter, the authors deal with the application of BSA to the optimal design of

RF circuits, namely low-noise amplifiers. BSA performance, viz., robustness and speed, are checked against the widely used particle swarm optimization technique, and other published approaches. ADS simulation results are given to show the viability of the obtained results.

Chapter 15, "Design of Telecommunications Receivers Using Computational Intelligence Techniques," was written by Laura-Nicoleta Ivanciu and Gabriel Oltean. It proposes system-, block- and circuit-level design procedures that use computational intelligence techniques, taking into consideration the specifications for telecommunications receivers. The design process starts with selecting the proper architecture (topology) of the system, using a fuzzy expert solution. Next, at the block level, the issue of distributing the parameters across the blocks is solved using a hybrid fuzzy-genetic algorithms approach. Finally, multiobjective optimization using genetic algorithms is employed in the circuit-level design. The proposed methods were tested under specific conditions and have proved to be robust and trustworthy.

Chapter 16, "Enhancing Automation in RF Design Using Hardware Abstraction," was written by Sabeur Lafi, Ammar Kouki and Jean Belzile. It presents advances in automating RF design through the adoption of a framework that tackles primarily the issues of automation, complexity reduction, and design collaboration. The proposed framework consists of a design cycle along with a comprehensive RF hardware abstraction strategy. Being a model-centric framework, it captures each RF system using an appropriate model that corresponds to a given abstraction level and expresses a given design perspective. It also defines a set of mechanisms for the transition between the models defined at different abstraction levels which contributes to the automation of design stages. The combination of an intensive modeling activity and a clear hardware abstraction strategy through a flexible design cycle introduces intelligence, enabling higher design automation and agility.

Chapter 17, "Optimization Methodology Based on IC Parameter for the Design of Radio-Frequency Circuits in CMOS Technology," was written by Abdellah Idrissi Ouali, Ahmed El Oualkadi, Mohamed Moussaoui, and Yassin Laaziz. It presents a computational methodology for the design optimization of ultra-low-power CMOS radio-frequency front-end blocks. The methodology allows us to explore MOS transistors in all regions of inversion. The power level is set as an input parameter before we begin the computational process involving other aspects of the design performance. The approach consists of tradeoffs between power consumption and other radio-frequency performance parameters. This can help designers to seek quickly and accurately the initial sizing of the radio-frequency building blocks while maintaining low levels of power consumption. A design example shows that the best tradeoffs between the most important low-power radio-frequency performances occur in the moderate inversion region.

Volume 2—Computational Intelligence in Digital and Network Designs and Applications

Part I—Digital Design

Chapter 1, “Sizing Digital Circuits Using Convex Optimization Techniques,” was written by Logan Rakai and Amin Farshidi. It collects recent advances in using convex optimization techniques to perform sizing of digital circuits. Convex optimization techniques provide an undeniably attractive promise: The attained solution is the best available. In order to use convex optimization techniques, the target optimization problem must be modeled using convex functions. The gate sizing problem has been modeled in different ways to enable the use of convex optimization techniques, such as linear programming and geometric programming. Statistical and robust sizing methods are included to reflect the importance of optimization techniques that are aware of variations. Applications of multiobjective optimization techniques that aid designers in evaluating the tradeoffs are described.

Chapter 2, “A Fabric Component Based Approach to the Architecture and Design Automation of High-Performance Integer Arithmetic Circuits on FPGA,” was written by Ayan Palchaudhuri and Rajat Subhra Chakraborty. FPGA-specific primitive instantiation is an efficient approach for design optimization to effectively utilize the native hardware primitives as building blocks. Placement steps also need to be constrained and controlled to improve the circuit critical path delay. Here, the authors present optimized implementations of certain arithmetic circuits and pseudorandom sequence generator circuits to indicate the superior performance scalability achieved using the proposed design methodology in comparison to circuits of identical functionality realized using other existing FPGA CAD tools or design methodologies. The Hardware Description Language specifications as well as the placement constraints can be automatically generated. A GUI-based CAD tool has been developed that is integrated with the Xilinx Integrated Software Environment for design automation of circuits from user specifications.

Chapter 3, “Design Intelligence for Interconnection Realization in Power-Managed SoCs,” was written by Houman Zarrabi, A.J. Al-Khalili, and Yvon Savaria. Here various intelligent techniques for modeling, design, automation, and management of on-chip interconnections in power-managed SoCs are described, including techniques that take into account various technological parameters such as crosstalk. Such intelligent techniques guarantee that the integrated interconnections, used in power-managed SoCs, are well-designed, energy-optimal, and meet the performance objectives in all the SoC operating states.

Chapter 4, “Introduction to Optimization Under Uncertainty Techniques for High-Performance Multicore Embedded Systems Compilation,” was written by Oana Stan and Renaud Sirdey. The compilation process design for massively parallel multicore embedded architectures requires solving a number of difficult optimization problems, nowadays solved mainly using deterministic approaches. However, one of the main characteristics of these systems is the presence of uncertain data, such as the execution times of the tasks. The authors consider that embedded systems design is one of the major domains for which applying optimization under uncertainty is legitimate and highly beneficial. This chapter introduces the most suitable techniques from the field of optimization under uncertainty for the design of compilation chains and for the resolution of associated optimization problems.

Chapter 5, "Digital IIR Filter Design with Fix-Point Representation Using Effective Evolutionary Local Search Enhanced Differential Evolution," was written by Yu Wang, Weishan Dong, Junchi Yan, Li Li, Chunhua Tian, Chao Zhang, Zhihu Wang, and Chunyang Ma. Previously, the parameters of digital IIR filters were encoded with floating-point representations. It is known that a fixed-point representation can effectively save computational resources and is more convenient for direct realization on hardware. Inherently, compared with floating-point representation, fixed-point representation may make the search space miss much useful gradient information and, therefore, raises new challenges. In this chapter, the universality of DE-based MA is improved by implementing more efficient evolutionary algorithms (EAs) as the local search techniques. The performance of the newly designed algorithm is experimentally verified in both function optimization tasks and digital IIR filter design problems.

Chapter 6, "Applying Operations Research to Design for Test Insertion Problems," was written by Yann Kieffer and Lilia Zaourar. Enhancing electronic circuits with ad hoc testing circuitry—so-called Design for Test (DFT)—is a technique that enables one to thoroughly test circuits after production. But this insertion of new elements itself may sometimes be a challenge, for bad choices could lead to unacceptable degradations of features of the circuit, while good choices may help reduce testing costs and circuit production costs. This chapter demonstrates how methods from operations research—a scientific discipline rooted in both mathematics and computer science, leaning strongly on the formal modeling of optimization issues—help us to address such challenges and build efficient solutions leading to real-world solutions that may be integrated into electronic design software tools.

Part II—Network Design

Chapter 7, "Low-Power NoC Using Optimum Adaptation," was written by Sayed T. Muhammad, Rabab Ezz-Eldin, Magdy A. El-Moursy, Ali A. El-Moursy, and Amr M. Refaat. Two power-reduction techniques are exploited to design a low-leakage-power NoC switch. First, the adaptive virtual channel (AVC) technique is presented as an efficient way to reduce the active area using a hierarchical multiplexing tree of VC groups. Second, power gating reduces the average leakage power consumption of the switch by controlling the supply power of the VC groups. The traffic-based virtual channel activation (TVA) algorithm is presented to determine traffic load status at the NoC switch ports. The TVA algorithm optimally utilizes virtual channels by deactivating idle VC groups to guarantee high-leakage-power saving without affecting the NoC throughput.

Chapter 8, “Decoupling Network Optimization by Swarm Intelligence,” was written by Jai Narayan Tripathi and Jayanta Mukherjee. Here the problem of decoupling network optimization is discussed in detail. Swarm intelligence is used for maintaining power integrity in high-speed systems. The optimum number of capacitors and their values are selected to meet the target impedance of the system.

Part III—Applications

Chapter 9, “The Impact of Sensitive Inputs on the Reliability of Nanoscale Circuits,” was written by Usman Khalid, Jahanzeb Anwer, Nor H. Hamid, and Vijanth S. Asirvadam. As CMOS technology scales to nanometer dimensions, its performance and behavior become less predictable. Reliability studies for nano-circuits and systems become important when the circuit’s outputs are affected by its sensitive noisy inputs. In conventional circuits, the impact of the inputs on reliability can be observed by the deterministic input patterns. However, in nanoscale circuits, the inputs behave probabilistically. The Bayesian networks technique is used to compute the reliability of a circuit in conjunction with the Monte Carlo simulations approach which is applied to model the probabilistic inputs and ultimately to determine sensitive inputs and worst-case input combinations.

Chapter 10, “Pin-Count and Wire Length Optimization for Electrowetting-on-Dielectric Chips: A Metaheuristics-Based Routing Algorithm,” was written by Mohamed Ibrahim, Cherif Salama, M. Watheq El-Kharashi, and Ayman Wahba. Electrowetting-on-dielectric chips are gaining momentum as efficient alternatives to conventional biochemical laboratories due to their flexibility and low power consumption. In this chapter, we present a novel two-stage metaheuristic algorithm to optimize electrode interconnect routing for pin-constrained chips. The first stage models channel routing as a travelling salesman problem and solves it using the ant colony optimization algorithm. The second stage provides detailed wire routes over a grid model. The algorithm is benchmarked over a set of real-life chip specifications. On average, comparing our results to previous work, we obtain reductions of approximately 39 % and 35 % on pin count and total wire length, respectively.

Chapter 11, “Quantum Dot Cellular Automata: A Promising Paradigm Beyond Moore,” was written by Kunal Das, Arijit Dey, Dipannita Podder, Mallika De, and Debashis De. The quantum dot cellular automata (QCA) is a promising paradigm to overcome the ever-growing needs in size, power, and speed. In this chapter we explore charge-confined low-power optimum logic circuit design to enhance the computing performance of a novel nanotechnology architecture, the quantum dot cellular automata. We investigate robust and reliable diverse logic circuit design, such as hybrid adders and other binary adder schemes, among them bi-quinary and Johnson–Mobius, in QCA. We also examine zero-garbage lossless online-testable adder design in QCA. Multivalued logic circuit design, with potential advantages such as greater data storage, fast arithmetic operation, and the ability to solve nonbinary problems, will be important in multivalued computing, especially in the ternary computing paradigm.

Chapter 12, “Smart Videocapsule for Early Diagnosis of Colorectal Cancer: Toward Embedded Image Analysis,” was written by Quentin Angermann, Aymeric Histace, Olivier Romain, Xavier Dray, Andrea Pinna, and Bertrand Granado. Wireless capsule endoscopy (WCE) enables screening of the gastrointestinal tract by a swallowable imaging system. However, contemporary WCE systems have several limitations—battery, low processing capabilities, among others—which often result in low diagnostic yield. In this chapter, after a technical presentation of the components of a standard WCE, the authors discuss the related limitations and introduce a new concept of smart capsule with embedded image processing capabilities based on a boosting approach using textural features. We discuss the feasibility of the hardware integration of the detection–recognition method, also with respect to the most recent FPGA technologies.

Finally, the editors wish to use this opportunity to thank all the authors for their valuable contributions, and the reviewers for their help for improving the quality of the contributions.

The editors are also thankful to Ronan Nugent, Springer Senior Editor, for his support, and for his continuous help.

Enjoy reading the book.

Sfax Mourad Fakhfakh
Puebla Esteban Tlelo-Cuautle
Paris Patrick Siarry
December 2014

Contents

Contributors

A.J. Al-Khalili Department of ECE, Concordia University, Montréal, Canada

Quentin Angermann ETIS UMR CNRS 8051, ENSEA-University of Cergy-Pontoise, Cergy, France

Jahanzeb Anwer University of Paderborn, Paderborn, Germany

Vijanth S. Asirvadam Universiti Teknologi Petronas, Teronoh, Malaysia

Rajat Subhra Chakraborty Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India

Kunal Das B. P. Poddar Institute of Management and Technology, Kolkata, India

Debashis De Department of Computer Science and Engineering, West Bengal University of Technology, Salt Lake City, Kolkata, India

Mallika De Dr. Sudhir Chandra Sur Degree Engineering College, Kolkata, West Bengal, India

Arijit Dey B. P. Poddar Institute of Management and Technology, Kolkata, India

Weishan Dong IBM Research-China, Beijing, China

Xavier Dray ETIS Lab and APHP Hôpital Lariboisière, Paris Diderot, Paris 7 University, Paris, France

M. Watheq El-Kharashi Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt

Ali A. El-Moursy Department of Electrical and Computer Engineering, University of Sharjah, Sharjah, UAE

Magdy A. El-Moursy Mentor Graphics Corporation, Cairo, Egypt

Rabab Ezz-Eldin Department of Electrical Engineering, Beni-Suef University, Beni-Suef, Egypt

Amin Farshidi University of Calgary, Calgary, AB, Canada

Bertrand Granado LIP6 Lab, University Pierre Et Marie Curie, Paris, France

Nor H. Hamid Universiti Teknologi Petronas, Teronoh, Malaysia

Aymeric Histace ETIS UMR CNRS 8051, ENSEA-University of Cergy-Pontoise, Cergy, France

Mohamed Ibrahim Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt

Usman Khalid Universiti Teknologi Petronas, Teronoh, Malaysia

Yann Kieffer LCIS, University Grenoble Alpes, Valence, France

Li Li IBM Research-China, Beijing, China

Chunyang Ma IBM Research-China, Beijing, China

Sayed T. Muhammad Department of Electrical Engineering, Beni-Suef University, Beni-Suef, Egypt

Jayanta Mukherjee Indian Institute of Technology Bombay, Mumbai, India

Ayan Palchaudhuri Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India

Andrea Pinna LIP6 Lab, University Pierre Et Marie Curie, Paris, France

Dipannita Podder B. P. Poddar Institute of Management and Technology, Kolkata, India

Logan Rakai University of Calgary, Calgary, AB, Canada

Amr M. Refaat Department of Electrical Engineering, Fayoum University, Fayoum, Egypt

Olivier Romain ETIS UMR CNRS 8051, ENSEA-University of Cergy-Pontoise, Cergy, France

Cherif Salama Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt

Yvon Savaria Department of EEE, Ecole Polytechnique de Montréal, Montréal, Canada

Renaud Sirdey Embedded Real Time Systems Laboratory, CEA, LIST, Gif-Sur-Yvette, France

Oana Stan Embedded Real Time Systems Laboratory, CEA, LIST, Gif-Sur-Yvette, France

Chunhua Tian IBM Research-China, Beijing, China

Jai Narayan Tripathi STMicroelectronics Pvt. Ltd., Gangapur (Bhilwara), Rajasthan, India

Ayman Wahba Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt

Yu Wang IBM Research-China, Beijing, China

Zhihu Wang IBM Research-China, Beijing, China

Junchi Yan IBM Research-China, Beijing, China

Lilia Zaourar CEA List, Gif-sur-Yvette, France

Houman Zarrabi Department of ECE, Concordia University, Montréal, Canada

Chao Zhang IBM Research-China, Beijing, China

Part I
Digital Design

Chapter 1
Sizing Digital Circuits Using Convex Optimization Techniques

Logan Rakai and Amin Farshidi

Abstract This chapter collects recent advances in using convex optimization techniques to perform sizing of digital circuits. Convex optimization techniques provide an undeniably attractive promise: The attained solution is the best available. In order to use convex optimization techniques, the target optimization problem must be modeled using convex functions. The gate sizing problem has been modeled in different ways to enable the use of convex optimization techniques, such as linear programming and geometric programming. Statistical and robust sizing methods are included to reflect the importance of optimization techniques that are aware of variations. Applications of multi-objective optimization techniques that aid designers in evaluating the trade-offs are described.

1.1 Introduction

Convex optimization and digital gate sizing have a rich history together beginning in 1985 [11, 25]. The field of convex optimization has evolved and although the goal of the original work is still relevant today, there are many more considerations around selecting the appropriate sizes of gates. Because of this, new applications of convex optimization techniques to gate sizing have continued to occur ever since. This chapter explains some of the contributions in the domain of gate sizing related to convex optimization. There are many works which cannot be covered, however, the coverage in this chapter serves as a broad overview of the kind of work that has been accomplished focusing mostly within the past decade. The mathematical preliminaries required to understand convex optimization and many of its different forms are given to allow the reader to appreciate the contributions that have been made. Furthermore, it is a goal of this chapter to inspire readers to continue the advancement and consider new formulations based on the wealth of prior work in the area.

L. Rakai (✉) · A. Farshidi
University of Calgary, Calgary, AB, Canada
e-mail: lmrakai@ucalgary.ca

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_1

The chapter is organized as follows: In Sect. 1.2, convex optimization is explained along with necessary background to understand the contributions in digital gate sizing using convex optimization. Recent linear programming formulations for gate sizing are described in Sect. 1.3. A historical account of and state-of-the-art advances in gate sizing using geometric programming are provided in Sect. 1.4. Section 1.5 covers process variation-aware convex optimization formulations using robust optimization and stochastic programming techniques. Advances in the simultaneous optimization of multiple objectives using convex optimization are explored in Sect. 1.6. Finally, the chapter concludes with Sect. 1.7 which presents an overview of the gate sizing works presented using convex optimization.

1.2 Mathematical Preliminaries

In order to fully appreciate the contributions of the work presented in this chapter, a brief overview of relevant optimization topics is presented. The goal is to provide the reader with only the most basic level of knowledge in each subject area. To obtain a more thorough understanding of topic, the reader is referred to the literature cited therein. Those who are familiar with the subjects may proceed to Sect. 1.3.

1.2.1 Convex Optimization

Convex optimization is the foundation that allows many fields in science and engineering to perform powerful analysis. Some of the main features of convex optimization are:

- Each problem has a single local optima.
- A broad class of practical problems, such as gate sizing, can be modeled or well-approximated using convex optimization.
- A wide array of high-quality software programs are available for solving convex optimization problems.

Theorems from the field of optimization are able to provide necessary and sufficient conditions to verify the optimality of a solution [6].

To begin understanding convex optimization, one must first understand convex functions. A function $f(\cdot)$ is convex if the following inequality is satisfied at every point in an interval specifying the domain of the function:

$$\theta f(\mathbf{x_1}) + (1-\theta) f(\mathbf{x_2}) \geq f(\theta \mathbf{x_1} + (1-\theta)\mathbf{x_2}), 0 \leq \theta \leq 1, \tag{1.1}$$

and $\mathbf{x_1}, \mathbf{x_2}$ are any inputs in the domain of $f(\cdot)$. The inequality states that linearly interpolating the function value between two points will overestimate the function value. This can be visualized as in Fig. 1.1a. In this figure, linearly interpolating between function values for any value θ results in an overestimation of the actual

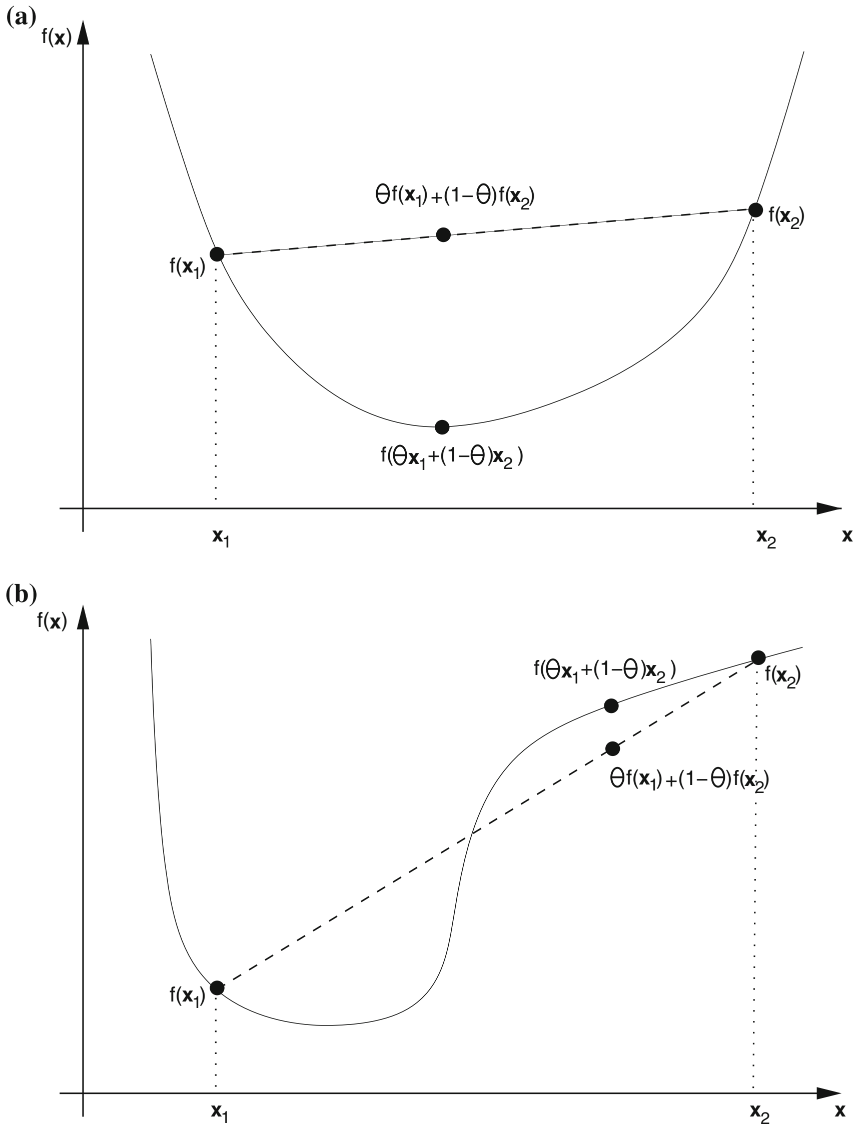


Fig. 1.1 An examples of a convex and non-convex functions. **a** Convex function. **b** Non-convex function

function value. If the greater than or equal to sign is replaced by a less than or equal to sign, the function is said to be concave. Functions that are both convex and concave are necessarily affine, or linear plus a constant, meaning they can be expressed as

$$f(\mathbf{x}) = \mathbf{a}^{\mathbf{T}}\mathbf{x} + b \tag{1.2}$$

where **a** is a vector and b is a scalar. An example of a function that is neither convex (non-convex) nor concave is presented in Fig. 1.1b.

Loosely speaking, convex optimization is optimizing over convex functions. More formally, a convex optimization problem formulation is written in standard form as follows:

$$\begin{aligned} \min \ & f(\mathbf{x}) \\ \text{s.t.} \ & g_i(\mathbf{x}) \leq 0, \ i = 1, \ldots, n \\ & h_i(\mathbf{x}) = 0, \ i = 1, \ldots, m \end{aligned} \tag{1.3}$$

where $\mathbf{x}$ is a vector variable to be optimized, $f(\cdot)$ is the convex objective function, $g_i(\cdot)$ are convex inequality constraint functions and $h_i(\cdot)$ are affine equality constraint functions. The number of inequality and equality constraints are represented by n and m, respectively. The optimization problem in (1.3) is a minimization problem which aims to find the value of $\mathbf{x}$ that minimizes the objective function subject to the constraints. If the problem were to solve for a value of $\mathbf{x}$ that maximizes a concave objective function, the problem can still be formulated in standard form by replacing $f(\mathbf{x})$ with $-f(\mathbf{x})$, and similarly, if the inequality constraints involve a greater than or equal to constraint.

Convex optimization problems can be solved efficiently on modern computers even for problems involving millions of variables. Many different algorithms and software implementations exist for solving convex optimization problems [3, 6]. However, the reader can take it for granted that such algorithms and software exist and do not need to know the inner workings.

There are many subclasses of convex optimization, each with their own properties but retaining the overarching properties of convex optimization. For example, if all of the functions in (1.3) are affine, the problem is called a linear program. Linear programs are among the easiest classes of optimization problems to solve. If $f(\mathbf{x})$ is a quadratic function, the problem is called quadratic program. The following sections discuss in more detail other classes of convex optimization problems relevant to gate sizing in more detail.

1.2.2 Geometric Programming

A class of convex optimization problems is known as geometric programming. The use of geometric is in the same sense as geometric mean or geometric series which involve products. As will soon be explained, the functions involved in geometric programming are products of variables. The term programming refers to mathematical programming which is a synonym for the field of optimization. A geometric programming optimization problem in standard form is defined as:

$$\begin{aligned} \min\ & f(\mathbf{x}) \\ \text{s.t.}\ & g_i(\mathbf{x}) \le 1,\ i = 1, \dots, n \\ & h_i(\mathbf{x}) = 1,\ i = 1, \dots, m \\ & \mathbf{x} > \mathbf{0} \end{aligned} \tag{1.4}$$

In addition, the optimization problem in (1.4) is a geometric programming, if for $i = 1, \dots, m$, $h_i(\mathbf{x})$ are monomials and $f(\mathbf{x})$ and $g_i(\mathbf{x})$, $i = 1, \dots, n$, are posynomials [6]. The function $h_i(\cdot)$ is a monomial if it is of the form:

$$h_i(\mathbf{x}) = c x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_{n_e}^{\alpha_{n_e}},$$

where $c > 0$ and the exponents, $\alpha_1, \dots, \alpha_{n_e} \in \Re$, are real numbers. Posynomials are summations of monomials with positive summation coefficients. Posynomials are like polynomials but can have real exponents and can have only positive real coefficients.

The advantage of geometric programming is that it can be converted into a convex problem [6] by variable substitution. Define the new variable z_i associated with x_i such that $z_i = \log(x_i)$, and introduce a new constant $c' = \log(c)$. A monomial $h(\cdot)$ will become:

$$h(\mathbf{z}) = e^{c'} e^{\alpha_1 z_1} \cdots e^{\alpha_{n_e} z_{n_e}} = e^{\boldsymbol{\alpha}^T \mathbf{z} + c'}.$$

The geometric programming optimization problem in (1.4) can be transformed to:

$$\begin{aligned} \min\ & \sum_{k=1}^{p_0} e^{\boldsymbol{\alpha}_{0k}^T \mathbf{z} + c'_{0k}} \\ \text{s.t.}\ & \sum_{k=1}^{p_i} e^{\boldsymbol{\alpha}_{ik}^T \mathbf{z} + c'_{ik}} \le 1,\ i = 1, \dots, n \\ & e^{\boldsymbol{\beta}_i^T \mathbf{z} + c'_i} = 1,\ i =, \dots, m \end{aligned} \tag{1.5}$$

where $\boldsymbol{\alpha}_{0k}$ contains all the coefficients of the objective posynomial, $\boldsymbol{\alpha}_{ik}$ contains the coefficients of ith inequality constraint posynomial and $\boldsymbol{\beta}_i$ contains the coefficients of the ith equality constraint monomial. p_0 is the number of monomials in the objective posynomial while p_i is the number of monomials in the ith inequality constraint posynomial.

By taking the logarithm of (1.5), the problem is converted to:

$$\min\ \log\left(\sum_{k=1}^{p_0} e^{\boldsymbol{\alpha}_{0k}^T \mathbf{z} + c'_{0k}} \right)$$

$$\text{s.t. } \log\left(\sum_{k=1}^{p_i} e^{\boldsymbol{\alpha}_{ik}^T \mathbf{z} + c'_{ik}}\right) \leq 0,\ i = 1, \ldots, n$$

$$\boldsymbol{\beta}_i^T \mathbf{z} + c'_i = 0,\ i = 1, \ldots, m$$

The objective and the inequality constraints are log-sum-exp functions that are convex functions [6]. Also, the equality constraints are linear functions. Therefore, the problem is a convex problem that can be solved using the available nonlinear convex optimization techniques [17].

1.2.3 Robust Optimization

In many optimization problems, the parameters and the variables of the models are considered to be deterministic values. However, in the real world, these parameters or variables include uncertainties. In gate sizing, the major source of uncertainty is process variations. For example, the length of a buffer might be nominally assigned 45nm, but, during the fabrication, the actual length can measure 20 % lower or higher (between 36 and 54 nm) [24].

In traditional optimization techniques, these uncertainties are ignored and only nominal solutions are obtained. These nominally optimal solutions can be very sensitive to variation in parameters and can even cause infeasibility in practice. In robust optimization techniques [1, 4], an optimized solution considering the uncertainties is obtained. Robust optimization provides a pessimistic solution, however, the optimal solution is feasible within an uncertainty set. Consider an optimization problem formulated in the format given below:

$$\begin{aligned} \min_{\mathbf{x}}\ & f(\mathbf{x}) \\ \text{s.t. }\ & g(\mathbf{x}) \leq \mathbf{0} \end{aligned}$$

where $\mathbf{x}$ are the variables of the problem, $f(.)$ is the objective of the problem and $g(.)$ represents the constraints. The robust formulation of this problem which includes uncertainties can be written as:

$$\begin{aligned} \min_{\mathbf{x}} \max_{\mathbf{u} \in U}\ & f(\mathbf{x}, \mathbf{u}) \\ \text{s.t. }\ & \max_{\mathbf{u} \in U} g(\mathbf{x}, \mathbf{u}) \leq 0 \end{aligned}$$

where $u \in U$ represents the uncertainties. The presented robust formulation minimizes the objective function considering the maximum uncertainties and insures that the maximum constraints are always kept feasible. A robust solution looks inferior if directly compared to an individual nominally optimal solution obtained under the assumption of perfect conditions. However, when uncertainty is considered, e.g., by using Monte Carlo simulations, the robust solution will have the least amount of changes in the objective value under the given uncertainty set.

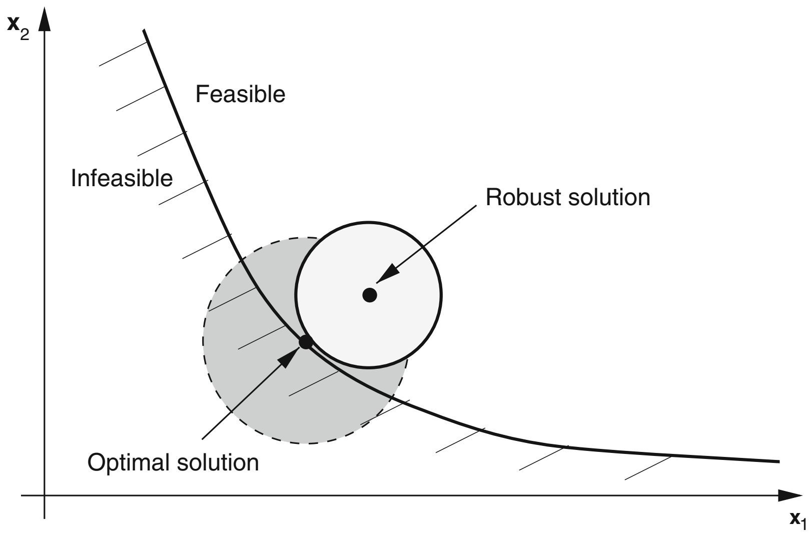


Fig. 1.2 An illustration of a solution attained by robust optimization and optimization not considering variations

The illustration in Fig. 1.2 is useful in visualizing what is happening when a robust optimization problem is being solved. In the illustration, assume the objective function decreases in the direction of lower values of the two variables x_1 and x_2 so the cost of a solution decreases as it moves closer to the origin. The solid black line divides the feasible region, where the constraints are satisfied, and the infeasible region, where at least one constraint is violated. There are two solutions in the figure, the solution labeled "optimal solution" is the solution attained by solving the problem without considering uncertainty. The solution labeled "robust solution" is the solution attained solving the same formulation but considering uncertainty. Each solution yields a set of possible values that consist of the solution attained for all possible values of $\mathbf{u}$ in the uncertainty set. In other words, the solution point is when $\mathbf{u} = \mathbf{0}$ and all non-zero values of u in the uncertainty set perturb the solution around that point. The circles around each solution indicate the possible solutions when uncertainty is nonzero. Three observations can be made:

1. The optimal solution point is better than the robust solution point because has a lower objective function value, i.e., it is closer to the origin.
2. The robust solution set is entirely inside the feasible region of the constraints.
3. The robust solution set is smaller than the solution set of obtained without considering uncertainty.

In many contexts it is desirable to have the robust solution that will always satisfy the constraints and be less sensitive to changes in parameters. Considering gate sizing, process variations are a source of uncertainty. The variations can cause functional specifications of a device to not be met resulting in a loss of yield. It is usually worthwhile to have higher yield at the price of a small increase in power or area.

1.2.4 Stochastic Programming

Similar to robust optimization, stochastic programming is aware of uncertainty in the data creating the model of the problem and formulation. Where they differ is that stochastic programming is aware of a probability distribution for the uncertain values, while robust optimization uses bounds and optimizes under worst-case assumptions. Stochastic programming is well known for yielding intractable problems when no analytical expressions for the distributions can be obtained but their convex approximations can be solved efficiently [8]. The two techniques can be thought of as complimentary in the sense stochastic programming approximations and robust optimization versions of the same initial formulation can be tractable in one and not the other. Because distributions are involved, stochastic programming requires having access to historical data or the ability to generate sample data to approximate or fit probability distributions.

Once a distribution of the uncertain values is known, it can be considered in various ways. One way is by considering expected values as in the following stochastic optimization problem formulation:

$$\begin{aligned} \min\ & \mathbf{E}f(\mathbf{x},\mathbf{u}) \\ \text{s.t.}\ & \mathbf{E}g_i(\mathbf{x},\mathbf{u}) \le 0,\ i=1,\ldots,n \\ & h_i(\mathbf{x}) = 1,\ i=0,\ldots,m \end{aligned} \tag{1.6}$$

where $\mathbf{E}$ is the expected value operator and $\mathbf{u}$ is the random variable with known distribution characterizing the uncertainty. An interpretation of this formulation for minimizing delay while considering the effects of process variations in gate sizing is that the most probable delay is being minimized. It can be proven that if $f(\cdot,\mathbf{u})$ is convex for all possible $\mathbf{u}$ then the expectation of the function is also convex [18]. This implies that if the objective and inequality functions are convex considering uncertainty, (1.6) is a convex optimization problem and can thus be solved efficiently to a global optima.

An alternative approach to considering uncertainty is to remove the uncertainty by taking the expected value of the random variables before optimizing. This approach is sometimes referred to as the certainty-equivalent [6] of (1.6) and can be formulated as:

$$\begin{aligned} \min\ & f(\mathbf{x},\mathbf{Eu}) \\ \text{s.t.}\ & g_i(\mathbf{x},\mathbf{Eu}) \le 0,\ i=1,\ldots,n \\ & h_i(\mathbf{x}) = 1,\ i=0,\ldots,m. \end{aligned} \tag{1.7}$$

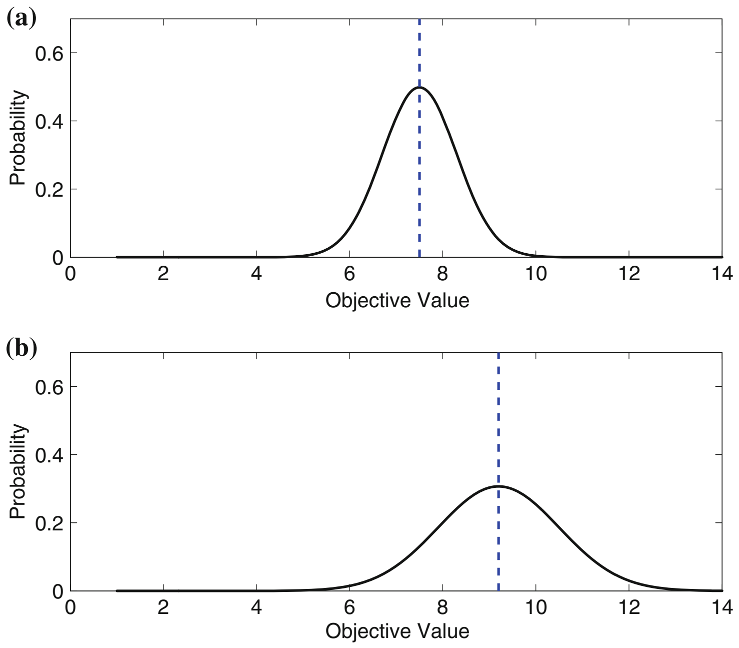


Fig. 1.3 A comparison of stochastic programming solutions. **a** Solution attained using the formulation (1.6). **b** Solution attained using the formulation (1.7)

Note that the expectation operates directly on the random variables and not on the function of the random variables. In this case, the random variables are no longer considered when solving in the optimization problem and are treated as constants. This approach can be useful when not enough data can be attained to properly characterize the underlying distribution of the uncertainty or when the functions involved lose convexity when uncertainty values are sampled form the tails of the distribution.

Figure 1.3 is presented to exemplify a typical difference between the solution attained by solving (1.6) and (1.7). The figures illustrate the distribution of the solution in the presence of uncertainty. The distribution of the solution attained by solving (1.6) is in Fig. 1.3a and the distribution of the solution attained by solving (1.7) is in Fig. 1.3b. The blue line in both figures indicates the expected value of each distribution. Because $f(\mathbf{x}, \mathbf{E}\mathbf{u}) \leq \mathbf{E}f(\mathbf{x}, \mathbf{u})$ [6] the solution to (1.7) is a lower bound on the solution to (1.6). However, when the expectation of each distribution's solution set is compared, the solution to (1.6) is superior in both expectation and variance. This is despite (1.7) having a lower objective value for the single point in the uncertainty set that the formulation solves for.

One last approach to stochastic optimization relevant to gate sizing works is to consider satisfying the constraints with a specified probability. A constraint which is satisfied to a certain probability is referred to as a chance constraint, which acknowledges the fact that there is a chance that the constraint is violated. Chance constraints have the following form:

$$P\left(g_i(\mathbf{x},\mathbf{u}) \le 0\right) \ge \eta_i \tag{1.8}$$

where $P(\cdot)$ is the probability function and $0 < \eta_i < 1$ is the certainty to which the constraint is satisfied. Assuming $g_i(\mathbf{x},\mathbf{u})$ is convex, the probability of satisfying the constraint is also a convex function for some very useful distributions of $\mathbf{u}$ including normal and log-concave distributions. Optimization formulations considering chance constraints fall into the class of chance-constrained optimization problems. A standard chance-constrained problem formulation is stated as:

$$\begin{aligned} \min\ & \mathbf{E} f(\mathbf{x},\mathbf{u}) \\ \text{s.t.}\ & P\left(g_i(\mathbf{x},\mathbf{u}) \le 0\right) \ge \eta_i,\ i = 1,\ldots,n \\ & h_i(\mathbf{x}) = 0,\ i = 1,\ldots,m. \end{aligned} \tag{1.9}$$

These formulations do require selecting values for η_i before solving. The choice can be made based on expert knowledge or common values such as 0.9, 0.95, or 0.99 can be used to loosely translate the desire of designers to satisfy specific constraints.

1.2.5 Multi-Objective Optimization

Multi-objective optimization problems can be expressed as:

$$\begin{aligned} \min\ & [f_1(\mathbf{x})\ \cdots\ f_O(\mathbf{x})]^T \\ \text{s.t.}\ & \mathbf{x} \in X \end{aligned} \tag{1.10}$$

where the goal is to minimize a vector-valued objective $[f_1(\mathbf{x})\ \cdots\ f_O(\mathbf{x})]^T$ over the feasible set X. The objective components are often competing meaning that all components cannot be simultaneously minimized. One vector $\mathbf{x}_a \in X$ is said to *dominate* another $\mathbf{x}_b \in X$ if

$$f_i(\mathbf{x}_a) \le f_i(\mathbf{x}_b),\ i = 1,\ldots,O,$$

with at least one constraint strictly less. A vector that is in X and not dominated by any other vector in X is said to be *Pareto optimal* and exists on the *Pareto front* when $O = 2$ and the *Pareto surface* when $O > 2$. If a vector $\mathbf{x}_a \in X$ is neither dominated nor dominates a vector $\mathbf{x}_c \in X$, the two vectors are not comparable. These concepts are illustrated in Fig. 1.4a where the dotted line is a convex Pareto front, i.e., the feasible set is on and above the line.

A designer must decide on which solution to choose from the many optimal solutions on the Pareto surface. This can be a very difficult task. Three different Pareto

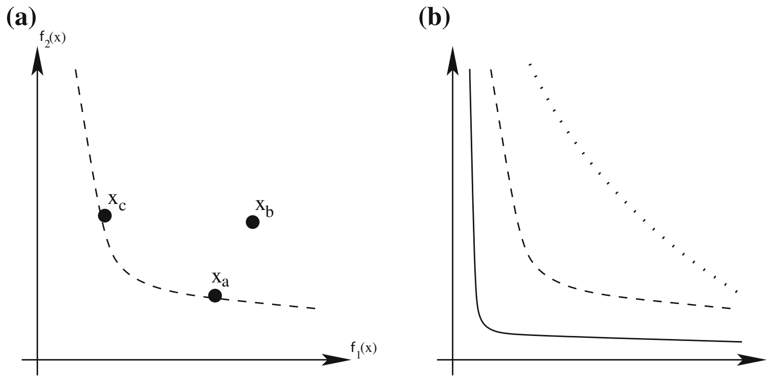


Fig. 1.4 Illustration of multi-objective optimization concepts. **a** Dominated and non-comparable vectors. **b** Pareto front curves

fronts are illustrated in Fig. 1.4b. The frontiers trace out Pareto optimal solutions with different pairs of objectives. In the most favorable case of the solid line, although every point on the line is Pareto optimal, designers are naturally drawn to the knee of the curve. At the knee, the incremental gain in each objective is maximized. When the knee is not very pronounced, as in the case of the dotted line, the designer has a more difficult decision to make. Decreasing one objective comes at the cost of a proportionate increase in the other. The analysis is further complicated when more than two objectives are considered. The shape of the Pareto surface is problem-dependent, but this chapter is limited to convex problems.

The most common way of solving multi-objective problems is scalarizing the vector-valued objective by forming a weighted sum of objective components [16]. The optimal solution obtained by minimizing a weighted sum is a single point on the Pareto surface. Weights are typically selected based upon designer priorities or heuristics [15, 26]. In the extreme but common case, designers select a single objective and make the other objective components into constraints. A solution can also be chosen after a potentially time-consuming search using several different weights to gain insight into the shape of the Pareto surface [16].

1.3 Linear Programming Formulations

Linear Programming (LP) is among the most basic problem types in convex optimization. There are a variety of solution techniques available for solving LPs that scale well and can handles hundreds of thousands to millions of variables and constraints. Berkelaar showed a technique to minimize power while maintaining delay by using an LP formulation [2]. The delay model used in the work is simple but nonlinear. In the work, they use piecewise linear approximation of the nonlinear delay function to formulate the problem as an LP. The strategy of approximating any convex delay function with a finite number of linear pieces is powerful and the number of pieces can control the error in the approximation. Since this early work, there has been great progress in using LPs in the context of sizing gates in digital circuits. The remainder of this section highlights a few of the most significant works.

The work in [7] begins by proving that greedily sizing gates is suboptimal and that convex optimization overcomes the shortcomings. An LP to minimize the power of a combinational circuit is used as a starting point in the work [19]. In the formulation, each gate in a circuit is given one alternate gate to choose from that will result in a reduction of power. The formulation is as follows:

$$\begin{aligned} \min\ & \sum_{v \in V} \gamma_v \Delta P_v \\ \text{s.t. } & t_v + d_v + \gamma_v \Delta d_v \le t_w \le T_{\max} \\ & 0 \le \gamma_v \le 1 \end{aligned} \tag{1.11}$$

where V is the set of all gates, $\Delta P_v < 0$ is the reduction in power if the alternate gate for gate v is chosen, and γ_v is the choice variable. The alternate gate is selected if $\gamma_v = 1$ and not selected if $\gamma_v = 0$, however the value can lie in between. The delay constraint says that the arrival time at gate v, t_v, plus the delay of gate v plus the increase in delay if the alternative gate is chosen, Δd_v, is less than the arrival time of downstream gate w. T_{max} is the maximum allowed delay of the circuit. Once the values of the choice variables are computed, the cell that reduces the power of gate v with delay less than $d_v + \gamma_v \Delta d_v$ is selected.

The formulation in [7] considers mutli-V$_{\text{dd}}$ and mutli-V$_{\text{th}}$, slew, separate rise and fall times, wire loads, all timing arcs as opposed to only the critical path of each gate, and allows upsizing ($\Delta d_v < 0$ and $\Delta P_v > 0$). The formulation is iteratively solved with the best alternate gate being provided at each iteration, if a potential alternate gate can reduce the power and the delay it is selected, if minimizing power is the objective the best alternate in terms of power reduction is provided, or the best reduction in delay if the delay is being minimized. If the goal is to minimize power, the objective is:

$$\min \sum_{v \in V} \gamma_v \Delta P_v \tag{1.12}$$

and the constraints

$$\begin{aligned} T &\le T_{max} \\ T &= \max_{v \in Outputs} \left\{ t_{v,rise}, t_{v,fall} \right\} \end{aligned} \tag{1.13}$$

are included. If the goal is to reduce delay, the objective is:

$$\min \left(\max(\tau, T - T_{max}) + k \sum_{v \in V} \gamma_v \Delta P_v \right) \tag{1.14}$$

where τ controls how much delay reduction is desired and k is used to equalize the relative weights of delay and power in the objective. As in (1.11), each gate with an alternative has a choice variable

$$0 \leq \gamma_v \leq 1, \forall v \in V. \tag{1.15}$$

Each timing arc has an arrival time constraint as follows:

$$\begin{aligned} t_{uv,fall} + d_{uv,rise} + \gamma_v \left(\Delta d_{uv,rise,v} + \beta_{vw} \Delta s_{uv,rise,v}\right) \\ + \sum_{x \in Fanout(v), x \neq w} \gamma_v \left(\Delta d_{uv,rise,x} + \beta_{vw} \Delta s_{uv,rise,x}\right) \leq t_{vw,rise} \end{aligned} \tag{1.16}$$

where $t_{uv,fall}$ is the falling arrival time at v from gate u. $d_{uv,rise}$ is the delay from the signal on timing arc uv to the output of v rising. $\Delta d_{uv,rise,x}$ and $\Delta s_{uv,rise,x}$ are the changes in delay and slew out of this timing arc if the cell of x changes. β is the transitive fanout delay/slew sensitivity. Multiplying by β_{vw} gives the worst-case transitive slew impact on delay. This constraint treats all gates that have a falling input causing a rising single output. The constraint for other polarities and number of outputs are similar. Static timing analysis and power analysis are performed before each iteration to compute power, slew, and delay values. Level converters are added as necessary when considering multi-V_{dd} supplies. The formulation is repeatedly solved until the solution converges.

The results from using this formulation showed a 10–16 % reduction in power compared to results achieved by the leading commercial software, however it was not as successful when minimizing delay. The runtime complexity is demonstrated to be linear in the number of gates and is therefore scalable. When multiple V_{th} gates are considered, the selection of V_{th} values is important as some combinations of levels showed little benefit and including more than two levels also showed minimal gains. The best pair of V_{th} values and V_{dd} values combined to give a 14 % improvement over single V_{th} and V_{dd}.

As the importance of leakage power in designs increased, new LP formulations were proposed [12]. Two formulations are proposed in that work: (1) leakage power minimization under timing constraints and (2) simultaneous circuit timing legalization to minimize the number of violating paths and leakage power optimization considering multiple gate lengths and gate length biasing from nominal values. Although using multiple gate length libraries can increase the dynamic power of a circuit, the reduction in leakage power overcomes those gains for a net reduction in power. In the work, 26 transitor gate lengths from 50 to 75 nm are considered.

A circuit is decomposed into a timing graph (V,E) with a single super-source S connected to all the initial vertices in the graph and a super-sink T connected to all the outputs in the graph. Based on that timing graph which includes all possible timing arcs, the leakage power optimization formulation is as follows:

$$\begin{aligned}\min \ & -\sum_{v\in V}\lambda_v x_v \\ \text{s.t.} \ & a_s = 0 \\ & d_T = 0 \\ & a_T \le T_{\max} \\ & a_u + w_{uv} + d_v^u \le a_v, \forall e_{uv} \in E \\ & d_v^u = d_{v0}^u + \alpha_v^u (x_v - L_{v0}) \\ & L_{v,min} \le x_v \le L_{v,max}\end{aligned} \tag{1.17}$$

where x_v is the length of gate v, initially of length L_{v0} and is constrained to be within $L_{v,min}$ and $L_{v,max}$. λ_v is the power/delay sensitivity coefficient of gate v obtained from timing and leakage libraries and computing ratios of changes in power over the corresponding changes in delay. Lookup tables store the power and delay characterization of the gates so the sensitivities can be readily computed. The arrival time and parameters that contribute delay along a path are depicted in Fig. 1.5. The arrival times at the super-source, super-sink and intermediate gates are a_S, a_T, and a_v, respectively. The delay of the super-sink d_T is zero and the delay along a timing arc from v to u is denoted d_v^u and d_v^{u0} is the initial delay. The net delay from u to v is denoted w_{uv}. Finally, α_v^u is a delay/length sensitivity coefficient.

A different objective function is how the authors consider timing violations and leakage power at the same time. The objective function in this case is

$$\min \gamma a_T - \sum_{v\in V}\lambda_v x_v \tag{1.18}$$

where the change in leakage power summation and a new term γa_T. γ is a user-specified parameter used to give more weight to the delay than leakage power. The same constraints as in (1.17) are used with on exception, the constraint involving the maximum delay is reversed.

$$a_T \ge T_{\max} \tag{1.19}$$

where $T_{\max}$ is now the desired lower bound on the circuit delay. The optimizer will try to reduce a_T to this lower bound if the cost in leakage power does not outweigh the benefit. The parameter γ tells the optimizer how much more important

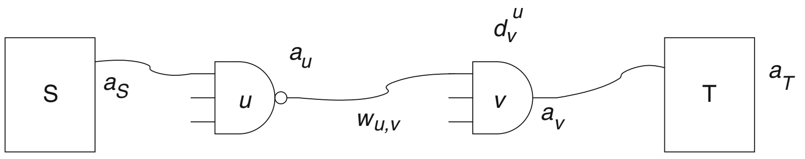


Fig. 1.5 Depiction of the delay terms using in the formulation in (1.17)

circuit delay is to leakage power. This is the most common technique for combining multiple objectives in optimization formulations. Refer to the multiple optimization preliminaries in Sect. 1.2.5 for more details.

The authors explain that commercial tools have been reluctant to adopt linear models out of fear for modeling inaccuracies and slew change effects. However, the results show that the proposed linear program outperforms commercial tools in several categories. A main reason is because the formulation is able to connect values from the golden timing engine into the formulation removing the miscorrelation between internal and golden timers seen in commercial tools. When considering gate length biasing, the permissible range of gate lengths is reduced to ± 5 nm. In this scenario, commercial tools can obtain better leakage power with degraded timing but the LP formulation improves timing and leakage power using either objective function. The scenario considering multi-length gates allows 3 gate lengths 50, 60 and 70 nm, starting from a solution using all 60 nm gates. This requires rounding to the nearest gate length after solving the LP formulations, but the results still improve over commercial tools despite the introduction of rounding errors. These benefits are obtained with a runtime nearly 1/5th that of commercial tools.

1.4 Geometric Programming Formulations

Geometric Programming (GP) is a type of convex optimization problems that can convert nonlinear and non-convex problems into convex forms and solve them efficiently using convex optimization techniques where the obtained solution is guaranteed to be globally optimum. The gate sizing problem, as well as closely related transistor, device, and wire sizing problems, can be formulated as Geometric Programs (GPs) [5, 14, 24]. In these works, one of area, power, or delay is chosen as the objective and the remaining objectives are made into constraints along with minimum gate size constraints.

In [11], a transistor sizing algorithm, TILOS (TImed LOgic Synthesizer), is presented that combines synchronous timing analysis with convex optimization techniques. TILOS considers three different optimization problems for transistor sizing. When a circuit needs to be fit in a system with a given clock period, the problem is formulated as minimizing area subject to circuit delay constraint. Minimizing circuit delay subject to circuit area is used when the circuit is desired to be as fast as possible with a limit on silicon area. The third problem is used when minimizing both circuit delay and area is important but relative weights are assigned to them.

To develop a delay model, transistors are modeled by a distributed Resistance-Capacitance (RC) model. Since these resistance and capacitance are convex functions of transistor sizes, the delay model is also a convex function. Then, all the three optimization problems can be efficiently solved using convex optimization techniques. This means that any solution is guaranteed to be globally optimum.

TILOS includes a static timing analyzer that identifies the latches and extracts all relevant timing paths in a circuit. By combining transistor sizing and static timing analysis, the need for the designer to determine which paths need to be optimized is eliminated. The designer just specifies the desired behavior of I/O signals, including clock signal, and TILOS determines what paths are to be optimized according to the static timing analysis. TILOS sizes not only the transistors in the combinational gates but also the ones in the latches. In addition, it organizes the subnetworks in complex gates based on the timing metrics so that the subnetworks with earlier arriving inputs are closer to the power supply.

In [22], circuit area is shown to be correlated to the power and the problem of gate sizing is solved by minimizing the area while the timing factors are dealt with as constraints:

$$\begin{aligned} \min \ & \text{Area}(\mathbf{x}) \\ \text{s.t.} \ & \text{Delay}(\mathbf{x}) \leq \text{T}_{\text{delay}} \\ & x_{w_g} \geq w_{\min}, x_{l_g} \geq l_{\min}, \forall g \in G \end{aligned} \tag{1.20}$$

where $\mathbf{x}$ is the vector of all lengths and widths of the gates. $[x_{w_g}, x_{l_g}]$ are the width and length of gate g belonging to the set of all gates, G, in the combinational circuit. Minimum length and width of the gates are represented by l_{min} and w_{min}, respectively which depend on the technology process. The arrival time of the signal at the end of the combinational circuit is shown by Delay$(\cdot)$. To ensure the functionality of the integrated circuit, the arrival time of the combinational circuit is required to be less than T_{delay} which is also known as Required Arrival Time (RAT).

This problem is then converted to the GP format to achieve a globally optimal solution as explained below: Area of each gate is found by multiplying the width and length of that gate. The objective is total gate area and is calculated by taking the summation of gate areas as:

$$\text{Area}(\mathbf{x}) = \Sigma_{g \in G} x_{l_g} x_{w_g},$$

which is a posynomial.

In delay calculations, Delay$(\mathbf{x})$ is found using Elmore delay model [9]. For an RC tree, the Elmore delay is the summation of the resistance of each segment times the downstream capacitance. When a buffer or a gate is inserted in an RC tree, it isolates the upstream and downstream capacitances. For simplicity, an example of a buffered circuit is considered in Fig. 1.6. A buffer b_i is modeled as an input capacitance, $c_{b_i,in}$, output resistance, r_{b_i}, and an intrinsic buffer delay, d_{b_i}, which is the product of buffer's output resistance and output capacitance, $c_{b_i,out}$, i.e., $d_{b_i} = r_{b_i} c_{b_i,out}$. In terms of the length and width of b_i, $x_{l_{b_i}}$ and $x_{w_{b_i}}$, the resistance and capacitances of b_i are:

$$r_{b_i} = \gamma_1 \frac{x_{l_{b_i}}}{x_{w_{b_i}}}, \qquad c_{b_i} = \gamma_2 x_{l_{b_i}} x_{w_{b_i}} + \gamma_3, \tag{1.21}$$

where $\gamma_1, \gamma_2, \gamma_3$ are technology dependent parameters which are always positive. Hence, the expressions for resistance and capacitance are posynomials. In Fig. 1.6, a

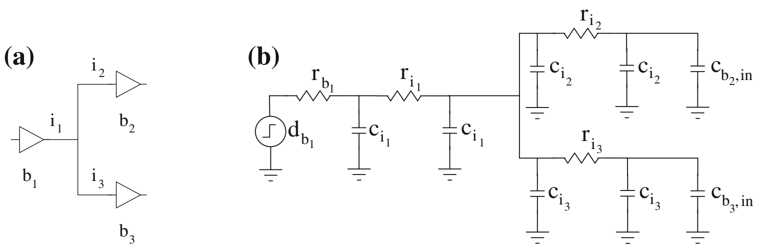


Fig. 1.6 An example of a buffered circuit and its equivalent RC network. **a** Buffered tree segment. **b** Equivalent RC network

buffered circuit and its equivalent RC network are shown. As an example, the delay from the input of buffer b_1 to the input of buffer b_2 is given by:

$$d_{b_1} + r_{b_1}C_{b_1} + r_{i_1}C_{i_1} + r_{i_2}C_{i_2},$$

where C_i denotes the capacitance downstream of the resistance of element i. Since the resistance and capacitance of gates, buffers and wires are posynomial functions of length and width and the expression for path delay is formed by multiplication and addition, the Elmore delay model is a posynomial. Although more accurate delay models are available and have proven to be effective [14], Elmore delay model is often used since it gives a reasonable delay approximation with a relatively low computational complexity.

Finally, in [22], the gate sizing problem of (1.20) is converted to GP standard format as:

$$\begin{aligned} \min\ & \text{Area}(\mathbf{x}) = \Sigma_{g\in G} x_{l_g} x_{w_g} \\ \text{s.t.}\ & \text{Delay}(\mathbf{x})\text{T}_{\text{delay}}^{-1} \leq 1 \\ & w_{\min} x_{w_g}^{-1} \leq 1, l_{\min} x_{l_g}^{-1} \leq 1, \forall g \in G \end{aligned} \tag{1.22}$$

In [5], the digital circuit gate sizing problem is formulated as a geometric programming problem. The digital circuit is assumed to be a set of digital gates. A variable $x_i \geq 1$ is associated with each gate representing a scale factor for the size of the transistors it is made from. These scale factors are considered as the optimization variables of digital gate sizing problem. Digital circuit area and power consumptions are affected by these scale factors and are formulated as:

$$A = \Sigma_{i=1}^{n} a_i x_i$$

$$P = \Sigma_{i=1}^{n} f_i e_i x_i,$$

where a_i is the area of gate i with unit scaling, f_i is the frequency of transition of gate i, and e_i is the power consumption when gate i transitions. These two functions are both posynomials and can be used to formulate the digital gate sizing problem in geometric programming format.

Circuit delay is another factor required to be optimized in digital gate sizing. To formulate the circuit delay, first, input capacitance (C_i) and driving resistance (R_i) of each gate are formulated in [5]:

$$C_i = \alpha_i + \beta_i x_i$$

$$R_i = \frac{\gamma_i}{x_i},$$

where α_i, β_i, and γ_i are technology dependant parameters. Then, the circuit delay (D_i) of gate i is modeled as:

$$D_i = \begin{cases} R_i \Sigma_{j \in F(i)} C_j, & \text{if } i \text{ is not an output gate,} \\ R_i C_i^{out}, & \text{if } i \text{ is an output gate,} \end{cases} \tag{1.23}$$

where C_i^{out} is the load capacitance of an output gate i and the set of all gates connected to the output of gate i is represented by $F(i)$. Note that this delay equation is a posynomial.

In the next step, the path delays are calculated using the gate delay equation. The path delays are posynomials since they only include summation of posynomials and monomials. Considering that the speed of a digital circuit is significantly affected by its worst-case path delay, the maximum path delay (D) of the circuit, which is also a posynomial, is chosen as the target of minimization and the digital gate sizing problem is formulated as:

$$\begin{aligned} \min\ & D \\ \text{s.t.}\ & A \le A_{\max} \\ & P \le P_{\max} \\ & x_i \ge 1, i = 1 \ldots n, \end{aligned} \tag{1.24}$$

where $A_{\max}$ and $P_{\max}$ are the upper bounds on total area and total power consumption, respectively, specified by the circuit designer. Since the objective and all the constraints are posynomials, the resultant optimization problem is a geometric programming problem that can be converted to a convex format and solved using convex optimization techniques. This formulation can be used to size any digital circuit whose gates are connected without forming a loop, i.e., there is not any path that starts and ends at the same gate. If there is a loop in the circuit topology, the path delays cannot be formulated as posynomials and the optimization problem will not be a GP.

An efficient and rapid algorithm for digital gate sizing of circuits with very large number of gates (around 1,000,000 gates) is developed in [13]. The goal is to determine the scale factors of all the digital gates while minimizing the circuit area subject to delay constraints. First, the optimization problem is solved using a reasonable delay approximation model such as the one presented in (1.23). This gives an initial solution that is feasible and is a good starting point for a more accurate delay optimization method. Then, a nonlinear back substitution method is proposed to convert the gate sizing optimization problem into an unconstrained problem. Since the objective of resultant unconstrained problem is not differentiable, it is solved repeatedly using a pseudo-Newton method that approximates the Hessian of the unconstrained problem to calculate the search direction for Newton method.

The proposed algorithm is as follows:

```
1. Compute a feasible initial solution $x$.
2. Repeat:
   2.a. Calculate the gradient, $g$, of the unconstrained problem.
   2.b. Approximate the Hessian, $H$, of the unconstrained problem.
   2.c. Find a search direction, $\Delta x$, by solving $H\Delta x = -g$.
   2.d. Compute the step size, $s$, using backtracking line search.
   2.e. Update the solution, $x = x + s\Delta x$, by solving $H\Delta x = -g$.
3. Until the stopping criteria satisfied.
```

This algorithm has a linear complexity with a finite number of iterations. The efficiency of the algorithm is significantly affected by how fast the search direction is calculated and how accurate this direction is to get to the optimal solution in a reasonable amount of time. The numerical results demonstrate the linear complexity of this algorithm as a circuit with 100,000 gates is sized in four minutes while a circuit with a million gates is solved in approximately 40 min.

Clock network buffer sizing is a special case of digital gate sizing [21]. It is however different with digital gate sizing since it only deals with determining the sizes of inverters (called buffers in clock network terminology) in the clock network rather than other logic gates in the circuit. In Fig. 1.7a, a clock problem instance is presented and in Fig. 1.7b, a clock network for distributing the clock signal is shown. The quality of a clock signal, measured by slew, degrades while going through the wires which may lead to lowering circuit performance. Therefore, buffers are inserted in clock networks to maintain clock signal quality. In Fig. 1.7c, a buffered

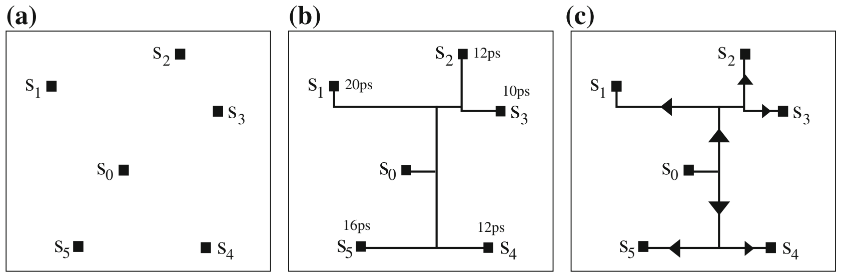


Fig. 1.7 An example clock network problem. **a** Clock problem instance. **b** Clock network. **c** Buffered clock networks

clock network is presented. Since inserting buffers generally increases the power consumption, buffer sizing is performed to minimize the power consumption while maintaining the clock signal quality. Bigger buffers can drive the signal for a longer distance in the wires but consume more power.

Another objective in clock network buffer sizing that makes it different with digital gate sizing is skew that is the maximum difference between the arrival times of clock signal at clock sinks. Unlike gate sizing where it is desired to minimize the path delays, in clock network buffer sizing, the difference between these delays is to be minimized. Skew is usually a competing objective with power consumption. Then, the clock network buffer sizing problem can be formulated as:

$$\begin{aligned} \min\ \text{Area}(\mathbf{x}) &= \sum_{b \in B} x_{w_b} x_{l_b} \qquad (1.25) \\ \text{s.t.}\ \ \max\{d_i(\mathbf{x}) - d_j(\mathbf{x})\} &\le t_{\text{skew}},\ \forall i, j \in S,\ i \neq j \\ \text{slew}_k(\mathbf{x}) &\le t_{\text{slew}}\ ,\ \forall k \in B \cup S \\ l_{\min} &\le x_{l_b},\ \ b \in B \\ w_{\min} &\le x_{w_b},\ \ b \in B \end{aligned}$$

where $\mathbf{x} = [\mathbf{x_w}; \mathbf{x_l}]$, and $\mathbf{x_w}$ and $\mathbf{x_l}$ are vectors representing buffer widths and lengths of the set of all buffers, B, in the clock tree. Area$(\mathbf{x})$ is the total area of the buffers. $d_i(\mathbf{x})$ is the delay of clock signal from clock source to clock sink i in the set of all sinks S that is calculated using Elmore delay model. t_{skew} is the target clock skew specified by the designer, and $l_{\min}$ and $w_{\min}$ are the minimum length and widths of a buffer that are technology dependant. The slew at each node k in the set of all nodes of the clock tree, i.e., clock sinks and buffers, is shown by slew$_k(\mathbf{x})$ and is calculated using a scaled version of Elmore delay model. Maximum allowed slew in the clock tree is specified by the designer and is represented by t_{slew}.

This optimization problem in its natural form is a nonlinear and non-convex problem and is very hard to solve. In addition, the solution to the problem can vary significantly depending on the initial starting point of the algorithm and the method used for solving. However, in [21], this problem is relaxed into geometric programming format and is solved efficiently using convex optimization technique. To relax this problem to geometric programming, the objective and the constraints are turned into either monomials or posynomials. This is accomplished as follows:

- Objective: Since the objective function is the summation of positive quadratic polynomials, it is already in geometric programming format and no transformation is needed.
- Skew constraints: Each skew constraint ensures that the maximum difference between the delays of two sinks is less than the maximum allowed skew. Clock signal delay, $d_i(\mathbf{x})$, is reasonably approximated using Elmore delay model which is a posynomial model. The maximum operator is equivalently formulated using a set of constraints instead of a single constraint. The negative coefficient resulting from subtraction needs to be relaxed. In the first step, $d_j(\mathbf{x})$ is replaced by the min-

imum sink delay, $d_{\min}$, resulting in $\max\{d_i(\mathbf{x})\} \le t_{\text{skew}} + d_{\min}$. This substitution addresses the problem with negative coefficient since $d_{\min}$ is now a constant and is taken to the right side of inequality, and makes the constraint tighter.
The maximum operator is eliminated by defining a dummy variable μ where $\mu \le t_{\text{skew}} + d_{\min}$ and adding a set of constraints $d_i(\mathbf{x}) \le \mu$, $\forall i \in S$.

- Slew constraints: In [21], slew at each node is approximated using a scaled version of Elmore delay model which is a posynomial and is already in geometric programming format. In this slew model, the slew at a node in the clock network is proportional to the Elmore delay from the previous upstream buffer output to that node as formulated below:

$$\text{slew}_k(\mathbf{x}) = ln(9) \times d_k^s(\mathbf{x}),$$

  where $d_k^s(\cdot)$ is the delay from the output of the previous upstream buffer to node k. The program runtime is lowered by deriving slew values from the delay values calculated for the skew constraints.
- Boundary constraints: Minimum length and width constraints are affine functions and already in geometric programming format.

Applying the above geometric programming relaxations, the clock network buffer sizing is formulated as a geometric programming in [21]:

$$\begin{aligned} \min \ \text{Area}(\mathbf{x}) = & \sum_{b \in B} x_{w_b} x_{l_b} \\ \text{s.t. } & \mu \, (t_{\text{skew}} + d_{\min})^{-1} \le 1 \\ & d_i(\mathbf{x})\mu^{-1} \le 1, \ \forall i \in S \\ & \text{slew}_k(\mathbf{x}) t_{\text{slew}}^{-1} \le 1 \ , \forall k \in B \cup S \\ & l_{\min} x_{l_b}^{-1} \le 1, \ b \in B \\ & w_{\min} x_{w_b}^{-1} \le 1, \ b \in B \end{aligned} \tag{1.26}$$

where μ is a dummy variable used to relax the problem into geometric programming format. $d_{\min}$ is a constant representing the minimum sink delay. This problem is solved iteratively where in each iteration, an updated minimum sink delay, $d_{\min}$, is used. Convex optimization techniques are used to solve this problem and ensure the solution found is the globally optimal solution. Finally, it should be mentioned that only clock networks without any cycles can be solved using this technique since the networks with at least a cycle in them cannot be formulated in geometric programming format.

1.5 Variation-Aware Formulations

An emerging issue in digital circuit design is process variation introduced because of advanced process technologies. Designing modern digital circuits under ideal conditions and assuming all gates, buffers and wires have nominal design sizes is not sufficient. This is because after fabrication, sizes can be up to 20 % different from the nominal design sizes [23]. Therefore, these process variations need to be addressed at the design stage to improve the reliability of digital circuits. Robust optimization is an optimization tool for incorporating uncertainties in problem variables and parameters that takes a pessimistic approach and considers the worst-case variations.

In [23], the maximum delay minimizing gate sizing problem with area constraint is solved using robust optimization while process variations are introduced in the gate sizes (width and length). This technique is developed for robust gate sizing and cannot be applied to the special case of clock network buffer sizing where skew is another constraint. This work is further developed in [21] to handle robust clock network buffer sizing. The clock network buffer sizing problem in (1.25) is formulated as a geometric programming problem that can be solved using convex optimization techniques as explained in detail in Sect. 1.4. This problem is deterministic and does not consider process variations. In [21], this problem is further developed to incorporate the uncertainties in buffer sizes due to process variations using robust optimization.

Process variations cause changes to the length and width of buffers. These changes are shown to be spatially correlated in [21], meaning that the variations in buffer sizes are related if two buffers are close to each other. A matrix **P** is defined as the covariance matrix in the length and width of the buffers. The covariance matrix, **P**, represents the intra-die correlation between buffer size variations, e.g., if two buffers are located close to each other, the variation in their sizes will be closer than the buffers that are located at a distance. Covariance matrix **P** is calculated using a grid-based model originally proposed in [23]. Once the covariance matrix **P** is obtained, an uncertainty set U is defined as the ellipsoid: $U = \{\mathbf{u} + \mathbf{P}^{1/2}\mathbf{v} \mid ||\mathbf{v}||_2 \leq \mathbf{1}, \quad \mathbf{v} \in \Re^{2|B|}\}$, where the nominal value of the width and lengths are represented by **u**. **v** is a vector with norm smaller than one.

Process variations significantly affect the timing metrics, i.e., slew and skew, of the clock networks. This is very important since the skew and slew affect the circuit performance and failing to meet these constraints may lead to circuit failure. Therefore, in [21], it is aimed to consider the effects of uncertainties in buffer lengths and widths only in the skew and slew constraints.

In order to formulate the problem which includes the uncertainties, vector **x** in (1.25) is replaced by an uncertain vector of variables $\tilde{\mathbf{x}}$ in the skew and slew constraints. The new skew and slew constraints are rewritten as:

$$\begin{aligned}
&\max\{d_i(\tilde{\mathbf{x}}) - d_j(\tilde{\mathbf{x}})\} \leq t_{\text{skew}}, \ \forall i, j \in S, \ i \neq j \\
&\text{slew}_k(\tilde{\mathbf{x}}) \leq t_{\text{slew}} \ , \forall k \in B \cup S
\end{aligned}$$

First, $d_j(\tilde{\mathbf{x}})$ in the skew constraints is replaced by $d_{\min}$ taken to the right hand side to eliminate the negative operator. Then, $\tilde{\mathbf{x}}$ is replaced by $\mathbf{x} + \delta\mathbf{x}$ and, we will have:

$$\begin{aligned} d_i(\mathbf{x} + \delta\mathbf{x}) &\leq (t_{\text{skew}} + d_{\min}),\ \forall i \in S \\ \text{slew}_k(\mathbf{x} + \delta\mathbf{x}) &\leq t_{\text{slew}}\ , \forall k \in B \cup S \end{aligned}$$

where $\delta\mathbf{x}$ is the ellipsoidal uncertainty vector defined by $\delta\mathbf{x} = \mathbf{P}^{1/2}\mathbf{v},\ ||\mathbf{v}|| \leq 1$. Finally, these equations are linearized by applying first order Taylor series approximation and considering a worst-case robust optimization where only worst-case uncertainty is considered:

$$\begin{aligned} d_i(\mathbf{x}) + \max\{\nabla_{\mathbf{x}}^T d_i(\mathbf{x})\delta\mathbf{x}\} &\leq (t_{\text{skew}} + d_{\min})\ ,\ \forall i \in S, \\ \text{slew}_k(\mathbf{x}) + \max\{\nabla_{\mathbf{x}}^T \text{slew}(\mathbf{x})\delta\mathbf{x}\} &\leq t_{\text{slew}}\ , \forall k \in B \cup S \end{aligned}$$

To formulate these constraints so that they fit into geometric programming format, the reformulation of skew constraints is explained in the following. The slew constraints also follow the same procedure. The skew constraints after Taylor series expansion can be rewritten as:

$$d_i(\mathbf{x}) + \max_{||\mathbf{v}|| \leq 1} (\boldsymbol{\phi}_1^T \mathbf{P}^{1/2}\mathbf{v} + \boldsymbol{\phi}_2^T \mathbf{P}^{1/2}\mathbf{v}) \leq t_{\text{skew}} + d_{\min}.$$

where $\boldsymbol{\phi}_1$ are the positive terms of $\nabla_{\mathbf{x}}^T d_i(\mathbf{x})$ and $\boldsymbol{\phi}_2$ are the negative terms. It should be noted that the uncertainties are just considered in the skew and slew constraints, as considering them in the objective can produce a more conservative power solution than is desired. These robust skew constraints consider the process variations in the buffer lengths and widths. However, they still need to be relaxed into geometric programming format to be solved efficiently. Therefore, the maximum operation is relaxed by defining two robust variables as $\mathbf{r}_1 = ||\mathbf{P}^{1/2}\boldsymbol{\phi}_1||,\ r_2 = ||\mathbf{P}^{1/2}\boldsymbol{\phi}_2||$, $\boldsymbol{\phi}_1^T\mathbf{P}\boldsymbol{\phi}_1\mathbf{r}_1^{-2} \leq 1$ and $\boldsymbol{\phi}_2^T\mathbf{P}\boldsymbol{\phi}_2\mathbf{r}_2^{-2} \leq 1$. The robust slew constraints are derived using a similar approach.

The robust buffer sizing problem is then formulated as:

$$\begin{aligned} \min\ \text{Area}(\mathbf{x}) &= \sum_{b \in B} x_{w_b} x_{l_b} \qquad (1.27) \\ \text{s.t.}\ & d_i(\mathbf{x}) + \mathbf{r}_{1_i} + \mathbf{r}_{2_i} \leq t_{\text{skew}} + d_{\min}, \forall i \in S \\ & \boldsymbol{\phi}_{1_i}^T \mathbf{P}\boldsymbol{\phi}_{1_i}\mathbf{r}_{1_i}^{-2} \leq 1, \forall i \in S \\ & \boldsymbol{\phi}_{2_i}^T \mathbf{P}\boldsymbol{\phi}_{2_i}\mathbf{r}_{2_i}^{-2} \leq 1, \forall i \in S \\ & \text{slew}_k(\mathbf{x}) + \mathbf{r}_{3_k} + \mathbf{r}_{4_k} \leq t_{\text{slew}}\ , \forall k \in B \cup S \\ & \boldsymbol{\phi}_{3_i}^T \mathbf{P}\boldsymbol{\phi}_{3_i}\mathbf{r}_{3_i}^{-2} \leq 1, \forall i \in B \cup S \end{aligned}$$

$$\begin{aligned}
&\boldsymbol{\phi}_{4_i}^T \mathbf{P} \boldsymbol{\phi}_{4_i} \mathbf{r}_{4_i}^{-2} \leq 1, \forall i \in B \cup S \\
&l_{\min} x_{l_b}^{-1} \leq 1,\ b \in B \\
&w_{\min} x_{w_b}^{-1} \leq 1,\ b \in B.
\end{aligned}$$

Considering that this problem is in geometric programming format, it can be solved efficiently using the existing convex optimization techniques. In [21], it is shown that this formulation provides improvement in both skew and power consumption of clock networks and at the same time is resistant to the changes due to process variation, hence, providing a robust network operating at low power.

Another tool in the convex optimization belt for considering variations is stochastic programming. One work that uses chance constraints to reliably size gates under variations in delays is presented in [20]. The following linear delay model is used to begin with:

$$d_i = a_i - b_i s_i + c_i \sum_{j \in Fanout(i)} s_j \tag{1.28}$$

where d_i is the delay of a gate i, s_i is the size of gate i and a_i, b_i and c_i are parameters fit empirically through circuit simulation for every cell in the library. Higher accuracy can be achieved using piecewise linear approximations instead of a single linear approximation. Using three regions, the authors state that the error can be reduced to less than 5 %. Using the delay model, a deterministic LP for minimizing area is:

$$\begin{aligned}
&\min \sum s_i \\
&\text{s.t. } D_p = \sum_{i \in p} a_i - b_i s_i + c_i \sum_{j \in Fanout(i)} s_j \\
&\quad D_p \leq T_{\max}, \forall p \in \text{Paths}
\end{aligned} \tag{1.29}$$

where $T_{\max}$ is the maximum allowed delay of the circuit and Paths is the set of all paths through the circuit. D_p is the delay of path p.

Variability can be captured by the b and c coefficients in the delay model. The maximum delay constraint is replaced with a probabilistic chance constraint

$$P(D_p \leq T_{\max}) \geq \eta, \forall p \in \text{Paths} \tag{1.30}$$

where η is the required probability of the path constraint to be satisfied. In reality each η would be different for each path but because of strong correlations within combinational blocks all values are equal and set to the desired timing yield. The chance-constrained formulation for minimizing area is:

$$\begin{aligned} \min \ & \sum s_i \\ \text{s.t.}\ & D_p = \sum_{i \in p} a_i - b_i s_i + c_i \sum_{j \in Fanout(i)} s_j \\ & P(D_p \leq T_{\max}) \geq \eta, \forall p \in \text{Paths}. \end{aligned} \tag{1.31}$$

Assuming normally distributed process variations, the path delays are also normally distributed. A normal distribution is characterized by its mean and variance. The mean delay of a gate in this case is

$$\bar{d_i} = \mathbf{E} d_i = a_i - \mathbf{E} b_i s_i + \mathbf{E} c_i \sum_{j \in Fanout(i)} s_j \tag{1.32}$$

where $\bar{d_i}$ is the mean delay of the path and **E** is the expected value operator. The delay variance of a gate i is

$$\sigma_{d_i}^2 = s_i^2 \sigma_{b_i}^2 + \left(\sum_{j \in Fanout(i)} s_j \right)^2 \sigma_{c_i}^2 \tag{1.33}$$

where σ is a standard deviation. By using the z-value or standard score for η and a normal distribution, the formulation is equivalent to:

$$\begin{aligned} \min \ & \sum s_i \\ \text{s.t.}\ & D_p = \sum_{i \in p} \bar{d_i} + \phi^{-1}(\eta) \left(\sum_{i \in p} \sigma_{d_i}^2 \right)^{1/2} \\ & D_p \leq T_{\max}, \forall p \in \text{Paths} \end{aligned} \tag{1.34}$$

where $\phi^{-1}(\eta)$ is the z-value for η. The formulation can be solved using convex optimization techniques. (The formulation specifies a second-order cone program [6]) However, the constraints are for each path and because the number of paths grows exponentially with the number of logic levels in the circuit, it can become intractable for large circuits. Instead a node-based formulation is derived which exhibits much better scalability. The following node-based delay constraint is derived in the work

$$\text{AT}_k \geq \text{AT}_i + \bar{d_i} + \phi^{-1}(\eta) \sqrt{s_i^2 \sigma_{b_i}^2 + \left(\sum_{j \in Fanout(i)} s_j \right)^2}, \forall i \in Fanin(k) \tag{1.35}$$

where AT_i is the arrival time for gate i. The resulting chance-constrained node-based formulation is

$$\begin{aligned}
&\min \sum s_i \\
&\text{s.t. } \mathrm{AT}_k \geq \mathrm{AT}_i + \bar{d}_i + \phi^{-1}(\eta)\sqrt{s_i^2\sigma_{b_i}^2 + \left(\sum_{j \in Fanout(i)} s_j\right)^2}, \forall i \in Fanin(k) \qquad (1.36) \\
&\quad \mathrm{AT}_o \leq T_{\max}, \forall o \in \mathrm{PO}
\end{aligned}$$

where PO is the set of primary outputs of the circuit.

Because of the node-based formulation, the runtime compared to previous statistical methods considering variations was an order of magnitude faster. The runtime complexity is subquadratic in circuit size, which is reasonably scalable. In their experiment preserving the delay of the circuit, they were able to reduce area by 22-30 % without sacrificing delay. The results further show that considering variations becomes more important as the tightness of the circuit delay constraint increase, i.e., circuit delay decreases. This is usually the case when designing high-performance circuits.

Another objective that can be considered in gate sizing is bin yield loss (BYL). BYL refers to when circuits are fabricated but don't meet the timing specification and must operate at a lower frequency. The circuits that operate within a specific range of frequencies are binned together and sold at a lower price than bins of higher frequency. BYL aims to minimize the loss in revenue from the practice of binning.

The authors derive a geometric program using Elmore delay to minimize bin yield loss without making any assumptions about the process variation distributions. For brevity, the resulting formulation from the derivation is presented, expressed as a geometric program in exponential form:

$$\begin{aligned}
&\min \mathrm{BYL}(\mathbf{x}, \boldsymbol{\omega}) \\
&\text{s.t. } t_j + d_i(\mathbf{x}, \mathbf{E}\boldsymbol{\omega}) \leq t_i, \forall j \in Fanin(i), \forall i \qquad (1.37) \\
&\quad t_i(y_i) \leq T_{\max}, \forall i \in PO \qquad (1.38) \\
&\quad x_{\min} \leq e^{x_i} \leq x_{\max}, \forall i
\end{aligned}$$

where $\mathbf{x}$ is the size of each gate, $\boldsymbol{\omega}$ is the vector of varying parameters including effective channel length and thickness of transistor oxide. d_i is the delay of gate i which is using the certainty-equivalent to consider variation, and t_i is the arrival time of gate i. BYL is expressed as an expected value which is an inner product in the continous domain:

$$\mathrm{BYL}(\mathbf{x}, \boldsymbol{\omega}) = \int_{-\infty}^{\infty} v(\mathbf{x}, \boldsymbol{\omega})\mathrm{PDF}(\boldsymbol{\omega})\delta\boldsymbol{\omega} \qquad (1.39)$$

where $v(\cdot, \cdot)$ is the amount of violations for a given size and variation vector and PDF$(\cdot)$ is the probability distribution function of ω. The authors prove that this formulation is convex without making any assumption of the underlying distribution.

To compute BYL, the authors empirically evaluate the expected value of violations using different variation vectors and running static timing analysis. The results demonstrate that the formulation can be used to reduce BYL by up to 72 % with only 6 % increase in area. The authors show that BYL is highly correlated to traditional non-binned yield-loss and because of this, the approach is also effective at minimizing tradiational yield loss. When compared to a sensitivity-based approach, the BYL formulation results in 61 % improvement in yield-loss.

1.6 Multi-Objective Formulations

Designers usually have many competing objectives to balance when performing gate sizing at modern technology nodes. The background on multi-objective convex optimization mentioned that the most common approach for considering multiple objectives is to add in scalarization weights and add together the contributions of each objective. This practice has been around for a long time and an example of using scalarization weights is presented in (1.18) in Sect. 1.3 where leakage power and circuit delay are considered simultaneously in the objective. The remainder of this section focuses on more recent advances in multi-objective optimization in gate sizing.

Compromise programming is an effective method for solving convex multi-objective problems [15]. In this method, targets are set a priori for each objective and a weighted error norm of the objective components and their targets is minimized. Suppose a multi-objective problem with n objectives $f_1, f_2, \ldots, f_n$ with target values of $f_1^*, f_2^*, \ldots, f_n^*$. Then, a compromise programming problem is formulated as [15]:

$$\min \left(\Sigma_{i=1}^{n} a_i |f_i(x) - f_i^*|^p\right)^{\frac{1}{p}},$$

where a_i are the weights associated with each objective indicating how important it is for that objective to get close to its target value. Ideally, the target is set to the utopia point which is a point where all the objectives are at their minimum value and is usually an infeasible point. However, this point is not often known a priori. In [15], compromise programming is successfully used to provide a balanced compromise between power and delay objectives when solving the digital gate sizing problem.

Satisficing Trade-Off Method (STOM) is another method for solving problems with convex competing objectives [15]. This method tries to achieve an adequate solution with satisfactory criteria rather than obtaining an optimal solution. STOM is an interactive method that finds a solution close to the designer's desired solution. In this method, first a Pareto optimal solution is obtained and presented to the designer. The designer then chooses which objectives have satisfactory values at that solution, which ones need to be improved, and which ones can be relaxed. This is an interactive process meaning that these satisfactory levels, also called aspiration levels, are updated at each step until the desired solution is reached.

A multi-objective optimization problem with n objectives is generally formulated using STOM as [15]:

$$\begin{aligned} &\min\ \left(z + \beta \Sigma_{i=1}^{n} a_i \left(f_i(x) - f_i^*\right)\right) \\ &\text{s.t. } a_i(f_i(x) - f_i^*) \leq z,\ i = 1, 2, \ldots, n \end{aligned} \tag{1.40}$$

$$a_i = \frac{1}{f_i^M - f_i^m},\ i = 1, 2, \ldots, n \tag{1.41}$$

where f_i^M, f_i^m and f_i^* are the maximum value, the minimum value and the aspiration level set by the designer for objective i, respectively. Usually, a small value of β is chosen and the optimization problem is solved. If the acquired solution is satisfactory for the designer, STOM stops at that solution. Otherwise, the problem is solved again using new aspiration levels set by the designer. Note that STOM may be used to achieve the best solution if the appropriate goals for different objectives are provided by the designer. In [15], both STOM and compromise programming are used to solve digital gate sizing problem resulting in balanced trade-off among delay and power objectives.

A recent method for solving problems with multiple objectives that are in geometric programming format is proposed in [10] and is called self-tuning multi-objective framework. This framework is applied to the digital gate sizing and clock network buffer sizing problems and is successful in automatically choosing a balanced trade-off among the objectives. This method is significant as it addresses the problem of having to produce a Pareto surface that can be a very time-consuming endeavor. In this work, a weighted sum of the objectives is considered as the target of minimization while the weighting parameters in the weighted sum of the objectives are considered as optimization variables rather than constants. This allows the optimization solver to find not only optimal sizes for the lengths and widths of the gates and buffers, but also for the weights applied to different objectives.

Converting the weights from parameters to variables in a general multi-objective formulation can change the nature of the problem significantly. For example, if the multi-objective problem is a quadratic programming problem, considering the weights as variables will force the problem to become a third-order problem, which is no longer convex. However, in the case of a GP, multiplying further variables does not affect its convexity. The general form of self-tuning multi-objective framework for geometric programming is then formulated as [10]:

$$\begin{aligned} \min\ & \alpha_1 f_1(\mathbf{x}) + \cdots + \alpha_o f_o(\mathbf{x}) \\ \text{s.t.}\ & g_i(\mathbf{x}) = 1,\ i = 1, \ldots, m \\ & h_j(\mathbf{x}) \leq 1,\ j = 1, \ldots, n \\ & \alpha_1 \cdots \alpha_o = e \end{aligned} \tag{1.42}$$

where multi-objective variable weights are represented by $\alpha_1, \ldots, \alpha_o > 0$. The first two sets of constraints, $g_i(\mathbf{x}) = 1,\ i = 1, ..., m$ and $h_j(\mathbf{x}) \leq 1,\ j = 1, ..., n$ are the equality and inequality constraints of the original problem. The last constraint $\alpha_1 \cdots \alpha_o = e$ is added to make sure that variable weights, α, are dependent on each other and not susceptible to scaling. In multi-objective problems that are not in GP format, the weights are usually constrained to sum to one, i.e., $\alpha_1 + \cdots + \alpha_O = 1$. However, in GP format, posynomial equality constraints are not allowed. Therefore, in this work, it is proposed to replace the summation by the product $\alpha_1 \cdots \alpha_o = e$. When this constraint is converted to the log-sum-exp format by the GP solver, it will be converted to a set of optimization variables that sum to one. Therefore, this framework will become equivalent to the original multi-objective formulation. This framework is applied to solve the gate sizing problem and achieve a good compromise between delay and power consumption. Finally, the special case of sizing problem, clock network buffer sizing problem, is solved using the self-tuning framework to balance the trade-off among area, power consumption and skew while shown to be scalable.

1.7 Conclusions

In this chapter, recent advances in sizing digital circuits using convex optimization techniques were presented. A background on preliminaries of convex optimization and its different types was given.

Different types of convex optimization formulations that have been used in digital gate sizing were discussed. These include linear programming, geometric programming, robust optimization, stochastic programming, and multi-objective optimization formulations. These formulations were analyzed and their advantages and disadvantages were discussed.

By this chapter, we hope that we were able to inspire the reader to use convex optimization and propose new formulations to keep pace with the ever-changing challenges in gate sizing.

References

1. Ben-Tal, A., Nemirovski, A.: Robust solutions of linear programming problems contaminated with uncertain data. Math. Program. **88**, 411–424 (2000)
2. Berkelaar, M., Jess, J.: Gate sizing in MOS digital circuits with linear programming. In: Proceedings of the Conference on European Design Automation, EURO-DAC ’90, pp. 217–221 (1990)
3. Bertsekas, D.: Nonlinear Programming, edn. Athena Scientific (1999)
4. Bertsimas, D., Brown, D., Caramanis, C.: Theory and applications of robust optimization. SIAM Rev. **53**, 464–501 (2011)
5. Boyd, S., Kim, S.J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optim. Eng. **8**(1), 67–127 (2007)

6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
7. Chinnery, D., Keutzer, K.: Linear programming for sizing, vth and vdd assignment. In: Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005. ISLPED '05. pp. 149–154 (2005). doi:10.1109/LPE.2005.195505
8. Dyer, M., Stougie, L.: Computational complexity of stochastic programming problems. Math. Program. **106**(3), 423–432 (2006)
9. Elmore, W.: The transient response of damped linear networks with particular regard to wideband amplifiers. J. Appl. Phys. **19**(1), 55–63 (1948)
10. Farshidi, A., Rakai, L., Behjat, L., Westwick, D.: Optimal gate sizing using a self-tuning multi-objective framework. Integr. VLSI J. **47**(3), 347–355 (2014). (Special issue: VLSI for the new era)
11. Fishburn, J., Dunlop, A.: TILOS: A posynomial programming approach to transistor sizing. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 326–328 (1985)
12. Jeong, K., Kahng, A., Yao, H., Rakai, D.: Revisiting the linear programming framework for leakage power vs. performance optimization. In: Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design, pp. 127–134 (2009)
13. Joshi, S., Boyd, S.: An efficient method for large-scale gate sizing. IEEE Trans. Circuits Syst. I: Regul. Pap. **55**(9), 2760–2773 (2008)
14. Kasamsetty, K., Ketkar, M., Sapatnekar., S.: A new class of convex functions for delay modeling and its application to the transistor sizing problem [CMOS gates]. IEEE Trans. CAD **19**(7), 779–788 (2006)
15. Kashfi, F., Hatami, S., Pedram, M.: Multi-objective optimization techniques for VLSI circuits. In: Proceedings of ISQED, pp. 156–163 (2011)
16. Kim, I., de Weck, O.: Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Struct. Multidiscip. Optim. **29**(2), 149–158 (2005)
17. Luenberger, D.: Theory of Linear and Integer Programming. Springer, Berlin (2003)
18. Nemirovski, A., Juditsky, A., Lan, G., Shapiro, A.: Robust stochastic approximation approach to stochastic programming. SIAM J. Optim. **19**(4), 1574–1609 (2009)
19. Nguyen, D., Davare, A., Orshansky, M., Chinnery, D., Thompson, B., Keutzer, K.: Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization [logic ic design]. In: Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on, pp. 158–163 (2003)
20. Orshansky, M., Nassif, S., Boning, D.: Design for Manufacturability and Statistical Design. A Constructive Approach. Springer, Berlin (2007)
21. Rakai, L., Farshidi, A., Westwick, D., Behjat, L.: Variation-aware geometric programming models for the clock network buffer sizing problem. IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **33**(4), 532–545 (2014)
22. Sapatnekar, S., Rao, V., Vaidya, P., Kang, S.: An exact solution to the transistor sizing problem for cmos circuits using convex optimization. IEEE Trans. CAD **12**, 1621–1634 (1993)
23. Singh, J., Luo, Z., Sapatnekar, S.: A geometric programming-based worst case gate sizing method incorporating spatial correlation. IEEE Trans. Comput.-Aided Design **27**(2), 295–308 (2008)
24. Singh, J., Nookala, V., Luo, Z., Sapatnekar, S.: Robust gate sizing by geometric programming. In: Proceedings of DAC, pp. 315–320 (2005)
25. Srivastava, A., Sylvester, D., Blaauw, D.: Statistical Analysis and Optimization for VLSI: Timing and Power. Springer, Berlin (2005)
26. Wu, T.H., Davoodi, A., Linderoth, J.: GRIP: Global routing via integer programming. IEEE Trans. CAD **30**(1), 72–84 (2011)

Chapter 2
A Fabric Component Based Approach to the Architecture and Design Automation of High-Performance Integer Arithmetic Circuits on FPGA

Ayan Palchaudhuri and Rajat Subhra Chakraborty

Abstract FPGA-specific primitive instantiation is an efficient approach for design optimization to effectively utilize the native hardware primitives as building blocks. Placement steps also need to be constrained and controlled to improve the circuit critical path delay. Here, the authors present optimized implementations of certain arithmetic circuits and pseudorandom sequence generator circuits to indicate the superior performance scalability achieved using the proposed design methodology in comparison to circuits of identical functionality realized using other existing FPGA CAD tools or design methodologies. The Hardware Description Language specifications as well as the placement constraints can be automatically generated. A GUI-based CAD tool has been developed that is integrated with the Xilinx Integrated Software Environment for design automation of circuits from user specifications.

2.1 Introduction

With a significant increase in circuit complexity for Field Programmable Gate Array (FPGA)-based designs, even the most sophisticated Computer Aided Design (CAD) tools often result in circuit implementations with unsatisfactory performance and resource requirements owing to their inability to optimally exploit the underlying FPGA architecture. Also, the CAD tool is unable to place the technology mapped subcircuits at the desired locations on the FPGA fabric. To overcome these shortcomings, a designer needs to identify the basic building blocks of the circuit, and make an effort to optimally construct them from the hardware primitives available on the FPGA and ensure their proper placement. Hence, implementations derived through the standard automatic logic synthesis-based design flow starting with the behavioral or Register Transfer Level (RTL) Hardware Description Language (HDL) of the circuit can be

A. Palchaudhuri (✉) · R.S. Chakraborty
Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur, Kharagpur 721302, West Bengal, India
e-mail: ayanpalchaudhuri@gmail.com

R.S. Chakraborty
e-mail: rschakraborty@cse.iitkgp.ernet.in

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_2

outperformed by more low-level “custom” design techniques. This chapter discusses certain FPGA-specific design techniques which needs to be adopted for optimal realization of high performance circuits and presents a relevant case study.

2.1.1 Overview of FPGA Design Philosophy

Modern FPGA CAD tools facilitate the automatic mapping of binary addition logic and homogeneous wide AND and OR gates to carry-chain structures [1] to accelerate signal propagation. It can also infer *wide function multiplexers* native to an FPGA slice, thereby achieving speedup by avoiding switch-box routing to the extent possible. However, it fails to do all of this, as soon as the final carry outputs of individual slices [2] or Lookup Table (LUT) outputs are tapped out or registered to facilitate pipelining of the architecture. In such a scenario, the designer has to spell out special directives in the HDL of the design or in the associated “constraints files,” so that in the *packing* or *clustering* step (as known in the FPGA CAD literature [3]) of the FPGA design flow, the desired technology mapped circuit is efficiently “packed” into the available hardware resources.

Most current FPGA vendors allow direct instantiation of the available primitives in the HDL code [4], and also a “mixed” style of HDL coding, whereby high-level behavioral code is intermingled with relatively “low-level” structural code. Placement steps also need to be constrained and controlled, otherwise the technology mapped logic elements get unevenly distributed across the FPGA fabric, resulting in greater routing and critical path delay. Although most modern FPGA vendors provide special hardware IPs for common integer arithmetic operations that can be directly instantiated in the HDL code, we would demonstrate that we can do better by adopting careful design techniques.

With sufficient modularity in the circuit architecture, it becomes easy to automate the generation of the HDL and constraint files, which are themselves very regular in their grammar. Target FPGA-specific *primitive instantiation* is an effective approach for design optimization [5], and is often the only approach, or is simpler than rewriting the RTL code to coax the logic synthesis tool to infer the desired architectural components. In case the entire circuit is not amenable to primitive instantiation, the philosophy can be adopted to design those sub-circuits that are amenable, and contribute significantly to the critical path. The only disadvantage of using such a design methodology is that the design becomes less portable and harder to maintain. In spite of this, the methodology is very effective in practice, considering that (a) often the target FPGA platform is known before the circuit is designed, and, (b) FPGAs from a related family from the same vendor are often backward compatible regarding the design elements supported. For example, newer versions of FPGAs of the “Virtex” family from Xilinx are expected to support primitives supported in some older Virtex versions. Thus, the HDL code for instantiating primitives targeting the older versions, and the constraints file to control the placement, can be reused in the newer version after small tweaks, if necessary.

2.1.2 Existing FPGA CAD Tools

Xilinx IP Core Generator and *FloPoCo* are the most common FPGA CAD tools available. We shall now present the features, advantages, and limitations of each of these tools.

2.1.2.1 Xilinx IP Core Generator

The standard *Xilinx IP Core Generator* has a GUI-based utility through which synthesizable HDL code for common integer arithmetic circuits (both combinational and pipelined versions) can be automatically generated. User gives the parameter *latency* as input for pipelined architecture realizations. Although such HDL generated is functionally correct, it fails to give high performance when implemented, because the synthesis tool performs inefficient technology mapping and the inferred logic elements are scattered in an apparently random fashion across the FPGA fabric, thereby causing large routing delays. Xilinx also allows the direct instantiation of "Digital Signal Processing" (DSP) hardware macros in the HDL targeted for FPGAs, to implement certain common arithmetic operations, but they can be outperformed by the proposed circuits through maximal forward path pipelining at the cost of higher latency.

2.1.2.2 *FloPoCo* (Floating-Point Cores)

FloPoCo is an open-source C++ framework for generating arithmetic cores for FPGAs [6]. *FloPoCo* provides a command-line interface through which the user can input operator specifications, and the program generates the corresponding synthesizable HDL. The main features of *FloPoCo* as listed in [7] are as follows:

- Supports integer, fixed point, floating point, and *Logarithmic Number System* (LNS) arithmetic.
- Supports pipelining by allowing the user to specify the desired operating frequency.
- Allows the user to specify the target FPGA implementation platform, and generates synthesizable HDL code optimized for that target platform since *FloPoCo* can perform target platform specific pipelining. Its frequency-directed pipelining paradigm takes into consideration the timing information about the target FPGA [7].

However, detailed experimentation with the latest released version of *FloPoCo* (v 2.5.0) [6], and implementation and characterization of the integer arithmetic circuit descriptions generated by it indicate certain potential drawbacks:

- *FloPoCo* only generates pure behavioral HDL code which cannot correctly infer the desired hardware primitives of the target FPGA platform and also has no control over the inference and placement of logic blocks on the FPGA fabric. This makes the post-synthesis performance of the circuit worse than the user-specified target frequency. Thus, *FloPoCo* provides no guarantee that the target frequency specified would be met in the final implementation.

- *FloPoCo* at times create very deep pipelines apparently to meet input frequency constraints, but *post place and route* implementations do not guarantee that the delay constraints are met.
- Pipelining behavior of *FloPoCo* is very inconsistent. It was observed that for integer adder circuit implementations, *FloPoCo* creates very deep pipelines, whereas it creates fairly unbalanced and irregular pipelines for integer dual subtractor implementations, where each of the pipeline stages have different complexities. At the same time, *FloPoCo* completely fails to pipeline integer multiplier circuits. Our observations about these inconsistencies in the pipelining behavior of the current version of *FloPoCo* have been concurred with by the creators of *FloPoCo* through personal correspondence. They have acknowledged that a bug exists in their program, which they have filed and would probably be taken care of in future releases.

The important approach to note is that most or all of the existing options for automatic generation of arithmetic circuit cores targeting FPGAs are agnostic of the "low-level" architecture of the target FPGA platform. Consequently, they can be predicted to be unable to take advantage of the hardware primitives, and to generate the most optimal circuit descriptions for the target FPGA.

2.2 Architecture of Target FPGA Platform

The Configurable Logic Blocks (CLBs) of FPGAs are the main logic resources for implementing sequential as well as combinatorial circuits. A typical CLB of Virtex-5 FPGA contains 2 "slices," with each slice (called a "SLICEL" or "SLICEM") comprising of four 6-input LUTs, four flip-flops (FFs), three wide function multiplexers, and a length-4 carry chain comprising of multiplexers and XOR gates [8, 9] as shown in Fig. 2.1.

For both Virtex-5 and Virtex-6 FPGAs, the 6-input LUT can also be configured as a dual output LUT which can configure a 5 (or less)-input, 2-output logic function with shared inputs, thereby reducing the requirement in the number of LUTs from two to one for certain logic expressions. The LUTs present in SLICEL can implement arbitrary combinational logic, whereas the LUTs in SLICEM can additionally implement *distributed RAM* elements [9]. The carry chain represents the fast carry propagation logic and the LUTs in the slice can be optionally connected to the carry chain via dedicated routes to implement complex logic functionality [4].

Each storage element or FF present in the slice can be controlled using the set, reset, clock, and clock enable signals. Virtex-6 slice architecture is quite similar to Virtex-5 slice architecture, other than the fact that it offers four additional FFs per slice but each FF has one control signal less, i.e., it does not have independent set and reset pins compared to Virtex-5 architecture [10].

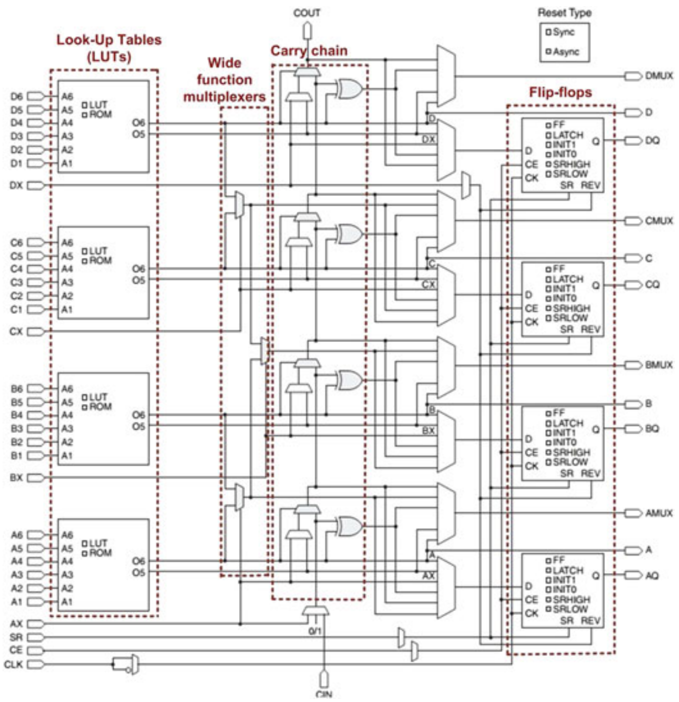


Fig. 2.1 Xilinx Virtex-5 slice architecture [9]

2.3 A Fabric Component-Based Design Approach for High-Performance Integer Arithmetic Circuits

Implementation of highly optimized arithmetic circuits targeted toward a specific FPGA family continue to remain a challenging problem as many fast arithmetic circuits proposed over the decades may not be amenable to a very optimized implementation. Often, the logic synthesis tools are unable to infer the desired native circuit components from the input HDL, as they explore only a small design space close to the input architectural description [11]. The logic synthesis algorithms are also unable to apply the logic identities and perform appropriate algebraic factoring and sub-expression sharing in many cases, especially when intermediate signals are tapped out [2] or registered to facilitate pipelining of the architecture. It is in general a nontrivial computational problem to decompose the Boolean equations describing the implemented circuit, to forms such that the sub-expressions can be mapped easily and efficiently to the fabric primitives on the target FPGA. It helps if the designer manipulates the Boolean equations *a priori* in the HDL, to forms that can be optimally mapped to the native target architecture by the CAD software.

A previous related work [12] had reported area requirements (for LUT-based FPGAs) in terms of the total number of k-input LUTs required to map a function of x variables, as

$$lut(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ 1 & \text{if } 1 < x \leq k \\ \lfloor \frac{x-k}{k-1} \rfloor + 2 & \text{if } x > k \text{ and } (k-1) \nmid (x-k) \\ \frac{x-k}{k-1} + 1 & \text{if } x > k \text{ and } (k-1) \mid (x-k) \end{cases} \tag{2.1}$$

The LUT estimation is based on the fact that LUTs are perhaps the principal components for implementing combinational logic in FPGAs, where combinational logic blocks with higher number of inputs are expected to be implemented by a cascade of LUTs, with permitted amount of parallel processing. However, for efficient designs, the designer must explore the additional logical capabilities that the LUTs of modern day FPGAs provide. In addition, the wide-function multiplexers and the carry chains can significantly reduce LUT requirements. The above closed-form expression in (2.1) for hardware estimation thereby suffers from the following limitations:

- It assumes that all LUTs provide single outputs, whereas modern FPGAs from Xilinx provide dual-output LUTs that can significantly reduce hardware cost, provided the logic functions to be mapped satisfy certain criteria.
- Certain logic expressions can be factored appropriately to form sub-expressions or manipulated such that they can be compactly realized using LUTs, wide function multiplexers, and carry chains, which can provide multiple outputs out of a single slice. The closed-form expressions in (2.1) possibly hint at an approximate upper bound on the number of LUTs required.
- The number of slices spanned by the logic elements give a more accurate estimate of the hardware overhead in comparison to LUT count.
  The philosophy behind estimating the LUT requirements [12] may not reflect its actual implementation on FPGA. For example, let us consider an 8:1 multiplexer, which is essentially an 11-input 1-output function.

$$\begin{aligned} f(s_2, s_1, s_0, a, b, c, d, e, f, g, h) = {} & s_2's_1's_0'a + s_2's_1's_0b + s_2's_1s_0'c + s_2's_1s_0d + s_2s_1's_0'e \\ & + s_2s_1's_0f + s_2s_1s_0'g + s_2s_1s_0h \end{aligned} \tag{2.2}$$

Going by (2.1), $lut(x) = 2$ for $k = 6$. This information indicates that the first six variables go as input to the first LUT and the remaining five variables along with the output of the first LUT go as input to the second LUT. However, on examining (2.2) closely, it can be observed that there is no possible way to decompose it to the following form comprising of two functions f_1 and f_2, which could have actually realized it using two LUTs:

$$f(s_2, s_1, s_0, a, b, c, d, e, f, g, h)$$
$$= f_2\underbrace{(f_1\overbrace{(x_1, x_2, x_3, x_4, x_5, x_6)}^{\text{implemented using 1 LUT}}, x_7, x_8, x_9, x_{10}, x_{11})}_{\text{implemented using 1 LUT}}$$

where each x_i is distinct and can be any one of the variables of the function f.

2.3.1 Guidelines for High-Performance Realization

We list certain simple but useful guidelines for compact and high-performance realization of circuits on modern high-end FPGAs from Xilinx:

1. A 6-input LUT can implement any arbitrary combinational logic function y, having a maximum of six inputs and a single output, like $y = f(x_1, \ldots, x_n)$, where $2 \leq n \leq 6$.
2. A 6-input LUT can implement any arbitrary five (or less)-input two-output function where each of the single-output functions may or may not have shared inputs. For example, consider two functions g and h, where $g = f(x_1, \ldots, x_n)$ with $X = \{x_1, \ldots, x_n\}$, and $h = f(y_1, \ldots, y_m)$ with $Y = \{y_1, \ldots, y_m\}$. Here, the sets X and Y are called the *support* [13] of the functions g and h. For packing g and h into a single LUT, either of the conditions must be satisfied:

   - $4 \leq |X| + |Y| \leq 5;$ if $X \cap Y = \emptyset$ (i.e., g and h are *orthogonal* functions)
   - $2 \leq |X| + |Y| \leq 10;$ if $X \cap Y \neq \emptyset$

   where $|X|$ and $|Y|$ are the cardinality of the support of the functions X and Y.
3. Let f be a Boolean function of n variables ($8 \leq n \leq 13$) which can be represented in the following form (see Fig. 2.2):

$$f(i_1, i_2, \ldots, i_n) = x_1' f_{x_1'} + x_1 f_{x_1}$$

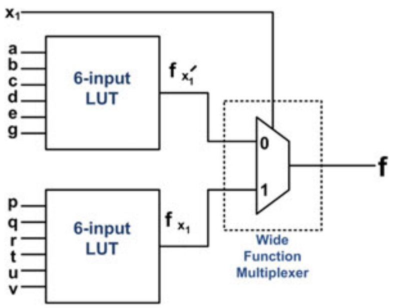


Fig. 2.2 Architecture mapping for Boolean logic that can be decomposed with respect to a single variable

Here, f_{x_1} and $f_{x_1'}$ are each six (or less)-input combinational functions that can be individually realized using one LUT each. The wide function multiplexer present in the same slice as that of the LUTs computes the final expression, as shown in Fig. 2.2. Equation (2.1) however evaluates to $lut(x) = 3$, where $x = x_{max} = 13$ (6 × 2 (two 6-input LUTs) + 1 (select line)) and $k = 6$. If there exists p functions of the form as in f, the design requires $\lceil p/2 \rceil$ slices, and $2p$ LUTs. An 8:1 multiplexer can be realized using this logic where the 6-input LUTs of Fig. 2.2 are configured as 4:1 multiplexers each (sharing the same select lines), and the wide function multiplexer selecting one of the LUT outputs.

4. Let f be a function of n variables ($17 \leq n \leq 26$) such that we can apply recursive decomposition twice on it as shown below:

$$\begin{aligned} f(i_1, i_2, \ldots, i_n) &= x_1' f_{x_1'} + x_1 f_{x_1} \\ &= x_1'(x_2' f_{x_1'x_2'} + x_2 f_{x_1'x_2}) + x_1(x_3' f_{x_1x_3'} + x_3 f_{x_1x_3}) \\ &= x_1'x_2' f_{x_1'x_2'} + x_1'x_2 f_{x_1'x_2} + x_1x_3' f_{x_1x_3'} + x_1x_3 f_{x_1x_3} \end{aligned}$$

Here, $f_{x_1'x_2'}$, $f_{x_1'x_2}$, $f_{x_1x_3'}$ and $f_{x_1x_3}$ are each 6 (or less)-input combinational functions that can individually be realized using one LUT each. Three wide function multiplexers present in the same slice as that of the LUTs computes the final expression as shown in Fig. 2.3. Equation (2.1) however evaluates to $lut(x) = 6$, where $x = x_{max} = 27$ (6×4 (four 6-input LUTs) + 3 (select lines)) and $k = 6$. If there exists p functions of the form as in f, the design requires p slices and $4p$ LUTs. A 16:1 multiplexer can thus be mapped in a single slice using four LUTs, and three wide function multiplexers.

5. Consider any expression R of the following form:

$$\begin{aligned} R = a'b + a[X] = a'b + a[c'd + c(Y)] &= a'b + a[c'd + c(e'f + e\{Z\})] \\ = a'b + a[c'd + c(e'f + e\{g'h + gi\})] \end{aligned} \tag{2.3}$$

where $X = c'd + cY$, $Y = e'f + eZ$ and $Z = g'h + gi$. Here i is a single input variable, and for every member of the pair (a, b), (c, d), (e, f) and (g, h), there can be either of the following possibilities: either both the members satisfy the criteria of being packed into a single LUT or the first member can be a six (or less)-input function and the second member can be a single variable. The above expression can be realized within a single slice using four LUTs and a carry chain, as shown in Fig. 2.4.

The Boolean logic (2.3) essentially represents a cascade of 2:1 multiplexer functions. Here, R can have a maximum of 29 (6×4 (four 6-input LUTs) $+$ 4 inputs $v_{4:1}$ external to the logic slice $+1$ external input to the bottom MUXCY of the carry chain) input variables. If the Boolean logic function to be realized is of the form such that the variable i in (2.3) can be substituted by another expression bearing a similar resemblance to (2.3), and in such a way, if a total of n such substitutions can be carried out only at the position of variable i, then the entire expression can be realized using $(n+1)$ slices and a maximum of $4(n+1)$ LUTs. From (2.1), we obtain $lut(x) = 6$, where $x = 29$ and $k = 6$ for which a minimum FPGA area of two slices are required. However, with the help of carry-chain fabric, the entire architecture can be compacted within a single slice. For example, a wide 24-input AND gate can be realized by the function R to fit the form of (2.3) as shown below:

$$
\begin{aligned}
R = a_1a_2\ldots a_{23}a_{24} &= \overline{a_{19}\ldots a_{24}}.0 + (a_{19}\ldots a_{24})[a_{18}a_{17}\ldots a_1a_0] \\
&= \overline{a_{19}\ldots a_{24}}.0 + (a_{19}\ldots a_{24})[\overline{a_{13}\ldots a_{18}}.0 + (a_{13}\ldots a_{18}) \\
&\quad [\overline{a_7\ldots a_{12}}.0 + (a_7\ldots a_{12})[\overline{a_1\ldots a_6}.0 + (a_1\ldots a_6).1)]]]
\end{aligned}
$$

Thus, going by Fig. 2.4, $a = a_{19}\ldots a_{24}$, $c = a_{13}\ldots a_{18}$, $e = a_7\ldots a_{12}$ and $g = a_1\ldots a_6$, $b = d = f = h = 0$, and $i = 1$. Hence each 6-input LUT realizes a 6-input AND gate and the outputs of the 6-input LUTs are AND-ed using the carry chain.
Example of a wide 24-input OR gate where the logic equation can be manipulated to fit the form of (2.3) as shown below:

$$
\begin{aligned}
R = a_1 + a_2 + \cdots + a_{23} + a_{24} &= (a_{19} + \cdots + a_{24}).1 + (\overline{a_{19} + \cdots + a_{24}})[a_{18} + \cdots + a_1] \\
&= (a_{19} + \cdots + a_{24}).1 + (\overline{a_{19} + \cdots + a_{24}})[(a_{13} + \cdots + a_{18}).1 + (\overline{a_{13} + \cdots + a_{18}}) \\
&\quad [(a_7 + \cdots + a_{12}).1 + (\overline{a_7 + \cdots + a_{12}})[(a_1 + \cdots + a_6).1 + (\overline{a_1 + \cdots + a_6}).0)]]]
\end{aligned}
$$

Thus, going by Fig. 2.4, $a = \overline{a_{19} + \cdots + a_{24}}$, $c = \overline{a_{13} + \cdots + a_{18}}$, $e = \overline{a_7 + \cdots + a_{12}}$ and $g = \overline{a_1 + \cdots + a_6}$, $b = d = f = h = 1$, and $i = 0$. Hence, each 6-input LUT realizes a 6-input NOR gate and the outputs of the 6-input LUTs are fed to the carry chain and a wide input OR gate is realized by following the absorption law $a + \overline{a}b = a + b$.
We discuss certain practical circuits where a wide input AND and OR gate are necessary for realization.

- Consider the design of a *priority encoder* which arbitrates among N units that are all requesting access to a shared resource. Access is to be granted to a single unit with highest priority decided by the LSB of the input. The corresponding logic equations can be described as

$$
\begin{aligned}
Y_0 &= N_0 \\
Y_i &= \underbrace{N_i \cdot \overline{N_{i-1}} \cdot \overline{N_{i-2}} \cdot \ldots \cdot \overline{N_2} \cdot \overline{N_1}}_{\text{wide input AND gate}} \quad \text{if } i \geq 1
\end{aligned}
$$

- An *incrementer* that adds 1 to an input word N can be described as

$$Y_0 = \overline{N_0}$$
$$Y_i = N_i \oplus \underbrace{(N_{i-1} \cdot N_{i-2} \cdot \ldots \cdot N_1 \cdot N_0)}_{\text{wide input AND gate}} \quad \text{if } i \geq 1$$

- A *decrementer* that subtracts 1 from an input word N can be described as

$$Y_0 = \overline{N_0}$$
$$Y_i = N_i \odot \underbrace{(N_{i-1} + N_{i-2} + \cdots + N_1 + N_0)}_{\text{wide input OR gate}} \quad \text{if } i \geq 1$$

- $K = A + B$ *Comparator* [15]
  To design a circuit to detect $A + B = K$, the usual approach followed by the FPGA CAD tool is to infer an adder that adds inputs A and B, and feed the sum and input K to an equality comparator. However, to significantly reduce hardware and computational overhead, a methodology was proposed in [15]. The key observation is the fact that if A and B are known, the carry into each bit to make $K = A + B$ can be determined. Thus, it is sufficient to check adjacent pairs of bits to verify that the carry-out produced by the previous bit and the carry-in required by the current bit are both the same. The truth Table 2.1 shows the *required* and *generated* carries. The required carry-in cr_{i-1} for bit i and the generated carry-out cp_{i-1} for bit $i - 1$ are obtained as $cr_{i-1} = A_i \oplus B_i \oplus K_i$ and $cp_{i-1} = (A_{i-1} \oplus B_{i-1})\overline{K_{i-1}} + A_{i-1} \cdot B_{i-1}$. Equality check for the ith bit position is performed using a single LUT as it can be computed using six distinct variables—A_i, B_i, K_i, A_{i-1}, B_{i-1} and K_{i-1}, by evaluating $EQ_i = cr_{i-1} \odot cp_{i-1}$. Final equality check is done using carry chain where all the outputs corresponding to equality checks at every bit position are AND-ed together.

$$EQ = \underbrace{EQ_j \cdot EQ_{j-1} \cdot \ldots \cdot EQ_i \cdot \ldots \cdot EQ_1 \cdot EQ_0}_{\text{wide input AND gate}}$$

  Additionally, we can obtain the following outputs from the XORCY gates of the carry chain: $L = a \oplus X$, $M = c \oplus Y$, $N = e \oplus Z$ and $O = g \oplus i$. The XORCY gates can compute the sum bits of an adder $s_i = p_i \oplus c_i$ where p_i is computed using LUT; $p_i = a_i \oplus b_i$. The carry-out bit of each stage is computed using MUXCY of carry chain; $c_{i+1} = \overline{p_i} a_i + p_i c_i$ (see Sect. 2.4.1.1 for details). Thus, an n-bit adder can be realized using $\lceil n/4 \rceil$ slices and a maximum of n LUTs.

6. LUTs of SLICEM can be configured as shift registers which are typically implemented for *Linear Feedback Shift Register* (LFSR) circuits. Each SLICEM LUT can be configured as a variable 1–32 clock cycle shift register [4] whose length

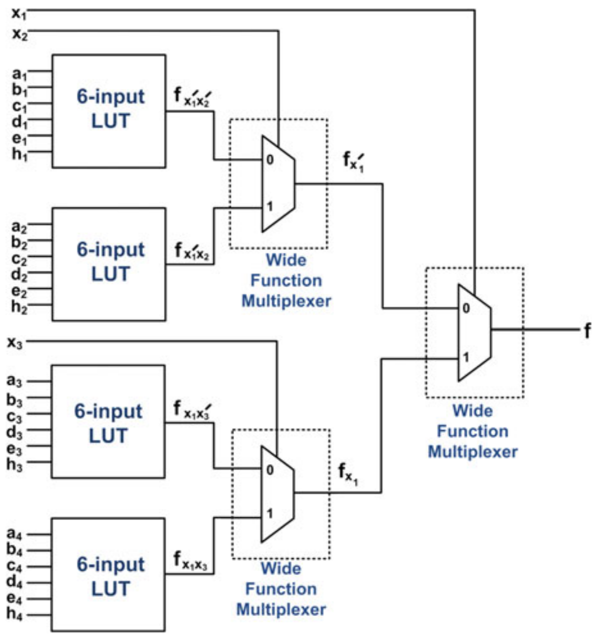


Fig. 2.3 Architecture mapping for Boolean logic that can be decomposed with respect to two variables

can be fixed, static, or dynamically adjusted by controlling $A[4{:}0]$ as shown in Fig. 2.5. The LUT can be described as a 32:1 multiplexer with the five inputs serving as binary select lines, and the values programmed into the LUT serves as the data being selected. Such LUTs can be cascaded with FFs and LUTs of other SLICEMs to realize greater shift lengths. Presence of these special LUTs reduce FPGA resource utilization compared to implementations using FFs only. Since each SLICEM in a Virtex-5 FPGA contains four LUTs and four FFs, a shift register of length 1–132 can be realized in a single slice; and a 1–136 clock cycle register can be realized in a single slice for Virtex-6 FPGAs as it contains four additional FFs. For realization of a shift register of length n, we require $\lceil n/132 \rceil$ slices with a maximum of $4\lceil n/132 \rceil$ LUTs and $4\lceil n/132 \rceil$ FFs for Virtex-5 FPGA, and $\lceil n/136 \rceil$ slices with a maximum of $4\lceil n/136 \rceil$ LUTs and $8\lceil n/136 \rceil$ FFs for Virtex-6 FPGA.

2.4 Architecture of Arithmetic Circuits

We present the pipelined implementations of a few important and widely used circuits with controlled placement on the fabric logic such that the critical path delay can be significantly reduced and the throughput increased.

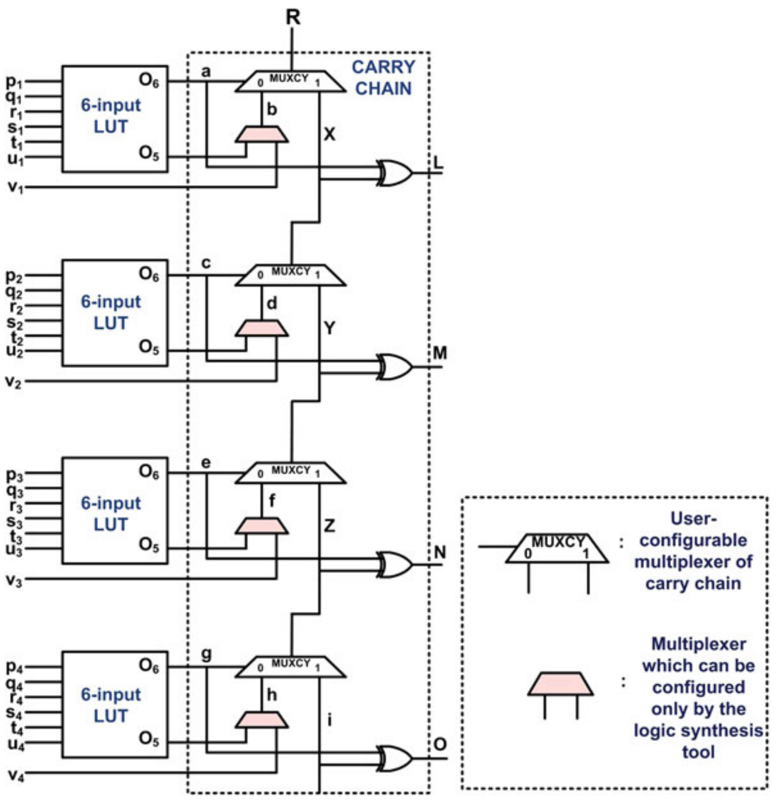

R
CARRY CHAIN
6-input LUT
O_6
O_5
MUXCY
a
b
c
d
e
f
g
h
X
Y
Z
i
L
M
N
O
p_1 q_1 r_1 s_1 t_1 u_1 v_1
p_2 q_2 r_2 s_2 t_2 u_2 v_2
p_3 q_3 r_3 s_3 t_3 u_3 v_3
p_4 q_4 r_4 s_4 t_4 u_4 v_4
User-configurable multiplexer of carry chain
Multiplexer which can be configured only by the logic synthesis tool

Fig. 2.4 Architecture mapping for Boolean logic that can exploit the carry chain

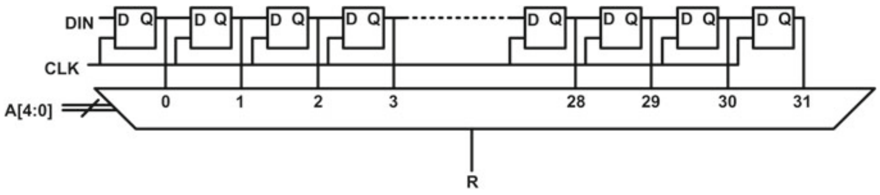


Fig. 2.5 Configuration of an LUT (of SLICEM) as a shift register

2.4.1 Integer Adder Architecture

2.4.1.1 Hybrid Ripple Carry Adder

A pipelined "hybrid ripple carry adder" (hybrid RCA), using the carry chain, LUTs and FFs available in a Xilinx slice, can be realized as shown in Fig. 2.6. The outputs of the "XORCY" gates provide the sum bits, whereas the output of each MUXCY

Table 2.1 Required and generated carries [14]

A_i	B_i	K_i	cr_{i-1} (required)	cp_i (produced)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

calculates the intermediate carries [16]. Latches can be inserted on the carry propagation path for pipelining the design. Thus, for an n-bit adder, $\lceil n/4 \rceil - 1$ pipeline stages are required. The LUTs compute the *propagate* function $p_i = a_i \oplus b_i$. Let $g_i = a_i b_i$ be the corresponding *generate* function. If c_i is the ith carry-in bit, the i-th sum bit can be calculated by XOR-ing the LUT and MUXCY outputs as

$$s_i = p_i \oplus c_i = a_i \oplus b_i \oplus c_i$$

The output of each MUXCY gate computes the ith carry-out as

$$\begin{aligned} c_{oi} &= g_i + p_i c_i = a_i b_i + (a_i \oplus b_i) c_i \\ &= (\overline{a_i} \overline{b_i} + a_i b_i) a_i + (a_i \oplus b_i) c_i = \overline{p_i} a_i + p_i c_i \end{aligned}$$

2.4.1.2 Xilinx DSP Slice-Based Adder

Adders can also be realized using embedded DSP48E slices in Virtex-5 FPGAs. Each slice can accept operands of width 48 bits. To realize adders with larger operand width $n(> 48)$, we require $\lceil n/48 \rceil$ DSP48E slices. Such designs can be pipelined by activating the pipeline registers internal to the slice. An $n(> 48)$-bit pipelined DSP slice-based adder requires $\lceil n/48 \rceil - 1$ pipeline stages. For addition, the attributes "ALUMODE" and "OPMODE" have to be set to "0000" and "0001111" respectively [17]. Figure 2.7 illustrates the DSP adder architecture.

2.4.1.3 *FloPoCo*-based Adder

The behavioral synthesizable HDL generated by *FloPoCo* for Virtex-5 adders with the user-specified constraints of operating frequency remaining the same as obtained through our approach of constrained placement, shows that the circuit description

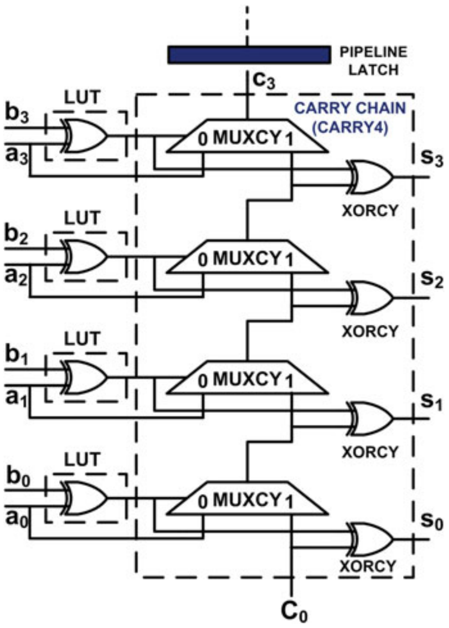


Fig. 2.6 Basic building bock for pipelined implementation of hybrid ripple carry adder (RCA)

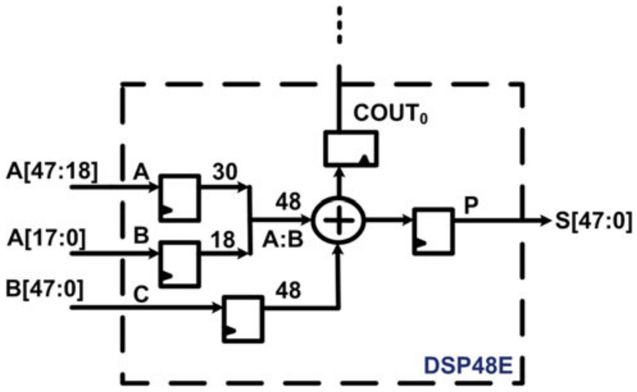


Fig. 2.7 Xilinx DSP slice-based adder [17]

generated is pipelined after every 2-bit addition as shown in Fig. 2.8. This in itself proves that the architecture cannot enjoy the benefit of utilizing a complete length-4 carry chain natively available in the Virtex-5 family. This results in a complete LUT-based implementation and the frequency constraints are not met.

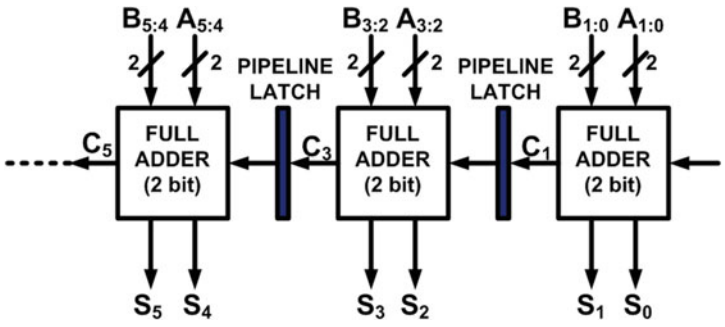


Fig. 2.8 Architecture for *FloPoCo* generated adder

2.4.1.4 Fast Carry Adder Using Carry-Lookahead Mechanism

The novel adder proposed in [16] had been designed using carry-lookahead mechanism by splitting an n-bit adder into two independent, identical portions L-RCA and H-RCA, each of which calculates $n/2$ sum bits (assuming n to be even). The H-RCA receives its carry input from a *fast carry generator* circuit. Both the L-RCA and H-RCA are architecturally identical to the pipelined implementation of the hybrid RCA shown in Fig. 2.6. The architecture of the fast adder architecture proposed in [16] for 64-bit operands is shown in Fig. 2.9. The reformulation of the carry-chain computation [18] has been addressed in the fast carry generator, where $P_{i:j}$ and

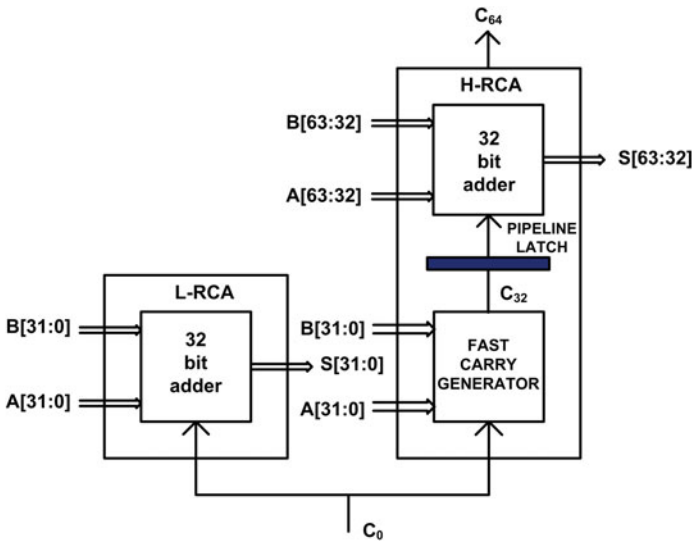


Fig. 2.9 Fast adder architecture proposed in [16]

$G_{i:j}$ denote the *group-propagated carry* and the *group-generated carry* functions, respectively, for a group of bit positions $i, i-1, \ldots, j$ (with $i \geq j$) [19].

$$P_{i:j} = \begin{cases} P_i, & \text{if } i = j \\ P_i P_{i-1:j} & \text{if } i \geq j \end{cases} \tag{2.4}$$

$$G_{i:j} = \begin{cases} G_i, & \text{if } i = j \\ G_i + P_i G_{i-1:j} & \text{if } i \geq j \end{cases} \tag{2.5}$$

where $P_i = a_i \oplus b_i$ and $G_i = a_i b_i$.

The recursive Eqs. (2.4)–(2.5) can be further generalized to $P_{i:j} = P_{i:m} P_{m-1:j}$ and $G_{i:j} = G_{i:m} + P_{i:m} G_{m-1:j}$ where $i \geq m \geq j+1$. For the mth bit position, $(i \geq m \geq j)$, we have $c_m = G_{m-1:j} + P_{m-1:j} c_j$. In Fig. 2.10, $G_{i:j}$ and $P_{i:j}$ are calculated using 6-input LUTs, where $i = j+1$ and $m = i+1$. c_m are calculated using the carry chain.

Thus,

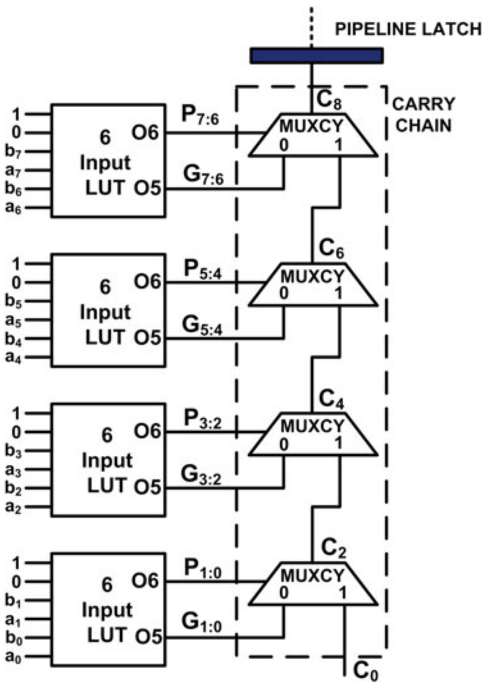


Fig. 2.10 Architecture for fast carry generator [16]

$$\begin{aligned}
P_{i:j} &= P_i P_j = (a_i \oplus b_i)(a_j \oplus b_j) \\
G_{i:j} &= G_i + P_i G_{i-1:j} = a_i b_i + (a_i \oplus b_i) a_j b_j \\
c_m &= G_{m-1:j} + P_{m-1:j} c_j = G_{m-1:m-2} + P_{m-1:m-2} c_{m-2} \\
&= G_{i:j} + P_{i:j} c_{m-2} = \overline{P_{i:j}} G_{i:j} + P_{i:j} c_{m-2}
\end{aligned}$$

Hence, c_m can be obtained from c_{m-2} using only a single multiplexer in the fast carry generator, which is in contrast to the hybrid RCA that computes c_m from c_{m-2} using two multiplexers of the carry chain. Hence, c_8 can be obtained from c_0 within a single slice, which is shown as follows where (2.6) assumes the same form as (2.3):

$$\begin{aligned}
c_8 &= \overline{P_{7:6}} G_{7:6} + P_{7:6} c_6 = \overline{P_{7:6}} G_{7:6} + P_{7:6}[\overline{P_{5:4}} G_{5:4} + P_{5:4} c_4] \\
&= \overline{P_{7:6}} G_{7:6} + P_{7:6}[\overline{P_{5:4}} G_{5:4} + P_{5:4}[\overline{P_{3:2}} G_{3:2} + P_{3:2}[\overline{P_{1:0}} G_{1:0} + P_{1:0} c_0]]]
\end{aligned} \tag{2.6}$$

The $n/2$-bit H-RCA requires $\lceil n/8 \rceil - 1$ pipeline stages, while the $n/2$-bit fast carry generator requires $\lceil n/16 \rceil - 1$ pipeline stages. Overall, an n-bit fast carry adder requires $\lceil 3n/16 \rceil - 1$ pipeline stages, including the pipeline stage between the fast carry generator and the H-RCA.

2.4.1.5 Adder Implementation Results

The adder circuit has been compared for five design styles—FPGA fabric-based adder (IP Core) generated by the GUI utility in ISE, DSP slice-based adder, the *FloPoCo* generated adder, the fast carry generator-based adder [16] and hybrid RCA. The circuits were implemented on a Xilinx Virtex-5 FPGA, device family XC5VLX330T, package FF1738 and speed grade-2 using the *Xilinx ISE* (v 12.4) design environment and all the *post place and route* implementation results are tabulated and compared with [16]. The dashed entries in Table 2.2 indicate that the equivalent results are either not applicable or not reported in [16]. The speed of operation, resource utilization, and power-delay product (PDP) of the architectures have been compared with those reported in the existing literature (if any) for different modes of implementation. Power-delay product has been calculated as the product of the power dissipation, the (minimum) clock-period (toggle rate of 12.5 %), and the latency (in terms of the number of clock cycles required to complete the computation). Although not explicitly mentioned in [16], the authors informed us through personal correspondence that they inserted register banks in the Fast Carry Adder only at the input and output ports of the circuit to estimate the frequency. The important trend to note here is that in all cases, constrained placement adders give substantially better performance than the corresponding unconstrained placed adders of the same operand width.

A partial floorplan view for a 128-bit adder is shown in Fig. 2.11 for two different implementation modes: fabric IP Core-based pipelined 128-bit adder with unconstrained placement, and a 128-bit pipelined hybrid RCA using proposed design

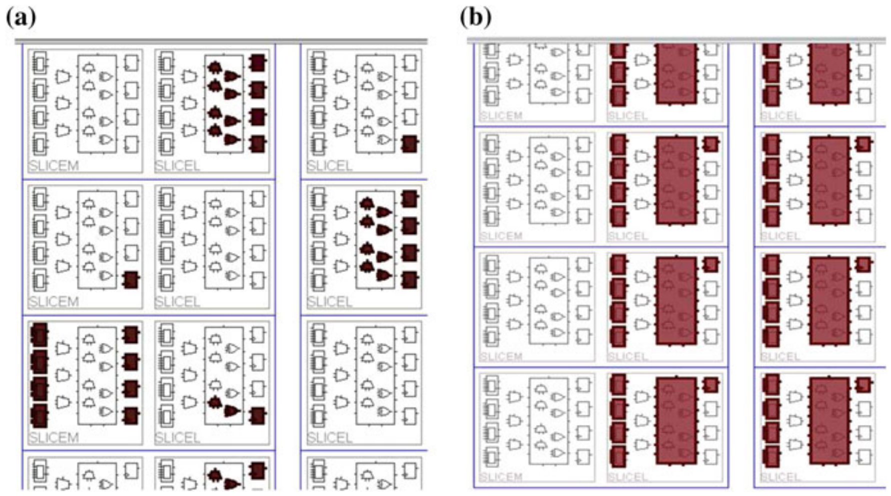


Fig. 2.11 Partial floorplan views for 128-bit adder for fabric ISE GUI-based adder and hybrid RCA with constrained placement. **a** Partial floorplan of IP core-based fabric adder with unconstrained placement. **b** Partial floorplan of hybrid RCA with constrained placement

methodology. From this figure, it is clearly visible that synthesis tools perform an unoptimized inference of logic elements and their random and haphazard placement.

2.4.2 Loadable Bidirectional Binary Counter Architecture

2.4.2.1 Proposed Counter Architecture

An up/down counter can be realized as a combination of a D-FF-based Parallel-In Parallel-Out (PIPO) register and an incrementer/decrementer, which accepts the output of the register as its input, and feedbacks its outputs to the input of the register, as shown in Fig. 2.12. Thus, counters have higher routing complexity in comparison to adders. The carry chain has been configured as wide AND gate for up-counter and wide OR gate for down counter as shown in Fig. 2.12, where larger counters can be realized by successive cascading of the "Stage 1" block. The PIPO register has been realized using the "FDRSE" Xilinx primitive which is a D-FF with synchronous reset, set, and clock enable. Pipeline latency cannot be tolerated in a counter design, as the inputs to the PIPO register come at a specific instant of time and outputs are expected to be obtained in the following clock cycle. Hence, the pipelined latches are realized using the "FDCPE" Xilinx primitive [4] which is a D-FF with clock enable and asynchronous preset and clear. These FFs are presetted if the output from the previous carry chain of the adjacent slice is high and cleared if low. For an n-bit counter, $\lceil n/4 \rceil - 1$ asynchronous pipeline stages are required.

Table 2.2 Integer adder implementation results

		Design with unconstrained placement							Pipelined design with constrained placement						
Operand width	Adder circuit	Freq (MHz)	Latency (#clk cycles)	Power delay product (nj)	#FF	#LUT	#DSP	#Slice	Freq (MHz)	Latency (#clk cycles)	Power delay product (nj)	#FF	#LUT	#DSP	#Slice
32	Fabric adder (IP core)	378.36	7	0.67	98	116	0	44	–	–	–	–	–	–	–
	DSP slice adder	550.00	2	0.08	0	0	1	0	–	–	–	–	–	–	–
	FloPoCo adder	714.29	16	1.56	138	133	0	51	–	–	–	–	–	–	–
	Fast carry adder [16]	521.00	–	–	98	41	0	–	**808.41**	**5**	**0.18**	**9**	**40**	**0**	**10**
	Hybrid RCA	–	–	–	–	–	–	–	**809.06**	**7**	**0.26**	**8**	**32**	**0**	**8**
48	Fabric adder (IP Core)	348.43	11	1.47	157	191	0	78	–	–	–	–	–	–	–
	DSP slice adder	550.00	2	0.10	0	0	1	0	–	–	–	–	–	–	–
	FloPoCo adder	664.45	24	3.66	210	205	0	112	–	–	–	–	–	–	–
	Fast carry adder [16]	472	–	–	146	61	0	–	**789.27**	**8**	**0.53**	**14**	**60**	**0**	**15**
	Hybrid RCA	–	–	–	–	–	–	–	**806.45**	**11**	**0.76**	**12**	**48**	**0**	**12**
64	Fabric adder (IP core)	468.60	15	2.48	217	263	0	117	–	–	–	–	–	–	–
	DSP slice adder	500.00	3	0.23	0	0	2	0	–	–	–	–	–	–	–
	FloPoCo adder	595.59	32	6.43	282	277	0	166	–	–	–	–	–	–	–
	Fast carry adder [16]	397.00	–	–	194	81	0	–	**788.02**	**11**	**1.08**	**19**	**80**	**0**	**20**
	Hybrid RCA	–	–	–	–	–	–	–	**806.45**	**15**	**1.32**	**16**	**64**	**0**	**16**

(continued)

Table 2.2 (continued)

Operand width	Adder circuit	Design with unconstrained placement							Pipelined design with constrained placement						
		Freq (MHz)	Latency (#clk cycles)	Power delay product (nj)	#FF	#LUT	#DSP	#Slice	Freq (MHz)	Latency (#clk cycles)	Power delay product (nj)	#FF	#LUT	#DSP	#Slice
96	Fabric adder (IP core)	413.22	23	6.05	337	407	0	182	–	–	–	–	–	–	–
	DSP slice adder	500.00	3	0.26	0	0	2	0	–	–		–	–	–	–
	FloPoCo adder	471.48	48	15.05	426	507	0	208	–	–	–	–	–	–	–
	Fast carry adder [16]	303.00	–	–	290	121	0	–	**787.40**	**17**	**2.45**	**29**	**120**	**0**	**30**
	Hybrid RCA	–	–	–	–	–	–	–	**754.15**	**23**	**3.30**	**24**	**96**	**0**	**24**
128	Fabric adder (IP core)	414.94	31	11.80	457	551	0	271	–	–	–	–	–	–	–
	DSP slice adder	500.00	4	0.47	0	0	3	0	–	–	–	–	–	–	–
	FloPoCo adder	522.19	64	25.91	570	747	0	305	–	–	–	–	–	–	–
	Fast carry adder [16]	–	–	–	–	–	–	–	**787.40**	**23**	**4.42**	**39**	**160**	**0**	**40**
	Hybrid RCA	–	–	–	–	–	–	–	**760.46**	**31**	**8.16**	**32**	**128**	**0**	**32**

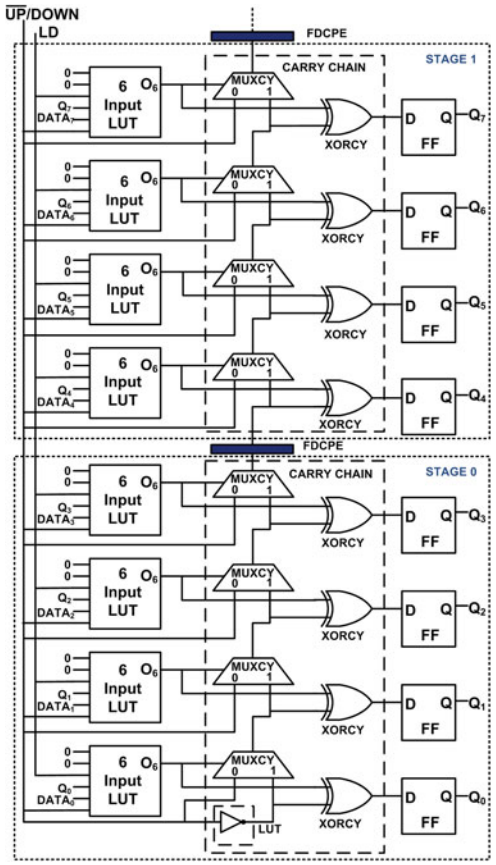


Fig. 2.12 Architecture for loadable, *up*/*down* counter targeted towards Xilinx Virtex-5 FPGA

The basic building block of this architecture is a 4-bit counter realized within a single slice. The logic functionality of accepting a new data $DATA_i$, when the "load control" signal LD to load external data to the FFs is high, and accepting the output from the FFs when LD is low, along with the XOR operation, is taken care of by the 6-input LUT configured as $O_6 = (\overline{LD} \cdot Q_i + LD \cdot DATA_i) \oplus \overline{UP}/DOWN$, where the counter counts up if $\overline{UP}/DOWN = 0$, and counts down if $\overline{UP}/DOWN = 1$. The terminal count is detected by the carry output of the most significant carry chain. As the external data to be loaded into the register is not supplied directly to the input of the FFs, but comes from the output of the incrementer/decrementer logic, the user must send the value $(x - 1)$ to load an up-counter with the value x, or send $(y + 1)$ to load a down-counter with the value y.

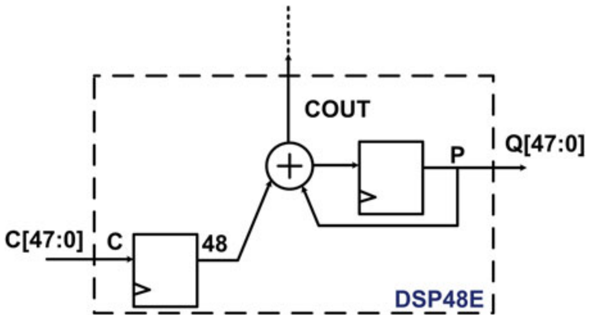


Fig. 2.13 Xilinx Virtex-5 DSP slice-based counter [17]

2.4.2.2 DSP Slice-Based Counter

Counters can also be designed using DSP48E slices, where n-bit counters can be realized by cascading $\lceil n/48 \rceil$ DSP48E slices along a column, as shown in Fig. 2.13. To achieve the desired functionality, the slices have been configured as a 48-bit accumulator each by setting the attributes "OPMODE" as 0101100, and "ALUMODE" as 0000 for addition and 0011 for subtraction [17]. The additional usage information to be taken note of is that while the counter is operating as a down-counter and the user wants to load the registers with the value x, the two's complement of x must be given as input. Currently, *FloPoCo* (v 2.5.0) does not generate HDL for counters.

2.4.2.3 Counter Implementation Results

The circuits were implemented on a Xilinx Virtex-5 FPGA, device family XC5VLX330T, package FF1738 and speed grade-2 using the *Xilinx ISE* (v 12.4) design environment and all the *post place and route* implementation results are tabulated. Operating frequencies for both the counter synthesized from behavioral description and that generated through the GUI utility deteriorate steadily with increase in the number of output bits. In contrast, the proposed design shows better operand width scalability with respect to frequency. *FloPoCo*, however, does not support counter implementations till its latest release (Table 2.3).

2.5 Compact FPGA Implementation of Cellular Automata Circuits

Cellular Automata (CA) circuits are useful for test pattern generation and construction of Built-In Self Test (BIST) structures within VLSI chips [20]. The regular, modular, and cascadable structure of CA with only local neighborhood dependence of the cells makes it suitable for VLSI implementation [20, 21]. The perceived

Table 2.3 Counter implementation results

Operand width	Counter circuit	Freq (MHz)	Power delay product (pJ)	#FF	#LUT	#DSP	#Slice
32	Behavioral counter	504.80	48.67	32	49	0	15
	Fabric counter (ISE GUI)	536.19	50.52	32	47	0	14
	DSP slice counter	550.00	35.65	0	0	1	0
	Proposed counter	**587.89**	**37.30**	**39**	**41**	**0**	**17**
48	Behavioral counter	400.32	43.39	48	70	0	29
	Fabric counter (ISE GUI)	427.35	59.09	48	69	0	30
	DSP slice counter	550.00	40.80	0	0	1	0
	Proposed counter	**567.21**	**57.10**	**59**	**61**	**0**	**25**
64	Behavioral counter	367.51	53.33	64	94	0	32
	Fabric counter (ISE GUI)	387.59	67.47	64	93	0	39
	DSP slice counter	500.00	50.46	0	0	2	0
	Proposed counter	**565.61**	**59.85**	**79**	**81**	**0**	**33**
96	Behavioral counter	292.65	89.08	96	139	0	46
	Fabric counter (ISE GUI)	298.95	84.39	96	137	0	43
	DSP slice counter	500.00	61.64	0	0	2	0
	Proposed counter	**562.75**	**72.59**	**119**	**121**	**0**	**48**

(continued)

Table 2.3 (continued)

Operand width	Counter circuit	Freq (MHz)	Power delay product (pJ)	#FF	#LUT	#DSP	#Slice
128	Behavioral counter	231.70	121.02	128	186	0	64
	Fabric counter (ISE GUI)	246.49	119.03	128	184	0	67
	DSP slice counter	500.00	79.30	0	0	3	0
	Proposed counter	**563.70**	**111.12**	**159**	**161**	**0**	**64**

regularity and locality of interconnects in a CA are often *logical* rather than *physical*, and difficult to achieve in practical implementations. Implementation of CA on FPGAs often turns out to be inefficient, because usually the user has limited control on the inference of logic elements, along with their placement and routing.

In [22], authors had reported faster implementation of CA on FPGA hardware, compared to optimized software implementation by achieving a speedup in the range of 14–19. A methodology for VLSI implementation of CA algorithms, where an automatic translation scheme from CA algorithms to the corresponding VHDL was proposed in [23]. FPGA-based CA implementation was also reported in [24]. However, to the best of our knowledge, there has been no reported work regarding the principles and design philosophy for efficient low-level implementation of CA on modern families of actual FPGAs, aiming to map the CA structures optimally to the native architecture of the FPGA.

To demonstrate our proposed design philosophy, consider a *null boundary*, maximal length linear CA [25] which is a collection of a discrete lattice of cells, where each cell has a D-FF with associated combinational logic. If the CA has n cells, then the state at any instant may be expressed as $Y_t = \{q_0(t), q_1(t), \ldots, q_{n-1}(t)\}$, where $q_i(t)$ denotes the state of the ith cell at the tth instant of time. The state of the ith cell at the $(t+1)$th instant of time is denoted by $q_i(t+1)$, where $q_i(t+1) = f(q_{i-1}(t), q_i(t), q_{i+1}(t))$, which determines the combinational logic for each stage. Here, '$f()$' is known as the *rule of the CA* [25], which, if expressed in the form of a truth-table, the equivalent decimal output is called *rule number of the CA*. For example, the next state logic equations for *rule-90* and *rule-150* CAs are given as $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$ and $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$ respectively [26] with their circuit representations as depicted in Fig. 2.14.

If the CA is *linear*, the combinational logic functions $f()$ involves only XOR logic. A CA having a combination of XOR and XNOR logic is called an *additive* CA, whereas for *non-linear* or *non-additive* CA, $f()$ involves AND/OR logic [27]. If all CA cells obey the same rule, then it is termed as *uniform* CA, else it is a *hybrid*

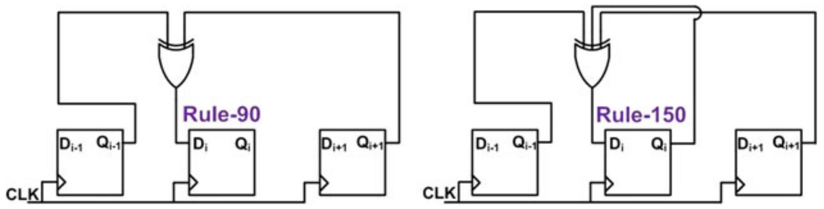


Fig. 2.14 Combinational logic for cells corresponding to *rule-90* and *rule-150*

CA. By convention, CA is usually described by a string of 0's and 1's, where, for example, '1' refers to rule-150 and '0' refers to rule-90. Our proposed methodology can efficiently implement two-rule *linear*, *additive*, *uniform* and *hybrid* CAs.

2.5.1 Adapting CA to the Native FPGA Architecture

Packing is a key step in the FPGA tool flow that is tightly integrated with the boundaries between *logic synthesis*, *technology mapping* and *placement* [3]. For Virtex-5 FPGAs, the packing technique targets the dual-output LUTs to achieve area efficiency by exploring the feasibility of packing two logic functions into a single LUT [3]. This is possible whenever the two logic functions have no more than five distinct input variables. In such cases, a more efficient mapping of the design is expected, culminating into shorter interconnect wirelength, which in turn results in lesser critical path delay. However, our implementation results for Virtex-6 family of FPGAs, which is an advanced and modified version of Virtex-5, clearly show that in spite of the methodology adopted by the common FPGA CAD tools, the packing behavior is highly unpredictable and the tool fails to configure the LUTs in the dual output mode. This doubles the overall LUT and slice requirement.

Consider a 1-D CA where the next state of a particular cell depends only on itself or on one or both of its two immediate neighbors. It is easy to deduce that in such cases, any two adjacent cells can have a maximum of four distinct inputs. In such a situation, it is possible to pack the next state logic for any two adjacent cells of a CA into a single LUT. Since Virtex-6 architectures facilitate registering of both the LUT outputs using two FFs present in the same slice as that of the LUT, we can achieve a compact FPGA realization of the architecture [28]. The architecture for a 16 cell 1-D linear maximal length CA for the (primitive) polynomial $x^{16}+x^5+x^3+x^2+1$ (or the equivalent *hybrid* rule $< 0001111001001000 >$) [29] is shown in Fig. 2.15. Thus, for an n-cell maximal length CA architecture, $\lceil n/8 \rceil$ Virtex-6 FPGA slices are required.

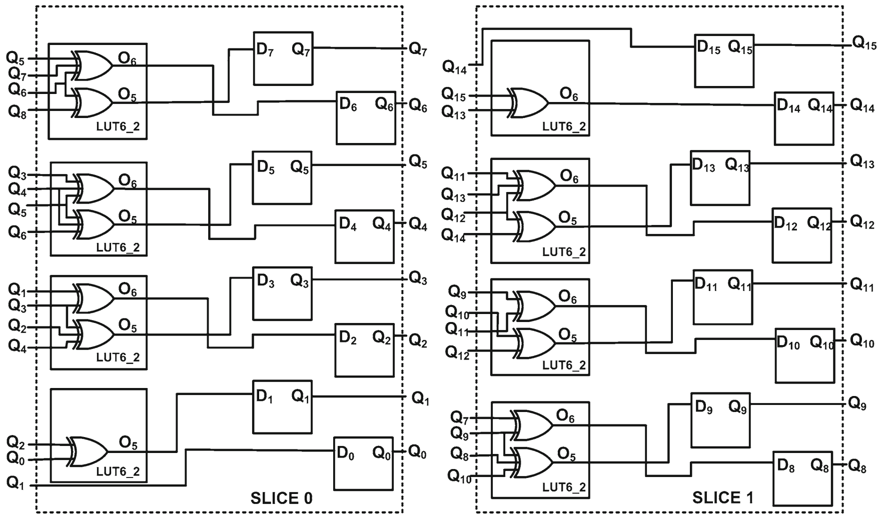


Fig. 2.15 Optimized architecture for a high-performance 16-bit 1-D linear CA for the (primitive) polynomial $x^{16} + x^5 + x^3 + x^2 + 1$ (or the equivalent hybrid rule $< 0001111001001000 >$) mapped on Xilinx Virtex-6 FPGAs [28]

2.5.2 CA Implementation Results

The CA circuits were implemented on Xilinx Virtex-6 FPGA, device family XC6VLX550T, package FF1760 and speed grade -2 using the *Xilinx ISE* (v 12.4) design environment. Polynomials of CAs of the order 32, 48, 64, 80, and 96 were implemented using two different techniques—RTL coding followed by unconstrained automatic logic synthesis by ISE and the custom design technique using the proposed design methodology. It was observed that for an RTL description of the CA circuit, the Xilinx *post place and route* results indicate that double the FPGA area gets consumed than what a compact realization should have taken. The operating speed for the CA circuits also drastically reduces as the order of the polynomial is steadily increased, which is undesirable from the point of view of hardware acceleration. The implementation results were compared with respect to their frequency of operation, PDP, and hardware resource requirement (FFs, LUTs and slices), and are tabulated in Table 2.4. The polynomials are from [30] and, for example, the entry in the polynomial field of Table 2.4, 32 28 27 1 0, represents the polynomial $x^{32} + x^{28} + x^{27} + x + 1$.

Table 2.4 CA implementation results

Polynomial	Mode of implementation	Freq (MHz)	Power delay product (pJ)	#FF	#LUT	#Slice
32, 28, 27, 1, 0	RTL design	1014.20	31.61	32	30	8
	Proposed design	**1103.75**	**31.15**	**32**	**16**	**4**
48, 28, 27, 1, 0	RTL design	320.41	37.36	48	46	12
	Proposed design	**1089.32**	**40.26**	**48**	**24**	**6**
64, 4, 3, 1, 0	RTL design	361.40	43.80	64	64	16
	Proposed design	**1083.42**	**52.92**	**64**	**32**	**8**
80, 38, 37, 1, 0	RTL design	414.08	64.05	80	78	20
	Proposed design	**976.56**	**59.08**	**80**	**40**	**10**
96, 49, 47, 2, 0	RTL design	361.79	70.92	96	96	24
	Proposed design	**908.27**	**62.59**	**96**	**48**	**12**

2.6 Design Automation

The design of all the circuits has been automated using a CAD tool developed by us. We call the CAD tool *FlexiCore*, short for "**Flexi**ble Arithmetic Soft **Core** Generator." It is flexible in a sense that the operand widths for the mapped circuits can be varied, and the CAD tool allows partial control to the user over the placement of the circuits on the FPGA fabric. The tool is developed in JAVA, and includes a simple GUI. The CAD software executable is invoked from the TCL command—prompt in-built in Xilinx ISE using a top-level TCL script. Currently, *FlexiCore* can generate synthesizable HDL and placement constraints for adders/subtractors, absolute difference circuits, multipliers, squarers, universal shift registers, comparators, counters, and CA-based pseudorandom binary sequence generators.

The *FlexiCore* design flow is depicted in Fig. 2.16. Here, the top-level script invokes a GUI which displays the list of circuits currently supported by *FlexiCore*, and prompts the user to enter (in the GUI entry fields) the circuits (along with their operand widths and whether the user wants pipelined/non-pipelined version), for which the user wants constrained placement-based high-performance design. The user can also optionally enter the starting coordinate for the entire constrained placement exercise. If this is not provided, *FlexiCore* determines the feasible starting coordinate from the existing project constraints file called User Constraints File (.ucf).

After the user enters her options, *FlexiCore* examines the feasibility of placement of the selected building blocks on the FPGA fabric, with the starting coordinate

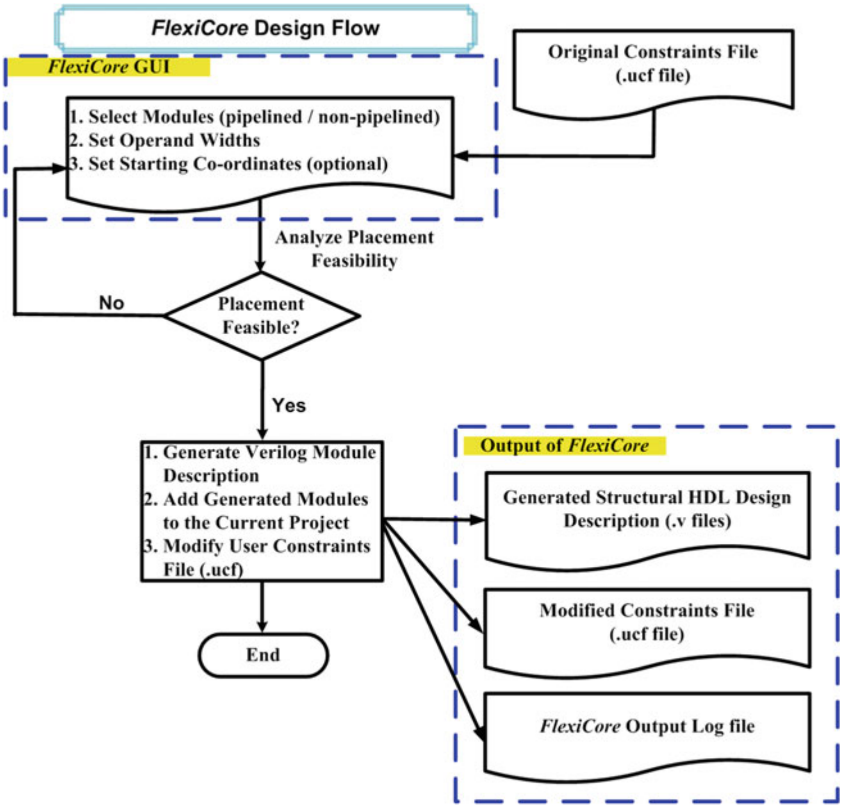

FlexiCore Design Flow
FlexiCore GUI
1. Select Modules (pipelined / non-pipelined)
2. Set Operand Widths
3. Set Starting Co-ordinates (optional)
Original Constraints File (.ucf file)
Analyze Placement Feasibility
Placement Feasible?
No
Yes
1. Generate Verilog Module Description
2. Add Generated Modules to the Current Project
3. Modify User Constraints File (.ucf)
End
Output of FlexiCore
Generated Structural HDL Design Description (.v files)
Modified Constraints File (.ucf file)
FlexiCore Output Log file

Fig. 2.16 The *FlexiCore* design flow for arithmetic circuits

entered by the user as origin, or the starting coordinate inferred. It takes into consideration the existing placement constraints, if any, in the project constraints file. If the placement is deemed feasible, *FlexiCore* performs the following:

- Generates the Verilog module descriptions for the selected circuit building blocks, and adds the files to the current project. Care is taken to ensure that no hardware primitive on the FPGA is used more than once in building the high-performance building blocks. Pipeline registers as required, are automatically inserted. At present, *FlexiCore* supports two options—either a maximally pipelined implementation (optimized for Virtex-5 and Virtex-6 platform), or a purely combinational circuit. We expect to support variable latency circuits in future.
- Modifies the project User Constraints File (.ucf), by adding the placement constraints for the generated high-performance circuit building blocks.
- Creates a log file to provide the user with all the necessary information about the generated modules.

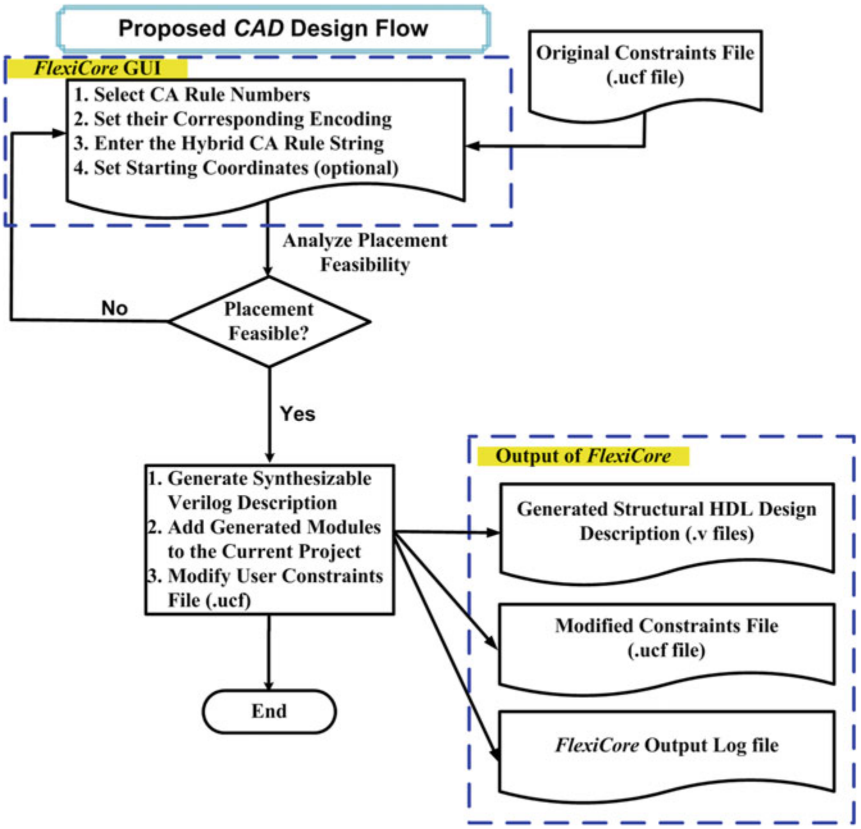

Proposed CAD Design Flow
FlexiCore GUI
1. Select CA Rule Numbers
2. Set their Corresponding Encoding
3. Enter the Hybrid CA Rule String
4. Set Starting Coordinates (optional)
Original Constraints File (.ucf file)
Analyze Placement Feasibility
Placement Feasible?
No
Yes
1. Generate Synthesizable Verilog Description
2. Add Generated Modules to the Current Project
3. Modify User Constraints File (.ucf)
End
Output of FlexiCore
Generated Structural HDL Design Description (.v files)
Modified Constraints File (.ucf file)
FlexiCore Output Log file

Fig. 2.17 The *FlexiCore* design flow for CA circuits [28]

If *FlexiCore* fails to find a feasible placement configuration, it reports it to the user and again prompts her to enter a (reduced) number of building blocks, or a different starting coordinate. Note that the situation where *FlexiCore* fails to find a feasible placement rarely arises, given the large availability of resources on a Virtex family FPGA. We did not find any such scenario with our real-life design testcases.

To accommodate the CA circuits into the CAD tool for their automatic generation, provision has been kept for the user to invoke the GUI, which displays all the list of rules corresponding to which equivalent CA circuits can be generated, and prompts the user to enter the following fields: the two CA rule numbers, their corresponding encoding of 0 and 1, and the hybrid CA rule comprising of a string of 0's and 1's. The CAD tool interprets the string by reading two bits at a time, calculates the *truth table* of the dual output LUTs appropriately for realizing the next state logic for the CA cells, and instantiates the required FPGA logic elements in the HDL code. The remaining CAD tool flow remains to be the same as for arithmetic circuits. The design flow, particularly for CA circuits, is shown in Fig. 2.17.

2.7 Case Study—a Greatest Common Divisor (GCD) Calculator

We shall present a case study of the hardware implementation of a Greatest Common Divisor (GCD) calculator. The architecture uses several of the arithmetic building blocks supported by *FlexiCore* such as the absolute difference circuit, counter, and a barrel shifter. The architecture has been derived from the Binary GCD algorithm [31] which has been explained in Algorithm 1. This algorithm uses simpler arithmetic operations than the conventional Euclidean GCD algorithm as it replaces complex operations such as division and multiplication with division and multiplications by powers of two (implemented using only shift operations), comparisons and subtractions [32], thereby making it suitable for hardware implementation .

The architecture for the algorithm at the block diagram level is shown in Fig. 2.18. We present two *multi-function* registers P and Q which are loaded in accordance with the control signals: active low load control signal $\overline{INIT}$, which accepts two unsigned integers as inputs whose GCD has to be computed, and LSBs of registers P and Q as depicted in Table 2.5. The multifunction registers and its associated combinational logic, which is a nonstandard representation of a 4:1 multiplexer, has been mapped intelligently to the 6-input LUTs and wide function multiplexers $MUXF7$ as shown

Algorithm 1: GCD Calculation Algorithm

```
   Input: 2 unsigned integers: $P$ and $Q$.
   Output: S : GCD of $P$ and $Q$
1  $Rem(P, Q)$: Remainder when $P$ is divided by $Q$
2  $abs(P - Q)$: Absolute difference of $P$ and $Q$
3  $min(P, Q)$: Minimum of $P$ and $Q$
4  $Computation\_Over\_Flag \leftarrow 0$, $R \leftarrow 0$
5  begin
6      while $P \neq Q$ do
7          if (Rem(P,2)==0) then
8              $P \leftarrow P/2$;
9              if (Rem(Q,2)==0) then
10                 $Q \leftarrow Q/2$;
11                 $R \leftarrow R + 1$;
12         else
13             if (Rem(Q,2)==0) then
14                 $Q \leftarrow Q/2$;
15             else
16                 $P \leftarrow abs(P - Q)$;
17                 $Q \leftarrow min(P, Q)$;
18 end
19 $S \leftarrow P * (2^R)$;
20 $Computation\_Over\_Flag \leftarrow 1$;
```

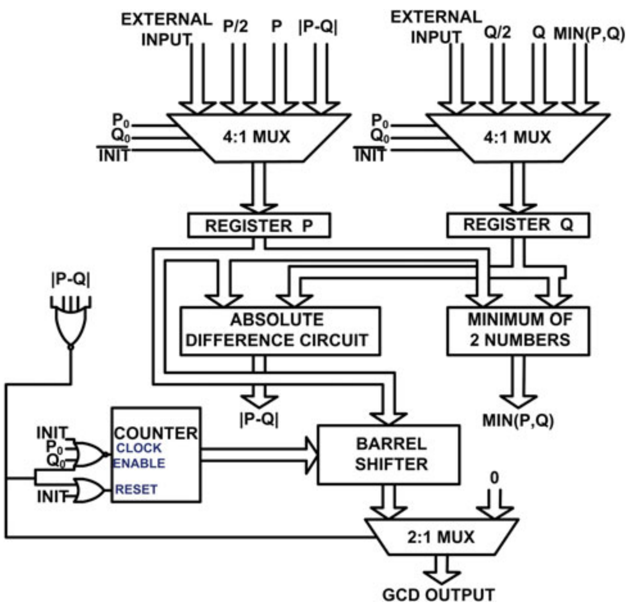


Fig. 2.18 Overall architecture of the GCD computation circuit

Table 2.5 Function table for the multi-function registers and counter

Control/Select signals			Registers		Counter
$\overline{INIT}$	P_0	Q_0	P	Q	R
0	X	X	LOAD	LOAD	0
1	**0**	0	$P/2$	$Q/2$	$R+1$
1	**0**	1	$P/2$	Q	R
1	**1**	0	P	$Q/2$	R
1	**1**	1	$\|P-Q\|$	$min(P, Q)$	R

in Fig. 2.19 to ensure compact implementation. The absolute difference circuit has been realized as was proposed in [33] which comprises of a less-than comparator (Fig. 2.20) and a subtractor (Fig. 2.21) as its sub-circuits. If A and B are two inputs, the n-bit less-than comparator generates a high signal if $A < B$. Each LUT accepts 2-bit sub-words $A_{i:i-1}$ and $B_{i:i-1}$, each of which has no more than four distinct inputs, and outputs two signals $AeqB_{i:i-1}$ and $AlessB_{i:i-1}$. $AeqB_{i:i-1} = 1$ if $A_{i:i-1} = B_{i:i-1}$ and $AlessB_{i:i-1} = 1$ if $A_{i:i-1} < B_{i:i-1}$. $AeqB_{i:i-1}$ drives the select line of the

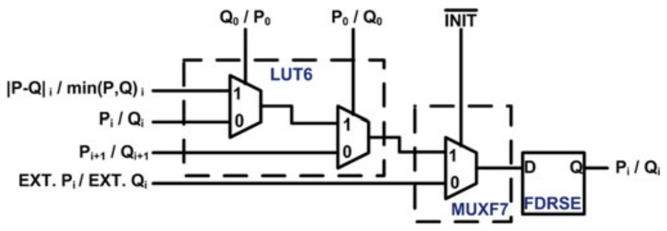


Fig. 2.19 Multifunction register

multiplexer of the carry chain and $AlessB_{i:i-1}$ is an input to the multiplexer which is selected if $AeqB_{i:i-1}$=0. If $x_i = A_i \odot B_i$ and $x_{i-1} = A_{i-1} \odot B_{i-1}$, then

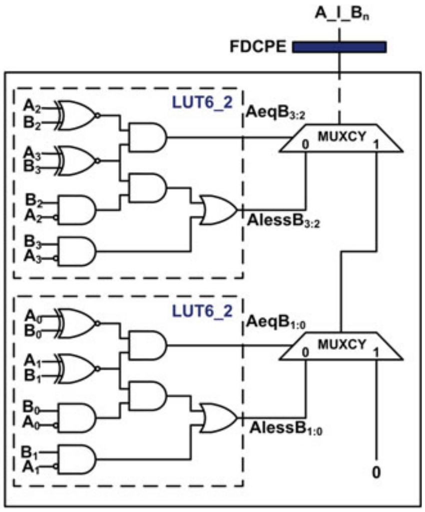
$$AlessB_{i:i-1} = \overline{A_i} B_i + x_i \overline{A_{i-1}} B_{i-1}$$
$$AeqB_{i:i-1} = x_i x_{i-1}$$

The output of the less-than comparator $A\_l\_B_n$ decides upon the operation $A - B$ or $B - A$. For an n-bit less than comparator, its output $A\_l\_B_n$ is obtained using the following Boolean logic recurrence relation:

$$A\_l\_B_n = \overline{AeqB_{n:n-1}} AlessB_{n:n-1} + AeqB_{n:n-1} A\_l\_B_{n-2}$$

where the *base* condition is $A\_l\_B_0 = 0$. This recurrence relation bears exact resemblance to (2.3) making it an ideal candidate for carry-chain implementation. When $A\_l\_B_n = 1$, B $+ \overline{A} + 1$ is computed, else A $+ \overline{B} + 1$ is computed, as shown in Fig. 2.21, where $\overline{A}$ and $\overline{B}$ are the 1's complement of A and B respectively.

Fig. 2.20 Module computing if $A < B$ [33]



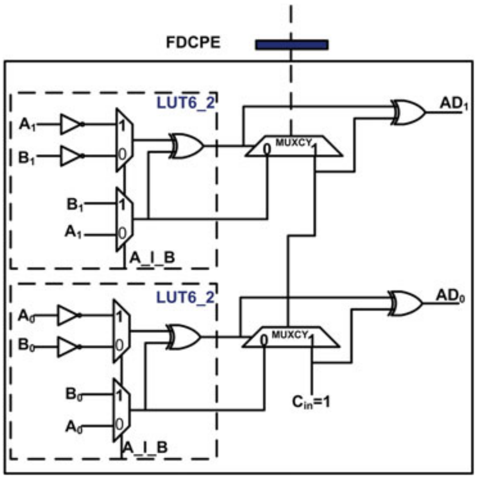


Fig. 2.21 Module computing absolute difference [33]

The absolute difference circuit has been pipelined using the "FDCPE" Xilinx primitive [4] where these FFs are presetted if the output from the previous carry chain of the adjacent slice is high and cleared if low. An intermediate output A_l_B (which decided whether to compute $A - B$ or $B - A$) of the absolute difference circuit serves as a select line to the multiplexer which outputs the minimum of two numbers. This architecture to compute the minimum of two numbers has been realized using dual output LUTs as shown in Fig. 2.22. The counter keeps track of the number of left shifts to be applied to the contents of the P register after the final iteration. The contents of the register P are left-shifted using a barrel shifter which is implemented using dual output LUTs as shown in Fig. 2.23 where stage i of the barrel shifter ($i \geq 0$) can implement a $2^i/0$ bit shift. Thus, the data to be shifted is given to the data inputs of the multiplexers, whereas the amount of left shift is given as input to the select lines of the multiplexers in the barrel shifter. The final output gives the GCD of two numbers.

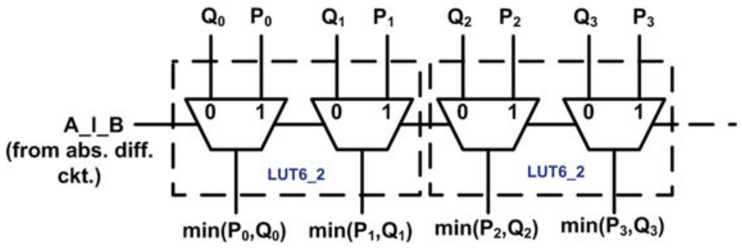

Q_0 P_0 Q_1 P_1 Q_2 P_2 Q_3 P_3
A_I_B (from abs. diff. ckt.)
LUT6_2
LUT6_2
min(P_0,Q_0) min(P_1,Q_1) min(P_2,Q_2) min(P_3,Q_3)

Fig. 2.22 Circuit to compute minimum of two numbers

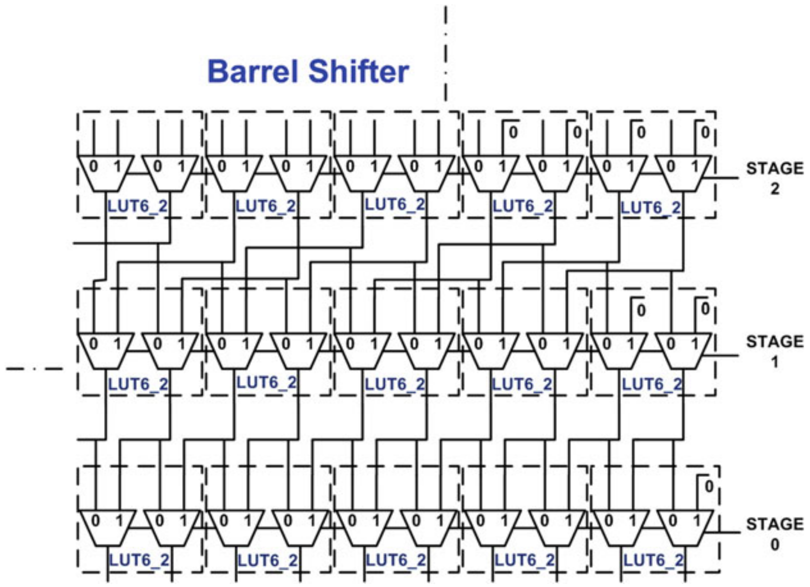


Fig. 2.23 LUT level implementation of the barrel shifter

2.7.1 GCD Implementation Results

The GCD computation circuit for 32-bit operands was implemented on the Xilinx Virtex-5 FPGA using two approaches: behavioral Verilog modelling, and second, using constrained arithmetic circuit descriptions generated by *FlexiCore*. Results are tabulated in Table 2.6, where the two input operands are 70 and 100. The results clearly indicate that using the second approach, the designer can achieve a higher frequency and lower PDP value with considerable lesser amount of hardware resources.

Table 2.6 Implementation results for a 32-bit GCD Circuit (Operands: 100 and 70)

Implementation mode	Freq (MHz)	Power delay product (pJ)	#FF	#LUT	#Slice
Behavioral modelling	160.49	745.85	69	356	208
Primitive instantiation	**214.73**	**508.87**	**87**	**298**	**93**

2.8 Conclusion

Manual instantiation of hardware primitives and macros, and their careful, constrained placement on the FPGA fabric leads to very promising performances in terms of speed. We have considered some arithmetic circuits and pseudorandom sequence generators which are very regular in their architectures and have shown how to configure them using the *bit-sliced* design paradigm where an entire architecture has been constructed using identical sub-circuits. Designs that are pipelined and have a very regular data flow, like those considered in our work, usually lend themselves to regular floorplanning. Since each slice of an FPGA are register-rich, pipelined implementations can be done with ease without consuming additional number of slices. The regular architectures also facilitate their design automation which is taken care of by the *FlexiCore* CAD tool. The tool is not only capable of generating the synthesizable HDL and placement directives for the designs, but can also examine the feasibility of placement of a circuit by ensuring that the area spanned by it on the FPGA fabric is not occupied by any other logic.

Acknowledgments The authors would like to thank Prof. Anindya Sundar Dhar, Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, and Dr. Debdeep Mukhopadhyay, Department of Computer Science and Engineering, IIT Kharagpur, for their valuable insights into the work. The authors would also like to acknowledge two undergraduate students of the Department of Computer Science and Engineering, IIT Kharagpur, Mohammad Salman and Sreemukh Kardas, for their contributions in developing the proposed CAD tool, *FlexiCore*.

References

1. Preuβer, T.B., Zabel, M., Spallek, R.G.: Accelerating computations on FPGA carry chains by operand compaction. In: 20th IEEE Symposium on Computer Arithmetic (ARITH), pp. 95–102 (2011)
2. Preuβer, T.B., Spallek, R.G.: Mapping basic prefix computations to fast carry-chain structures. In: International Conference on Field Programmable Logic and Applications (FPL), pp. 604–608 (2009)
3. Ahmed, T., Kundarewich, P.D., Anderson, J.H.: Packing techniques for Virtex-5 FPGAs. ACM Trans. Reconfig. Technol. Syst. (TRETS), **2**(18), 18:1–18:24 (2009)
4. Xilinx Inc., Virtex-5 Libraries Guide for HDL Designs, UG621 (v 11.3) (2009). Cited 16 September 2009, http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/Virtex5_hdl.pdf
5. Ehliar, A.: Optimizing Xilinx designs through primitive instantiation. In: Proceedings of the 7th FPGAworld Conference, pp. 20–27 (2010)
6. FloPoCo: Arithmetic core generator (2014). Cited 14 June 2014, http://flopoco.gforge.inria.fr/
7. Dinechin, F.de, Pasca, B.: Designing custom arithmetic data paths with FloPoCo. IEEE Des. Test Comput. **28**(3), 18–27 (2009)
8. Cosoroaba, A., Rivoallon, F.: Xilinx Inc., White paper: Virtex-5 family of FPGAs. Achieving Higher System Performance with the Virtex-5 Family of FPGAs WP245 (v1.1.1) (2006). Cited 7 July 2006, http://www.origin.xilinx.com/support/documentation/white_papers/wp245.pdf
9. Xilinx Inc., Virtex-5 FPGA user guide, UG190 (v 5.4) (2012). Cited 16 March 2012, http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

10. Xilinx Inc., Virtex-6 FPGA configurable logic block, UG364 (v 1.2) (2012). Cited 24 June 2009, http://www.xilinx.com/support/documentation/user_guides/ug364.pdf
11. Verma, A.K., Brisk, P., Ienne, J.P.: Challenges in automatic optimization of arithmetic circuits. In: 19th IEEE Symposium on Computer Arithmetic (ARITH), pp. 213–218 (2009)
12. Roy, S.S., Rebeiro, C., Mukhopadhyay, D.: Theoretical modeling of the Itoh-Tsujii inversion algorithm for enhanced performance on k-LUT based FPGAs. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1–6 (2011)
13. Hachtel, G.D., Somenzi, F.: Logic Synthesis and Verification Algorithms. Kluwer Academic Publisher, Dordrecht (1996)
14. Weste, N.H.E., Harris, D., Banerjee, A.: CMOS VLSI Design: A Circuits and Systems Perspective. 3rd edn. Pearson Publisher, New York (2011)
15. Cortadella, J., Llabería, J.: Evaluation of $A + B = K$ conditions without carry propagation. IEEE Trans. Comput. **41**(11), 1484–1487 (1992)
16. Zicari, P., Perri, S.: A fast carry-chain adder for Virtex-5 FPGAs. In: 15th IEEE Mediterranean Electrotechnical Conference (MELECON), pp. 304–308 (2010)
17. Xilinx Inc., Virtex-5 FPGA XtremeDSP design considerations user guide, UG193 (v 3.5) (2012). Cited 26 January 2012, http://www.xilinx.com/support/documentation/user_guides/ug193.pdf
18. Koren, I.: Computer Arithmetic Algorithms, 2nd edn. A.K.Peters Ltd, Natick (2002)
19. Brent, R.P., Kung, H.T.: A Regular layout for parallel adders. IEEE Trans. Comput. **C-31**(3), 260–264 (1982)
20. Sarkar, P.: A brief history of cellular automata. ACM Comput. Surv. (CSUR) **32**(1), 80–107 (2000)
21. Chowdhury, D.R., Chaudhuri, P.P.: Architecture for VLSI design of CA based byte error correcting code decoders. In: Proceedings of the 7th International Conference on VLSI Design, pp. 283–286 (1994)
22. Halbach, M., Hoffmann, R.: Improving cellular automata in FPGA logic. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, pp. 258–262 (2004)
23. Sirakoulis, G.C., Karafyllidis, I., Thanailakis, A., Mardiris, V.: A methodology for VLSI implementation of cellular automata algorithms using VHDL. Adv. Eng. Softw. **32**(3), 189–202 (2000)
24. Torres-Huitzil, C., Delgadillo-Escobar, M., Nuno-Maganda, M.: Comparison between 2D cellular automata based pseudorandom number generators. IEICE Electron. Express **9**(17), 1391–1396 (2012)
25. Das, A.K., Ganguly, A., Dasgupta, A., Bhawmik, S., Chaudhuri, P.P.: Efficient characterization of cellular automata. IEEE Proc. Comput. Digital Tech. **137**(1), 81–87 (1990)
26. Chaudhuri, P.P., Chowdhury, D.R., Nandi, S., Chattopadhyay, S.: Additive Cellular Automata Theory and its Application. vol. 1. IEEE Computer Society Press (1997)
27. Mukhopadhyay, D.: Group properties of non-linear cellular automata. J. Cell. Autom. **5**(1–2), 139–155 (2010)
28. Palchaudhuri, A., Chakraborty, R.S., Salman. M., Kardas, S., Mukhopadhyay, D.: Highly compact automated implementation of linear CA on FPGAs. In: Cellular Automata—11th International Conference on Cellular Automata for Research and Industry, pp. 388–397 (2014)
29. Cattell, K., Muzio, J.: Technical Report: Tables of linear cellular automata for minimal weight primitive polynomials of degrees up to 300. Issue: 163. University of Victoria (BC), Department of Computer Science (1991)
30. Bardell, P.H., McAnney, W.H., Savir, J.: Built-In Test for VLSI: Pseudorandom Techniques, Wiley, London (1987)
31. Stehlé, D., Zimmermann, P.: A binary recursive GCD algorithm. In: Proceedings of ANTS'04, Lecture Notes in Computer Science, vol. 3076, pp. 411–425. Springer, New York (2004)
32. Brent, R.P., Kung, H.T.: A systolic algorithm for integer GCD computation. In: IEEE 7th Symposium on Computer Arithmetic (ARITH), pp. 118–125 (1985)
33. Perri, S., Zicari, P., Corsonello, P.: Efficient absolute difference circuits in Virtex-5 FPGAs. In: 15th IEEE Mediterranean Electrotechnical Conference (MELECON), pp. 309–313 (2010)

Chapter 3
Design Intelligence for Interconnection Realization in Power-Managed SoCs

Houman Zarrabi, A.J. Al-Khalili and Yvon Savaria

Abstract In this chapter various intelligent techniques for modeling, design, automation, and management of on-chip interconnections in power-managed SoCs are described, including techniques that take into account various technological parameters such as crosstalk. Such intelligent techniques guarantee that the integrated interconnections, used in power-managed SoCs, are well-designed, energy-optimal, and meet the performance objectives in all the SoCs operating states.

3.1 Introduction

In deep submicron (DSM) realm, the design and the synthesis of VLSI Microsystems target numerous complex embedded applications. For this purpose, the number of the processing engines in a single chip is multiplied; which has led to the introduction of multiprocessor systems on-chip (MPSoC). These engines heavily communicate to synchronize their local data within the entire system network; this has resulted in the introduction of communication-centric design domain. Arising performance challenges need to be addressed together with the challenges due to system (energy) resource constraints. These contradictory requirements have pushed the VLSI system designers to investigate *intelligent* system-level solutions for the synthesis of lowenergy MPSoC [1].

H. Zarrabi (✉) · A.J. Al-Khalili
Department of ECE, Concordia University, Montreal, Canada
e-mail: zarrabi@ece.concordia.ca

A.J. Al-Khalili
e-mail: asim@ece.concordia.ca

Y. Savaria
Department of EEE, Ecole Polytechnique de Montreal, Montreal, Canada
e-mail: yvon.Savaria@polymtl.ca

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_3

Several *intelligent* solutions have been described in the literature that successfully can reduce the energy consumption of the DSM VLSI Microsystems. One such *intelligent* solution that addresses the simultaneous needs of both low energy and high performance in VLSI Microsystems is the use of the *power management design scheme* [2–5]. This *intelligent* class of technique has been extremely successful that has become a design trend for DSM MPSoC design. A power-managed MPSoC is typically designed and synthesized to enable its integrated components to run at multiple operating states, wherein each state may be associated with a distinct voltage and/or an operating frequency. A power management unit (PMU) *intelligently* adjusts the system (components) operating states based on the momentum of system workload.

In order to further improve the energyefficiency of DSM power-managed MPSoCs, the communication and/or synchronization mechanisms need to be controlled independently from the computing engines [6–9]. Yet, in order to fulfill these design objectives, the system elements involved with those interconnect-centric mechanisms, i.e., interconnections, need to be *intelligently designed and controlled (managed),* to meet the performance objectives, in all the system operating states. This issue becomes much more important specifically in DSM where, centimeter-long interconnections and communication mechanisms (like integrated buses) are seen within the VLSI Microsystems [10–12].

This chapter *describes a complete flow* which includes *intelligent* techniques for modeling, design, and management of on-chip interconnects, in power-managed VLSI. These *intelligent* techniques guarantee that the designed and managed interconnections have minimum energy requirements and that they meet all the performance objectives (operating frequencies and/or latencies) *in all the system operating states.* These techniques consider the impact of crosstalk in interconnects. This *intelligent* flow is applicable to the design of VLSI systems enabled with run-time power management, multiple voltage domains (voltage islands) [13], and multiple clock domains (MCD) [14, 15] systems, to name a few.

With respect to the context of interconnection design, uniform repeater insertion has been referred to, as a common interconnect optimization technique that can improve the signal latency, signal integrity, and signal reliability. By splitting a wire line into equally divided segments and inserting uniformly sized repeaters, the signal latency in the wire is reduced. In this case, also a better signal noise margin may be achieved. Many techniques have been stated on the uniform interconnection design and repeater insertion in the literature [16–26]. Among which, the optimal number and size of the repeaters to achieve the minimum signal latency have been given in [16, 23]. Some contributions that investigate the design of interconnects with latency, operating frequency, and area constraints, are found in [16, 17]. Some researchers have contributed to power-optimal interconnection synthesis considering latency and bandwidth constraints [18]. All previously reported methods, however, consider systems running at their nominal operating state. Some researchers have reported error-correcting techniques to improve the efficiency of power-managed communication mechanisms [19]. This chapter aims at *intelligently* helping the design of interconnect-centric components, synchronization and/or communication mechanisms, etc., to meet their performance objectives, in all the system operating states, while minimizing the energy requirements.

With respect to the context of dynamic voltage scaling (DVS) design schemes for power-managed VLSI, DVS is mainly designed and synthesized using performance models derived for CMOS logic [2]. This underlying assumption has been made repeatedly including in some modern works as well [15, 27–29]. The authors have recently shown that modeling DVS systems based on CMOS logic in DSM, where interconnect delays dominate those of logic, can be inaccurate [30]. Consequently, the associated DVS schemes should be *intelligently* modified to maintain accuracy. In this chapter, for the purpose of interconnection management, an *intelligent* DVS scheme is stated that considers the effect of interconnect parasitic into account. For this purpose, some design metrics that distinguish the performance of DVS system components according to their interconnection delay portion are stated. Based on the defined design metrics, a compact delay model and a method to *intelligently* select the supply voltage for DVS are given. A scaling limit for hazard-free system operation is also described. This limit differs from the one determined by the process technology. The *described intelligent* DVS scheme, improves the performance and energy savings of interconnections in power-managed VLSI. This chapter further extends its associated preliminary concepts, as appeared in [31, 32].

The organization of the chapter is as follows: in the second section, performance models as well as their impact and implications are described for *intelligent* modeling of on-chip interconnections in power-managed VLSI systems. Based on these models, in the third section *intelligent* design space exploration methods are conducted for crosstalk-aware energy-optimal on-chip interconnections, with latency and/or operating frequency objectives, in power-managed VLSI. In the fourth section, the *intelligent* energy-aware management scheme to control interconnections in power-managed VLSI is described. The complete flow encompassing the *intelligent* methods is illustrated in the fifth section. In the sixth section, two integrated buses in a power-managed environment, are designed and controlled as a case study. HSPICE simulations are provided to confirm the validity of the described *intelligent* techniques. Conclusions are given in the seventh section.

3.2 Intelligent Models of Interconnections in Power-Managed SoCs

3.2.1 The Performance Models

For accurate performance modeling and analysis of on-chip interconnections in DSM power-managed VLSI, we consider their generic building block model. This model is based on the typical generic structure given in Fig. 3.1. Based on this figure, the nominal propagation delay time ($T_{50\,\%}$) of a single interconnect stage in this

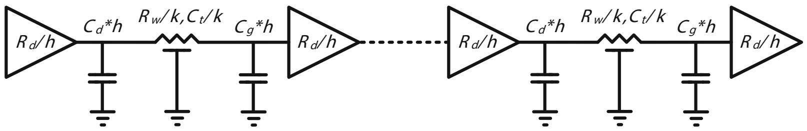


Fig. 3.1 A generic decoupled interconnect model in microsystems

structure can be expressed by (3.1) [33, 34]. Table 3.1 summarizes the definitions and the notations, used throughout this chapter.

$$T(l, w, h, k) = 0.69R_d(C_b + C_t/hk) + 0.38R_w(C_t/k^2 + 1.81C_g h/k) \quad (3.1)$$

Equation (3.1), shows that the interconnect delay is a function of the interconnect length (l) and its width (w), as well as, the size (h), and the number (k) of repeaters. In (3.1), the coefficient 0.38 converts *RC* time constants of a *distributed RC* delay model into a $T_{50\,\%}$ contribution to the wire delay, and the coefficient 0.69 represents the delay contribution for the rest of the *Lumped* components in the *RC* loops [33]. Equation (3.1) is based on the well-known Elmore delay model to compute the device and interconnect delays [35]. Although the Elmore delay model may provide conservative delay estimation in DSM designs, for routing wire trees with many branches, it is still a good delay measurement for two-pin nets which includes the majority of all nets in real designs [36]; and thus the focus of this work is for high-level modeling, design, and management purposes.

The effective interconnect capacitance (c_t) seen in the model is based on the consideration of the intrinsic interconnect capacitances (c_w) and neighboring coupling capacitances (c_c) multiplied by the effective switching factor Eta (η), as in (3.2). Equation (3.2) represents the decoupling technique that models the total effective capacitance c_t which is a widely accepted method of approximation for extracting the effective interconnects capacitances [37–39]. It has been shown that a typical value of 2 for η is practical when signal transition times is adequately fast [37, 40, 41]. The value of η, in the presence of aggressive crosstalk in DSM designs, can reach the value of 4 [26, 39, 42].

$$c_t = c_w + \eta c_c \quad (3.2)$$

An important point with respect to the interconnect delay model given by (3.1), is the driver output resistance R_d, which is a supply voltage-dependent parameter [34]. Each operating state i is associated with an operating supply voltage V_{ddi}, in this case the driver output resistance R_d, at the operating state i, is approximated by [34]:

$$R_{di} = \mu \underbrace{\left[\frac{V_{ddi}}{(V_{ddi} - V_t)^{\alpha}}\right]}_{N_i} \quad (3.3)$$

Table 3.1 Summary of the definitions and the notations used throughout this chapter

Notation	Meaning
l	Interconnect length
w	Interconnect width
h	Size of interconnect repeaters
k	Number of interconnect repeaters
R_d	Output resistance of a min-sized repeater
C_g	Input capacitance of a min-sized repeater
C_d	Output capacitance of a min-sized repeater
C_b	$C_b = C_g + C_d$
r_w	Interconnect resistance (unit length)
R_w	Interconnect resistance, $R_w = r_w * l$
c_w	Interconnect capacitance (unit length)
C_w	Interconnect capacitance, $C_w = c_w * l$
c_c	Interconnect coupling capacitance (unit length)
C_c	Interconnect coupling capacitance, $C_c = c_c * l$
η	Signal coupling switching factor
c_t	Effective interconnect capacitance (unit length), $c_t = c_w + \eta c_c$
C_t	Effective Interconnect capacitance, $C_t = c_t * l$
R_{di}	Output resistance of a min-sized repeater at system state i
V_{ddi}	The operating supply voltage at system state i
V_t	Device threshold voltage
α	Device saturation index
μ	Device resistance coefficient
R_{d0}	Nominal output resistance of a min-sized repeater
s_i	System operating state i
S_i	Normalized scaling factor by which R_{di} scales with V_{ddi}
h_i	Size of repeaters, needed by system state i
k_i	Number of repeaters, needed by system state i
h_{il}	Size of repeaters, needed by state i, for latency objective
k_{il}	Number of repeaters, needed by state i, for latency objective
h_{if}	Size of repeaters, needed by state i, for frequency objective
k_{if}	Number of repeaters, needed by state i, for frequency objective
L_i	Interconnect latency at system state i
f_i	Interconnect operating frequency at system state i
α_s	Wire switching activity factor
ϑ	The product of h and k, $\vartheta = h^*k$
N	Number of interconnects in parallel
A_R	Area overhead of a min-sized repeater
τ_d	Relative RC time constant due to repeaters
τ_w	Relative RC time constant due to interconnects

In (3.3), μ is a coefficient that incorporates the design and the technology parametric values. Equation (3.3) shows that the driver output resistance is supply voltage-dependent and varies with N_i. Accordingly, at the nominal operating conditions, $R_{d0} = \mu * N_0$ (R_{d0} and N_0 represent the nominal values of R_{di} and N_i respectively). By substituting μ into (3.3), we conclude:

$$R_{di} = R_{d0} * S_i \tag{3.4}$$

where:

$$S_i = \underbrace{\frac{V_{ddi}}{(V_{ddi} - V_t)^\alpha}}_{N_i} / \underbrace{\frac{V_{dd0}}{(V_{dd0} - V_t)^\alpha}}_{N_0}$$

In (3.4), (N_i/N_0) is the *normalized scaling factor by which the output resistance R_{di} scales* due to supply voltage changes. We call this scaling factor S_i. Now, if we rewrite (3.1) using the scaling factor S_i, we will have

$$T_i(l, w, h, k) = 0.69[R_{d0}S_i](C_b + C_t/hk) + 0.38R_w(C_t/k^2 + 1.81C_gh/k) \tag{3.5}$$

Equation (3.5) is an *intelligent* line-delay model that only reflects the *nominal design parameters*, assuming that other design elements are supply voltage-independent. Note that R_{d0} is the driver output resistance at the nominal supply voltage for a given design, for which the key technology parameters can be obtained from predictive models [43], ITRS [44], or by manual analysis.

In a power-managed microsystem, each operating state i denoted by s_i, is associated with a different supply voltage V_{ddi} (referred to as dynamic voltage scaling (DVS)) and/or with a different operating frequency f_i (referred to as dynamic frequency scaling (DFS)). Now assuming that in a power-managed microsystem subject to operate at multiple states, at each state, various designs attribute for repeaters (i.e., h and k), may be needed to satisfy some performance objectives; in this case, the given interconnect performance model (3.5), can be modified to:

$$T_i(l, w, h, k) = 0.69\,\underbrace{R_{d0}(C_b + C_t/h_ik_i)}_{\tau_d}\,S_i + 0.38\,\underbrace{R_w(C_t/k_i^2 + 1.81C_gh_i/k_i)}_{\tau_w} \tag{3.6}$$

The difference between the performance models (3.5) and (3.6) is that in the latter, distinct repeater design attributes (h_i, k_i) are coupled with operating state s_i.

In (3.6), τ_d and τ_w, respectively, represent the relative RC time constants due to repeaters and interconnect. Normalizing (dividing) the performance model (3.6) by its value at the nominal operating state ($T_0 = T_{i|i=0}$), regardless of the repeaters repeater design attributes (i.e., (h_i, k_i)), yields:

$$T_{ni}(l, w, h_i, k_i) = \underbrace{\left(\frac{1}{1 + 0.55\frac{\tau_w}{\tau_d}}\right)}_{\tau_{d-r}} S_i + \underbrace{\left(\frac{1}{1 + 1.81\frac{\tau_d}{\tau_w}}\right)}_{\tau_{w-r}} \tag{3.7}$$

In (3.7) the interconnect/wire delay portion (τ_{d-r}) and the repeater (driver) delay portion (τ_{w-r}) are obtained by means of dividing the delay components τ_d and τ_w, by the total delay time at the nominal operating state (assuming a same interconnect design).

The *intelligent* performance models (3.6) and (3.7) will be utilized later, for the design and management of interconnects in power-managed microsystems.

3.2.2 The Impact of Technology Parameters on the Intelligent Interconnect Performance Models

A 45-nm technology node based on the Berkeley predictive technology model (BPTM) is used throughout this chapter [43, 45]. The associated parameters for the adopted technology are given in Table 3.2 [36, 43–46].

The interconnect repeaters considered in this chapter, will be implemented as CMOS inverters. The repeaters are designed to be symmetrical with respect to their rising and falling signal transition times. The PMOS to NMOS transistor ratio, for this design technology, is considered to be 2.5.

The supply operating range is considered to be 0.4–1.0 V which implies perfect *super-threshold* operating rang with *no need* for standard device library optimizations or modifications [47]. It has been reported that 0.4 V supply voltage is a solution that delivers (close to) energy-optimal design, especially adopted for the synthesis of low duty cycle applications where mainly require ultra-low voltage operation [27, 48–50].

Another important parameter to be discussed is α, the velocity saturation index. The value of α in typical full-swing super-threshold designs is smaller than 2 [33].

Table 3.2 45 nm node technology parameters

V_{dd0} (V)	1.0
V_t (V)	0.292
r_w (Ω/mm)	129
c_w (fF/mm)	44
c_c (fF/mm)	10
R_{d0} (KΩ)	15.7
c_g (fF)	0.45
c_d (fF)	0.41
A_R (μm^2)	0.034

In the case of near-threshold (approaching sub-threshold) designs, the value of α is better approximated by a value close to 2 [34]. In general, there is no definite closed-form value or exact formulation for α, and its value varies according to process technology, design parameters, as well as *the supply voltage* [30, 34, 51]. For this work, we performed some simulation-based analyses with the selected technology, in order to extract the correct profile of α.

The timing behavior for a cascade of minimal-sized symmetrical inverters for the given operating range 0.4–1 V has been depicted in Fig. 3.2 subject to a step input stimuli. We used the *intelligent* model (3.7), to extract the approximate value for α from HSPICE simulation results. When the circuit is at the nominal supply voltage, i.e., 1 V, the value of α that provides the best approximation is $\alpha \approx 1.2$. It is of interest however, that the value of α, providing the best performance fit increases as the supply voltage decreases. According to Fig. 3.2, $\alpha \approx 2$ is close to the best value to predict delay scaling over a range of supply voltages when the supply is scaled from 1 to 0.4 V. Note that smaller values of α can lead to predicted delay scaling much lower than simulated delays when V_{dd} is reduced. The values of α that provide the best delay approximations with (3.7) as compared to the full circuit simulation when V_{dd} is varied between 0.4 and 1 V are given in Table 3.3. This information is useful when modeling, designing, and managing power-managed VLSI Microsystems.

An important fact with respect to Table 3.3 is that, the (α, V_{dd}) pairs in this table are coupled (since α is a function of V_{dd}); that means for each supply voltage operating region there is an associate α; if in the analysis the obtained pair does not comply with Table 3.3, the analysis should be repeated based on the fundamental variable, i.e., V_{dd}. A third-degree curve-fitted polynomial function that characterizes the relation of V_{dd} and α, according to Table 3.3, can be represented by

$$F_\alpha(V_{dd}) = -5.55V_{dd}^3 + 12.38V_{dd}^2 - 9.99V_{dd} + 4.37 \tag{3.8}$$

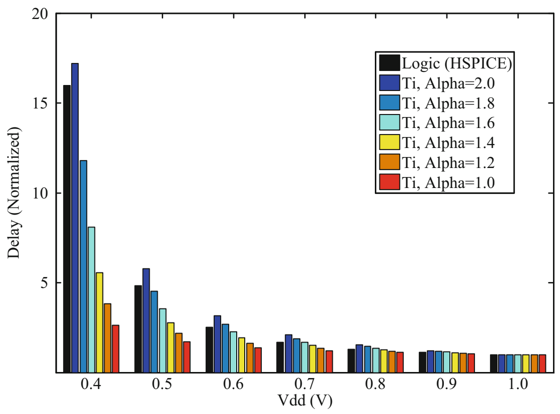


Fig. 3.2 Extracting the approximate value for velocity saturation index that best matches HSPICE transient simulation results. The delays are normalized by their respective nominal values at $V_{dd} = 1$

Table 3.3 Detailed profile of the saturation index over the full operating range 0.4–1.0 V

V_{dd} (V)	α (HSPICE)	F_α (model)
0.4	2.0	2.00
0.5	1.8	1.78
0.6	1.6	1.63
0.7	1.6	1.54
0.8	1.4	1.46
0.9	1.4	1.36
1.0	1.2	1.20

Comparison of HSPICE and the described curve-fitted function

Table 3.3 characterizes the *intelligent* relation between V_{dd} and α that obtained from HSPICE, as well as, the quantified value of the described curve-fitted functional model F_α. Either of these relations, i.e., (α, V_{dd}) or $(F_\alpha(V_{dd}), V_{dd})$, can be utilized *intelligently* in the processes of modeling, design, and management of interconnections in power-managed systems.

According to Fig. 3.2, some residual error exists for all values of α, assumed to hold over the same operating range, for performance estimation. Figure 3.3 depicts the average error observed, when different quantities for α are used for estimating performances *over the full operating range*. According to Fig. 3.3, $\alpha \approx 1.8$ produces the minimum average error (less than 4 % smaller than the error observed when $\alpha = 2$) over the full operating range 0.4–1 V. However, $\alpha \approx 2$, provides the best approximation of the maximum possible scaling, which is a very important design characteristic, especially for the synthesis of low duty cycle applications. Figure 3.3 can help for performance modeling, design, and management of power-managed systems, when a single value of saturation index, need to be used. Further in-depth analysis and the applications of saturation index modeling in power-managed systems, can be found in [30].

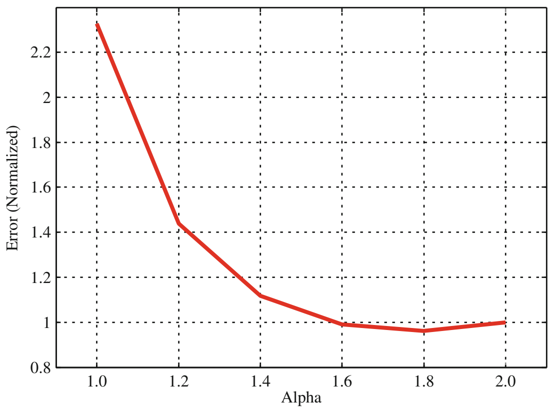


Fig. 3.3 Average error when the performance is estimated with different values of assumed to hold over the full operating range 0.4–1 V, normalized with respect to $\alpha = 2$

3.2.3 The Implications of the Performance Models

Figure 3.4 depicts τ_{d-r} in interconnect-centric designs based on the same 45 nm technology node using (3.7). According to Fig. 3.4, *as interconnects become longer and the driver (repeater) size increases, τ_{d-r} tends to decrease*. Now, Fig. 3.5 reports T_{ni} as a function V_{dd} of and τ_{d-r}. In the domain depicted in the figure, T_{ni} passes 17 for logic (e.g., an ALU); whereas, for a typical interconnect-centric subsystem (e.g., a bus) with a $\tau_{w-r} = 50\,\%$ ($\tau_{d-r} = 50\,\%$), T_{ni} drops below 10. It is of interest that based on (3.7), smaller τ_{d-r} implies less delay scaling with voltage. Thus, *in advanced nanometer technologies, as wires become longer and the frequency of operation increases, the performance of microsystems does not scale down as much with voltage scaling.*

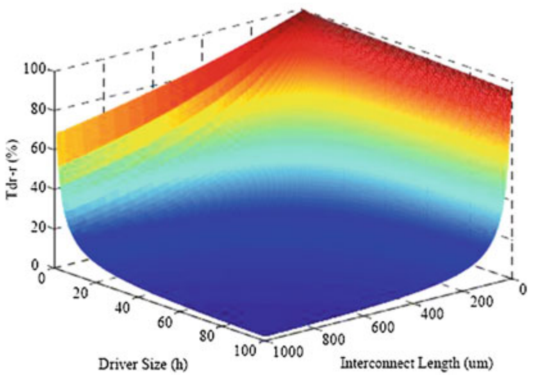


Fig. 3.4 The driver delay portion obtained based on the driver size (h) and the global interconnect length in 45 nm ($k = 1, \alpha = 2$)

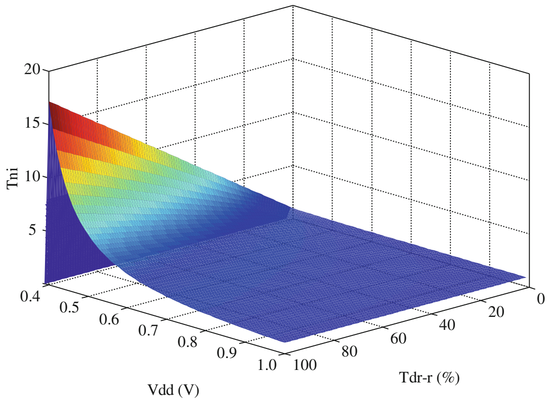


Fig. 3.5 Subsystems delay subject to voltage scaling, based on the driver delay portion (τ_{d-r}) and the supply voltage, ($k = 1, \alpha = 2$)

3.3 Intelligent Design of Interconnections in Power-Managed SoCs

In this section, utilizing the performance models given in the second section, we describe methods to explore the design space for interconnect in power-managed microsystems. These intelligent methods guarantee that the designed interconnects will meet the performance objectives in all the system operating states.

3.3.1 Designing Interconnections with Latency Objectives

Let us consider a power-managed microsystem, projected to run at a given set of operating states with certain latency objectives, which can be posed by system cycle-time constraints, and is denoted by $\{(L_i, V_{ddi})|i \in \text{operating states } 0\ldots n\}$. For this, the interconnect repeaters should be designed in a way to meet the target interconnect latency objectives at the projected system processing states.

The line latency is k times the delay of a line segment. In this case, this latency in power-managed systems at the operating state i, is modeled by

$$L_i \geq k_i * T_i \tag{3.9}$$

In (3.9), T_i replaces (3.6). Equation (3.9), gives $\{(h_{il}, k_{il})|i \in \text{operating states } 0\ldots n\}$ that characterizes the repeater design space for which target interconnect latencies are met at the operating state i. Note that, according to this equation, the obtained solutions are *upper-bounded*, since $k_i \leq L_i/T_i$. The interconnect latency objectives must be met in all the design operating states. Thus, the latency-aware interconnect design subspace in a power-managed system is obtained through

$$\{(h_{PM-L}, k_{PM-L})\} = \{\{(h_{0l}, k_{0l})\} \cap \cdots \cap \{(h_{nl}, k_{nl})\}\} \tag{3.10}$$

3.3.2 Designing Interconnections with Frequency Objectives

Let us consider a power-managed microsystem, projected to operate at a given set of operating states with certain operating frequency objectives, which can be posed based on system workload model, and is denoted by $\{(f_i, V_{ddi})|i \in \text{operating states } 0\ldots n\}$. For this, the interconnect repeaters should be designed to meet the target operating frequencies at the projected system operating states. We target interconnect operating frequency instead of interconnect bandwidth, since our analyses are based on $T_{50\,\%}$, whereas for bandwidth, the analyses are based on signal full transition time [18]. Note that as defined, the latency and bandwidth constraints are decoupled, as multiple bus cycle times (inverse of bus frequency) may fit in one bus latency time.

This model covers the case where a bus is pipelined with multiple bits transmitted in a single latency time.

To solve this design problem, (3.6) is adopted to extract the required design values (h_{if}, k_{if}) that enable interconnects to meet target operating frequencies $\geq f_i$ at the operating state i. Solving (3.6) for k_i as a function of h_i, assuming the operating state i, results into

$$k_i = K(h_i) = \frac{\beta + \sqrt{\beta^2 - \gamma R_w C_t}}{-2\gamma} \tag{3.11}$$

where:

$$\beta = 0.69[(R_{d0} * S_i)C_t/h_i + h_i R_w C_g]$$

$$\gamma = 0.69(R_{d0} * S_i)C_b - T_i$$

In (3.11), $T_i = 1/f_i$. The solution gives $\{(h_{if}, k_{if}) | i \in \text{operating states } 0 \ldots n\}$ that defines the design space. Note that the *lower bound* of the design space is obtained according to f_i and the design space for operating frequencies greater then f_i is also valid. In order to ensure that interconnect operating frequency objectives are met in all the design operating states, the solution should belong to the obtained solutions of all the operating states. In this case, the interconnect design subspace to meet the operating frequency objective, in the power-managed systems becomes

$$\{(h_{PM-f}, k_{PM-f})\} = \big\{\{(h_{0f}, k_{0f})\} \cap \cdots \cap \{(h_{nf}, k_{nf})\}\big\} \tag{3.12}$$

Depending on the interconnect design objectives, *the valid portion of the design space* $\{(h_{PM}, k_{PM})\}$ for interconnect repeaters in power-managed systems is defined by (3.10) when the design is only constrained by latency objectives (case I). It is defined by (3.12) when the only constraints are operating frequency objectives (case II). Finally, the intersection of the two defines the space of valid solutions when both sets of objectives must met simultaneously (case III). In other words,

$$\{(h_{PM}, k_{PM})\} = \begin{cases} \{(h_{PM-L}, k_{PM-L})\} & \text{(I)} \\ \{(h_{PM-f}, k_{PM-f})\} & \text{(II)} \\ \{(h_{PM-L}, k_{PM-L})\} \cap \{(h_{PM-f}, k_{PM-f})\} & \text{(III)} \end{cases} \tag{3.13}$$

3.3.3 Designing Interconnections with Energy Objectives

The line average energy model in a power-managed VLSI system, running at the operating state i, in a unified form, is given by [52–54]:

$$E_i = \alpha_s[C_t + \vartheta_{PM} C_b] V_{ddi}^2 \tag{3.14}$$

In this equation, α_s denotes the line (wire) switching activity factor and $\vartheta_{PM} = h_{PM} k_{PM}$. The only design variable in this model is ϑ_{PM}. Also, as can be seen from (3.14), the line energy requirement increases monotonically with ϑ_{PM}, since $\partial A / \partial \vartheta_{PM}$ is a positive value. *Therefore, the minimum energy is obtained for the minimum value of ϑ_{PM} in the valid portion of the design space* $\{(h_{PM}, k_{PM})\}$.

3.3.4 Designing Interconnections with Area Objectives

The spanning area overhead due to repeater insertion in interconnects has been reported in [17, 46]:

$$A = N \vartheta_{PM} A_R \tag{3.15}$$

In which N is the number of interconnects in parallel, A_R is the area overhead of a minimum-sized repeater defined in a given technology, and $\vartheta_{PM} = h_{PM} k_{PM}$. The only design variable in this model is ϑ_{PM}. Also, as can be seen from (3.15), the area also increases monotonically with ϑ_{PM}, since $\partial A / \partial \vartheta_{PM}$ is a positive value. *Therefore, the minimum area overhead in interconnects is similarly obtained for the minimum value of ϑ_{PM} in the valid portion of the design space* $\{(h_{PM}, k_{PM})\}$.

Due to the above observations, we consider ϑ_{PM} as a *figure of merit* for energy-optimal and area-optimal design solution extraction from the interconnect design space $\{(h_{PM}, k_{PM})\}$ for power-managed VLSI.

3.4 Intelligent Management of Interconnections in Power-Managed SoCs

In the previous section, *intelligent* design methods were introduced that guarantee the design of interconnects that meet the system performance objectives of all the system operating states. The goal of this section is to describe *intelligent* methods to manage/control the designed interconnect in power-managed microsystems for efficient operation. In order to introduce some power management design scheme for interconnections some *intelligent* design metrics and formulations are described.

3.4.1 The DVS Design Metrics

According to (3.7), each distinct segment in a power-managed system with scaling rate S_i (at the operating state i), has an effective scaling rate proportional to $\tau_{d-r} * S_i$. Since S_i is the global scaling rate which applies uniformly to the system component, then τ_{d-r} becomes *a distinctive factor for each interconnect performance* subject to dynamic state transition. We call the factor τ_{d-r}, *PLD*, which designates the *Portion of the Logic Delay (PLD)* in that interconnect segment. Accordingly, *PLD* is considered as a *design metric*, which determines the performance of an interconnect-centric component in a power-managed system subject to dynamic state transitions.

According to (3.7), each segment has also a unique complementary τ_{w-r} term. Similarly, we call τ_{w-r}, *PWD*, the *Portion of the Wire Delay (PWD)* in that interconnect segment. This concept can be formulated by (3.16). Accordingly, *PWD* could be also considered a *design metric*. Note that either metrics may be employed, as they are complementary.

$$PLD + PWD = 1 \tag{3.16}$$

3.4.2 A Compact DVS Delay Model Using the Design Metrics

Based on the above formulations, the delay model given by (3.7) using design metrics *PLD* and *PWD*, as well as (3.16), may be reformulated as (3.17).

$$\begin{aligned} T_{ni} &= PLD * S_i + [1 - PLD] \\ T_{ni} &= PLD * [S_i - 1] + 1 \end{aligned} \tag{3.17}$$

3.4.3 Scaling Limit for Hazard-Free System Operation

In VLSI, wire delays do not scale with the supply voltage (at least in its first approximation [34]). This implies that the system scaling (T_{ni}) is limited by interconnects. Therefore, in a microsystem subject to DVS, *the segment that has the maximum interconnect delay portion* (i.e., maximum *PWD* and hence the minimum *PLD*), *becomes the bottleneck against hazard-free scaling* (assuming a minimum value is defined for V_{ddi} which results into a maximum value for S_i). Utilizing Eq. (3.17), this limit can be formulated as

$$T_{n\text{-}limit} = PLD_{min} * [S_{max} - 1] + 1 \tag{3.18}$$

where PLD_{min} is the *PLD* of the segment in the system which has the minimum logic delay portion. In this equation, $T_{n\text{-}limit}$ indicates the scaling limit that can be reached by that DVS Microsystem (or its component) *to avoid hazard conditions*. Note that the scaling rate S_{max} is obtained with the supply voltage set at its minimum acceptable value. The minimum acceptable value for the supply voltage can be determined by the design and/or the process technology. For system components that mainly include logic like an ALU, $PLD_{min} \cong 1$ and therefore $T_{n\text{-}limit} \cong S_{max}$. *This is the same limit posed by the given process technology*. In interconnect-centric components such as in CDNs, NoCs, or buses, $PLD_{min} \ll 1$ and therefore $T_{n\text{-}limit} \ll S_{max}$. Therefore, *in interconnect-centric components, the system scaling limit can reach values significantly lower than the limit dictated by the process technology*.

3.4.4 Selecting Supply Voltages

Recalling from the fourth section, (3.17) is a reformulated *intelligent* delay model by which a target delay (or a target frequency) may be obtained. Equation (3.17) can also be considered as a function of independent variables. Such function, in its most concise form, is defined by

$$T_{ni} = F_{Ti}(PLD, V_{ddi})$$

In the definition of function F_{Ti}, it is assumed that PLD and V_{ddi} are considered to be the only variables on which the function depends. Now, if we compute the inverse of F_{Ti} w.r.t. V_{ddi}, we will have

$$V_{ddi} = F^{-1}_{Ti|V_{ddi}} = F_{Vi}(PLD, T_{ni}) \tag{3.19}$$

The function F_{Vi} defines an *intelligent* relation by which the supply voltage, based on a target delay *and a given PLD*, is obtained. *The obtained supply voltage, based on* (3.19), *is the solution for precise/efficient interconnect functionality at the operating state i. This supply voltage may be different than the one initially introduced by the system specifications which mainly consider logic*. An important fact about F_{Vi} is that its formation is heavily influenced by the value of α. When α equals to 1 or 2, the resulting quadratic expression has a tractable analytic solution expressed by (3.20) and (3.21).

3.4.5 The Interpolation Method

Formulating F_{Vi} for $1 < \alpha < 2$ is not a trivial task. In such cases, we may leverage some form of interpolation for quantifying F_{Vi} (i.e., quantifying V_{ddi} when $1.0 < \alpha < 2.0$). For this, we initially explore the two bounds, i.e., the behavior of (3.20) and (3.21), for a large spectrum of designs in 45 nm technology. Figure 3.6 depicts this exploration. According to this figure, the two configurations α for equal to 1 and 2

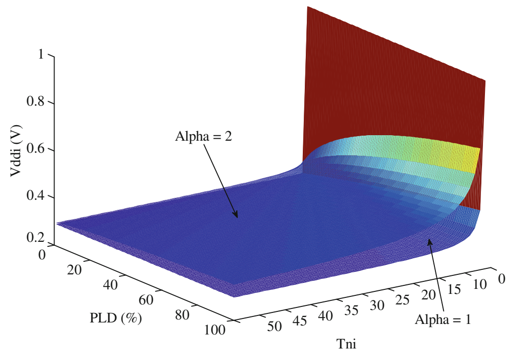


Fig. 3.6 Quantifying F_{Vi} for a large spectrum of designs for $\alpha = 1$ and $\alpha = 2$ in 45 nm technology

create bounds for trajectories with similar behavior, for a wide range of given design points (PLD, T_{ni}). Because of this behavior, averaging as a means for interpolation is attractive due to its simplicity and its efficiency. The procedure for formulating F_{Vi} for $1 < \alpha < 2$ is as follows. Knowing that F_{Vi}, based on the technology defined parameters and for α equal to 1 and 2 is bounded (referring to Fig. 3.6), we first define α_1 and α_2 to be in the range from 1 to 2 (as relevant to the context). Thus, they are also bounded and therefore their average $\left(\frac{\alpha_1+\alpha_2}{2}\right)$ is in the same range and is bounded. Accordingly, F_{Vi} for α equal to $\left(\frac{\alpha_1+\alpha_2}{2}\right)$ is bounded. In this case, we interpolate F_{Vi} by:

$$F_{Vi|\alpha=1} = \frac{N_0 V_t (T_{ni} + PLD - 1)}{N_0(T_{ni} + PLD - 1) - PLD} \tag{3.20}$$

$$F_{Vi|\alpha=2} = \frac{\begin{array}{c} PLD + 2N_0 T_{ni} V_t + 2PLDN_0 V_t - 2N_0 V_t \\ +\sqrt{PLD^2 + 4PLDN_0 T_{ni} V_t + 4PLD^2 N_0 V_t - 4PLDN_0 V_t} \end{array}}{2N_0(T_{ni} + PLD - 1)} \tag{3.21}$$

$$F_{Vi|\alpha} = \left(\frac{\alpha_1 + \alpha_2}{2}\right) \approx \frac{F_{Vi|\alpha=\alpha_1} + F_{Vi|\alpha=\alpha_2}}{2} \tag{3.22}$$

We may use (3.22) to *intelligently* quantify F_{Vi} for diverse quantities of α when $1 < \alpha < 2$. We may commence the iterative process of the averaging method with α equal to 1 and 2, utilizing (3.20) and (3.21), and iterate to converge to the desired α.

3.5 The Flow

The *intelligent* modeling, design, and management methods described earlier are encompassed in a complete flow with two main phases: phase #1 includes the design and phase #2 includes the management of interconnects, as shown in Fig. 3.7.

In the design phase, based on the *intelligent* modeling and design methods presented in the second and third sections, the valid portion of the design space for interconnects is explored. According to Fig. 3.7, initially the flow accepts the interconnect length l and width w of the wire, as well as the set of the system operating states $\{s_i\}$ with their design objectives $\{s_i = (L_i, f_i, V_{ddi})|i \in \text{operating states } 0 \ldots n\}$. Later, utilizing the presented models, the two design subspaces for interconnects which meet the latency and operating frequency objectives, $\{(h_{PM-L}, k_{PM-L}\}$ (using (3.9) and Table 3.3), and $\{(h_{PM-f}, k_{PM-f}\}$, (using (3.11) and Table 3.3), are obtained. Later, the valid portion of the system interconnects design space $\{(h_{PM}, k_{PM}\}$ based on either or both of these design subspaces (using (3.13)) is deduced. In the valid portion of the interconnect design space $\{(h_{PM-f}, k_{PM-f}\}$, the optimal design point, with respect to both energy and area efficiencies, is the one which has minimum $\vartheta(h * k)$ value. In practice, the number of interconnect repeaters (k) should be an integer number; and when no bit inversion is required, should be an even integer number. Since the analytical formulations may result into solutions that do not meet this objective; eventually, the obtained solution may need to be updated. For this purpose, the design point closest to the optimal design point in the design space $\{(h_{PM}, k_{PM}\}$ that has an (even) integer k, with the smallest ϑ, is reported as the design space exploration solution. At this point, the flow *intelligently* delivers the solution (h, k) of the given interconnect.

In the management phase, based on the solution (h, k) obtained from the design phase, initially the nominal interconnect delay (using (3.5)) and its *PLD* (using (3.7)) are both calculated. The required scaling at the operating states, based on (3.5) and (3.7) are obtained afterward. Later, the limit of scaling, which could be reached by the given interconnect is calculated, using (3.18). This limit is important for the design cases where timing violations should be avoided, especially in the cases of critical paths, race-free subsystems, etc. Afterward, this limit is compared with the given system scaling requirements. For each operating state i, if the required performance is not greater than the scaling limit permitted by interconnect, the supply voltage, using (3.19) and (3.22) when needed, is obtained; otherwise, the minimum defined supply voltage is assigned. At this point, the flow *intelligently* delivers the solution $\{V_{ddi}\}$, which implies the set of supply voltages required by that interconnect in all the system operating states.

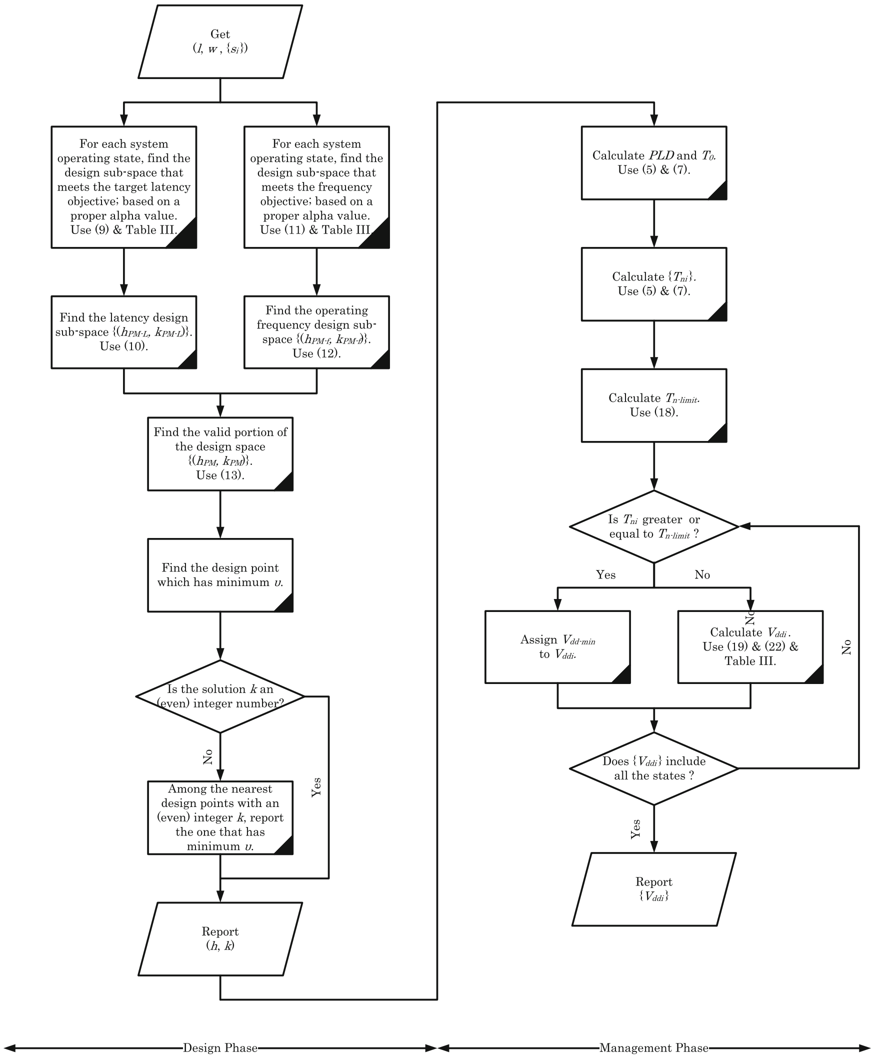

Get
$(l, w, \{s_i\})$
For each system operating state, find the design sub-space that meets the target latency objective; based on a proper alpha value. Use (9) & Table III.
For each system operating state, find the design sub-space that meets the frequency objective; based on a proper alpha value. Use (11) & Table III.
Find the latency design sub-space $\{(h_{PM\text{-}L}, k_{PM\text{-}L})\}$. Use (10).
Find the operating frequency design sub-space $\{(h_{PM\text{-}f}, k_{PM\text{-}f})\}$. Use (12).
Find the valid portion of the design space $\{(h_{PM}, k_{PM})\}$. Use (13).
Find the design point which has minimum υ.
Is the solution k an (even) integer number?
No
Yes
Among the nearest design points with an (even) integer k, report the one that has minimum υ.
Report
(h, k)
Calculate PLD and T_0. Use (5) & (7).
Calculate $\{T_{ni}\}$. Use (5) & (7).
Calculate $T_{n\text{-}limit}$. Use (18).
Is T_{ni} greater or equal to $T_{n\text{-}limit}$?
Yes
No
Assign $V_{dd\text{-}min}$ to V_{ddi}.
Calculate V_{ddi}. Use (19) & (22) & Table III.
No
Does $\{V_{ddi}\}$ include all the states ?
Yes
Report
$\{V_{ddi}\}$
Design Phase
Management Phase

Fig. 3.7 The interconnect design and management flow

3.6 Case Study: Intelligent Design and Management of Integrated Buses in a Power-Managed SoC

In this section, in order to validate the presented design and management schemes, a design case is chosen as a case study. We employ the presented methods for the design and management of integrated buses, in a power-managed platform. HSPICE simulations are performed to confirm the validity of the presented methods.

Let us consider a power-managed system, running at two distinct active operating states, for which the operating frequency and latency objectives, as well as the logic operating states supply voltages are specified as $\{s_0 = (2\,\text{GHz}, 2\,\text{ns}, 1\,\text{V}), s_1 = (500\,\text{MHz}, 10\,\text{ns}, 400\,\text{mV})\}$. The target communication mechanisms are two integrated 8-bit buses whose length and width are given as Bus #1 (10 mm, 8X) and Bus #2 (10 mm, 4X), which aim to meet the specified performance objectives, given earlier. Bus #1 is twice wide as Bus #2. The bus design model follows the methods adopted in [19, 55–59].

A 45 nm technology node, with the parameters given in Table 3.2, is adopted. Table 3.3 has been used to employ the suitable values for α in each operating state. It is also assumed that only even number of repeaters (k) can be used (no bus inversion).

We considered wide global wiring structures, normally used in the design of synchronization or communication mechanisms in VLSI systems. We first perform the design space exploration for a bus with 8 X w-min line widths. In the second design, we consider the bus line widths reduced to half.

In the design phase of the flow, the design space exploration is performed as follows. Initially, the solution boundaries that meet latency objectives for the two operating states, for the buses are found using (3.9). Recall that the associated boundaries are upper-bounds. Those bounds are shown by the red curve for s_0 (called s_{0-L}) and by the blue curve for s_1 (called s_{1-L}) in Fig. 3.8a for Bus #1, and in Fig. 3.8b for Bus #2. All the buses defined by the solutions inside the almost square regions defined by the red and blue boundaries meet the latency objective. The intersection of the two regions becomes the solution $\{(h_{PM-L}, k_{PM-L})\}$.

Later, (3.11) is employed to obtain the boundaries of the design space for operating frequency objectives. Those operating frequency-related boundaries are the red dotted curve for s_0 (called s_{0-f}) and the blue dotted curve for s_1 (called s_{1-f}) as shown in Fig. 3.8c for Bus #1 and in Fig. 3.8d for Bus #2. Recalling that larger h and k give higher operating frequencies, thus clearly the obtained boundaries are also bounds of the design space, with all solutions above each line meeting the respective frequency objective. The intersection of s_{0-f} and s_{1-f} defines $\{(h_{PM-f}, k_{PM-f})\}$.

In Fig. 3.8, the design spaces have been explored based on the two boundary values of η. When wire/interconnects are considered to be fully shielded/isolated, η equals zero and in the case of existing fully aggressive adjacent lines, η may reach four. According to Fig. 3.8, we notice that the design space shrinks when the crosstalk effects increase (η>0).

Before describing the valid portion of the bus design space, based on the results of the above explorations, let us evaluate the behavior of ϑ in the given operating states. Figure 3.9a, b depict the behavior of ϑ in the design space explorations of Bus #1 and Bus #2, respectively. According to this figure, ϑ increases with h; accordingly, the design points with smallest h, located at the most left side in the design spaces, are the solution candidates to offer least energy and area overhead. Also, when $\eta = 4$, almost twice larger ϑ_{PM}(hence twice larger bus overhead) is desired with respect to the case when $\eta = 0$, to design the same buses. In other words, twice overhead is associated with the interconnect/bus design, if shielding is not properly applied.

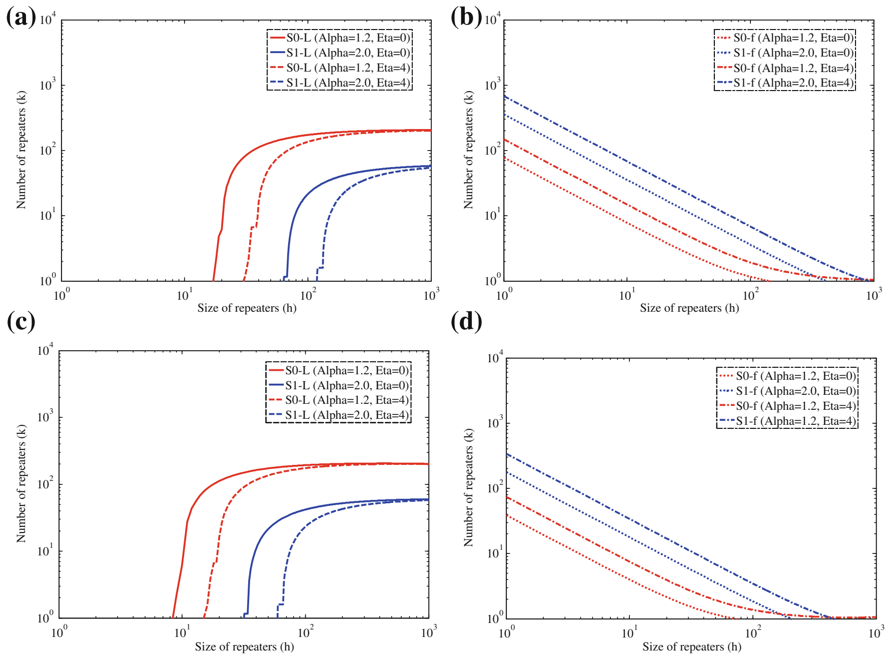


Fig. 3.8 The design space exploration to meet the latency objectives in **a** Bus #1, **b** Bus #2; to meet the operating frequency objectives in **c** Bus #1, **d** Bus #2

Lastly, the intersection of $\{(h_{PM-L}, k_{PM-L})\}$ and $\{(h_{PM-f}, k_{PM-f})\}$ defines the valid portion of the design space for $\{(h_{PM}, k_{PM})\}$ the bus, meeting all design constraints shown in Figs. 3.9c, d for Bus #1 and Bus #2, as the region encompassing rigid horizontal lines, assuming k as an even integer number.

Based on Fig. 3.9c, the theoretical design solution of this case study becomes the design point (138, 4.9). This design point is the intersection of the latency and operating frequency design boundaries of s_1. *Nevertheless, the design space, which here is imposed by* s_1 *may vary, when the design requirements changes*. Note that $k = 4.9$, the number of repeater (stages), calculated with continuous analytic expressions, is not an integer. Imposing the constraint that k must be an even integer as no bus inversion is desired, the feasible design solution with the smallest ϑ_{PM} becomes (168, 6). This solution is compared with four acceptable adjacent design points ($\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{e}$), representing each of the four neighboring design regions, as shown by the design point $\boldsymbol{a}$ in Fig. 3.9c, in order to highlight the efficiency of the presented methods.

The same design exploration is performed for a bus with its line widths reduced to half. The analytical exploration analogous to the first scenario becomes the design point (69, 4.9) as depicted in Fig. 3.9d. It is of interest that the solutions are reduced to half, with respect to the first design case. The feasible design space solution with the smallest ϑ_{PM}, in this design case, becomes (84, 6) according to the presented method, also as shown as the design point $\boldsymbol{f}$ in Fig. 3.9d. This solution is compared

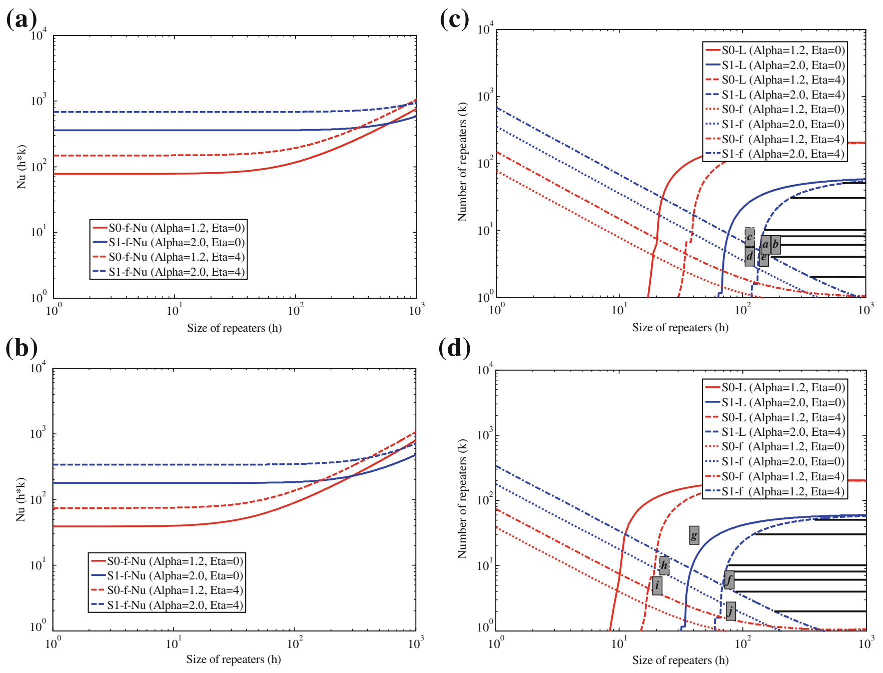


Fig. 3.9 The required ϑ in the design space exploration of **a** Bus #1 and **b** Bus #2. The design spaces of **c** Bus #1 and **d** Bus #2

with four acceptable neighboring design points (***g***, ***h***, ***i***, and ***j***), representing four sampled design regions, as shown by Fig. 3.9d, in order to highlight the efficiency of the presented methods. Recall that the ten adopted design points are positioned as a dot and as the center of the corresponding characters in Figs. 3.9c, d.

Two distinct signal transition patterns were applied to the buses: fully odd signal transition patterns (like "00000000" to "11111111"), representing the best design case and correlating to $\eta = 0$ design variable, and fully differential signal patterns (like "01010101" to "10101010"), representing the worst design case and correlating to $\eta = 4$ design variable. The detailed HSPICE simulation results of the ten candidate design points are given in Table 3.4.

According to Table 3.4 and based on HSPICE simulations, for the design of Bus #1, the design point ***a*** meets all design objectives with the least energy and area overhead. Design point ***b*** meets all the design objectives but has higher energy and area overhead than the optimal design (design point ***a***). In this table, wherever a violation occurs (i.e., an objective is not met) or a result is suboptimal, the associated result is underscored. Design point ***c*** does not meet the latency objectives when the signal patterns are fully differential. According to Fig. 3.9c, the design point ***c***is outside the design space defined by s_{1-L} when η equals 4. Design point ***d*** does not meet the latency or frequency objectives when the signal patterns are fully differential.

Table 3.4 HSPICE simulation results of the two integrated bus structures (performance violations or suboptimal results are $\underline{\text{underscored}}$)

Bus #	**1**					**2**				
Design point	***a***	***b***	***c***	***d***	***e***	***f***	***g***	***h***	***i***	***j***
$\boldsymbol{h}$	168	170	120	120	160	84	40	30	20	80
$\boldsymbol{k}$	6	6	8	4	4	6	12	10	4	2
$\boldsymbol{\vartheta}$	1008	1020	960	480	640	504	480	300	80	160
$\boldsymbol{A(\mu m^2)}$	274	$\underline{277}$	261	130	174	137	130	81	21	43
s_{0-L} ***(ns), Odd***	0.24	0.24	0.30	0.32	0.25	0.23	0.43	0.56	0.73	0.28
s_{0-f}***(GHz), Odd***	24.4	24.6	26.3	12.4	15.4	25.5	27.4	17.7	5.46	7.14
s_{0-E}***(pJ), Odd***	15.3	$\underline{15.4}$	11.2	16.2	18.3	7.73	3.81	3.57	5.78	14.0
s_{1-L}***(ns), Odd***	5.75	5.70	8.02	7.30	5.67	5.84	$\underline{11.9}$	$\underline{14.8}$	$\underline{19.5}$	5.30
s_{1-f}***(MHz), Odd***	1.04	1.05	0.99	0.54	0.70	1.02	1.00	0.67	$\underline{0.20}$	$\underline{0.37}$
s_{1-E}***(pJ), Odd***	1.51	$\underline{1.52}$	1.11	1.81	1.96	0.76	0.37	0.38	0.77	1.69
s_{0-L}***(ns), Even***	0.39	0.39	0.53	0.54	0.45	0.39	0.73	1.00	1.48	0.55
s_{0-f}***(GHz), Even***	15.1	15.2	14.9	7.28	8.86	15.0	16.4	9.93	2.70	3.57
s_{0-E}***(pJ), Even***	21.1	$\underline{21.2}$	15.5	24.5	26.6	10.5	5.19	5.24	10.3	22.8
s_{1-L}***(ns), Even***	10.0	9.95	$\underline{14.0}$	$\underline{12.9}$	10.0	10.0	$\underline{20.8}$	$\underline{26.4}$	$\underline{35.9}$	9.53
s_{1-f}***(MHz), Even***	0.59	0.60	0.57	$\underline{0.30}$	$\underline{0.39}$	0.59	0.57	$\underline{0.37}$	$\underline{0.11}$	$\underline{0.20}$
s_{1-E}***(pJ), Even***	2.37	$\underline{2.38}$	1.75	3.10	3.24	1.19	0.59	0.64	1.41	2.97

Table 3.5 Extracted parameters for the management of the integrated buses

Bus	***Design***	***PLD (%)***	T_{0-f} ***(GHz)***	T_{0-L} ***(ns)***	T_{s-f} @ s_0	T_{s-L} @ s_0	T_{s-f} @ s_1	T_{s-L} @ s_1	$T_{s-limit}$
1	***a*** (168, 6)	86	15.1	0.39	7.55	5.12	30.2	25.6	14.9
2	***f*** (84, 6)	86	15.0	0.39	7.50	5.12	30.0	25.6	14.9

According to Fig. 3.9c, the design point ***d*** is outside the design spaces defined by s_{1-L} or s_{1-f} when η equals 4. Design point ***e*** does not meet the frequency objectives when the signal patterns are fully differential. According to Fig. 3.9c, the design point ***e***is outside the design space defined by s_{1-f} when η equals 4. All these design points however meet the latency and performance objectives when the signal transitions are not fully differential.

A similar situation exists in the design of Bus #2. According to Table 3.4, and also validated by HSPICE simulations, the design point ***f*** meets all the design objectives with the least energy and area overhead, as considered to be the feasible near-to-optimum design point. Design point ***g*** does not meet the latency objectives s_1 regardless of the signal patterns; according to Fig. 3.9d, the design point ***g*** is outside the design space defined by s_{1-L}. Design point ***h*** does not meet the latency objectives s_1 regardless of the signal patterns, nor the frequency objectives s_{1-f} when $\eta = 4$, i.e., when signal patterns are fully differential. Design point ***i*** does not meet the latency or the frequency objectives regardless of the signal patterns; according to Fig. 3.9d, the design point ***i*** is outside the design spaces defined by s_{1-L} or s_{1-f}. Design point ***j*** does not meet the frequency objectives s_{1-f} regardless of the signal transition patterns. According to Fig. 3.9d, the design point ***j*** is outside the design spaces defined by s_{1-f}. All these design points on which Bus #2 is designed however, meet the latency and performance objectives s_0.

According to Table 3.4, the reduction of the bus wire widths to half has resulted into the reduction of the energy and area overhead, for the same design requirements (for a correct comparison Bus #2 designed based on ***f*** should be compared to Bus #1 designed based on ***a***).

As for the control/management phase of the flow, the process is performed as follows. Initially, based on the obtained design solutions for the integrated Bus #1 as ***a***, and for Bus #2 as ***f***, the s as well as the nominal delays of the buses (here both the nominal bus operating frequency T_{0-f} and latency T_{0-L}), using (3.5) and (3.7) are calculated. Later, the scaling required at s_0 and s_1 for the operating frequency T_{s-f} and latency T_{s-L}, based on (3.5) and (3.7) are obtained. The limit of scaling for the buses, assuming a minimum defined supply voltage 400 mV, and using (18) is further obtained. All the values obtained with the above steps for the given integrated buses are given in Table 3.5. An important design note is that smaller scaling implies larger supply voltage; therefore, in order for both the frequency and latency objectives be met during DVS, we must perform the design exploration for supply voltage selection based on the smaller scaling rate, i.e., the worst case, here imposed by latency objectives as $T_{s-f}>T_{s-L}$. Another important fact with respect

Table 3.6 HSPICE simulation results of the two integrated bus architectures subject to DVS based on the obtained design parameters (performance violations or suboptimal results are underscored) S: HSpice and M: model

Bus #	***Parameters***	T_{s-f}@ s_0			T_{s-L}@ s_0			T_{s-f}@ s_1	T_{s-L}@ s_1
		α= 1	**α= 1.2**	**α= 2**	**α= 1**	**α= 1.2**	**α= 2**	**α= 2**	**α= 2**
1	$V_{dd}{}^M$	0.318	0.352	0.455	0.332	0.374	0.500	0.364	0.371
	L (ns) ^{S}	24.0	13.0	2.65	18.6	8.83	1.54	10.6	9.37
	f (GHz) ^{S}	0.249	0.460	2.262	0.32	0.67	3.89	0.56	0.63
	E (pJ) ^{S}	1.48	1.81	3.10	1.60	2.06	3.78	1.94	2.02
	E Save (X) ^{S}	9.88	8.07	4.83	9.07	7.14	4.21	1.20	1.16
2	$V_{dd}{}^M$	0.318	0.352	0.455	0.332	0.374	0.500	0.364	0.371
	L (ns) ^{S}	24.0	13.0	2.65	18.6	8.96	1.64	10.63	9.35
	f (GHz) ^{S}	0.24	0.46	2.25	0.32	0.66	3.64	0.564	0.641
	E (pJ) ^{S}	0.74	0.91	1.54	0.80	1.03	1.92	0.97	1.01
	E Save (X) ^{S}	9.88	8.07	4.83	9.07	7.14	4.21	1.20	1.16

to Table 3.5 is, as the two considered buses have the same *PLD*, the acquired supply voltages for various operating states become equal. This is not the general case; if various bus lengths existed, various *PLD*s would be resulted and therefore various supply voltages would be required.

Later, the supply voltages for all the operating states need to be obtained. The related obtained solutions are given in Table 3.6. For the purpose of s_0, using (3.20) and (3.21), the supply voltages based on $\alpha = 1$ and $\alpha = 2$ are obtained. Later, using (3.22) and in two iterations, the supply voltage based on $\alpha = 1.2$ is calculated. The obtained supply voltage 352 mV is in the range of 400 mV which is coupled with $\alpha = 2$ according to Table 3.3. Therefore, the acceptable design solution is the one based on $\alpha = 2$, since α is a function of V_{dd}; this fact has been ignored conventionally in modeling and was first highlighted in [30], also discussed in the first section. In this case, the practical solution will be the one based on $\alpha = 2$. For the purpose of s_1, since the expected operating state supply voltage is 400 mV, we only perform the design exploration based on $\alpha = 2$. Since T_{s-f} and T_{s-L} are greater than$T_{s-limit}$, the minimum permitted value of 400 mV could be assigned to; assuming that smaller values are accepted (regardless of DC–DC converter capabilities, leakage issues, etc.), the obtained values could be also applied to V_{dd}.

According to Table 3.6, the obtained solutions based on T_{s-L} and $\alpha = 2$, meet the target operating frequency and latency performance objectives at both s_0 and s_1. Whereas, controlling the buses based on T_{s-f} or other values of results into cases where the frequency or latency objectives are not met in s_0 or s_1 (performance violations or suboptimal results are underscored in the table). According to the described methods, at $s_1$more than four times the energy is saved compared to typical uniform DVS in power-managed systems, assuming the buses operate at the same supply voltage as logic. This efficiency is 17 % at s_1. Also note that, all the above reported results are obtained under worst-case design conditions.

3.7 Conclusions

Deep submicron (DSM) multiprocessor system on a chip (MPSoCs) technologies have become interconnect-centric, while minimizing their energy consumption has become a major design concern. In such advanced technologies, system energy is regulated by means of varying the system components operating states during run-time.

In this chapter, a flow comprising *intelligent* methods for modeling, design, and control (management) of interconnections in power-managed VLSI systems, was described. The described methods guarantee that the designed and controlled interconnects have minimum energy requirements while they meet their performance objectives at all the desired operating states.

The *intelligent* methods presented in this chapter could help designing and controlling energy-optimal interconnections, interconnect-centric components and/or mechanisms, particularly synchronization and communication mechanisms, etc., that need to meet desired performance objectives across various power-managed platform technologies. To name a few: microsystems enabled with run-time power management, multiple voltage domains (voltage islands) and multiple clock domains (MCD) systems, to name a few.

References

1. Martin, G.: Overview of the MPSoC design challenge. In: Proceedings of the 43rd Annual Design Automation Conference, July 2006
2. Horowitz, M., Indermaur, T., Gonzalez, R.: Low-power digital design. In: Symposium on low power electronics (1994)
3. Benini, L., Micheli, G.D.: Dynamic Power Management: Design Techniques and CAD Tools. Kluwer Academic Publishers, Massachusetts (1998)
4. Burd, T.D., Brodersen, R.W.: Design issues for dynamic voltage scaling. In: Proceedings of the 2000 International Symposium on Low Power Electronics and Design, July 2000
5. Benini, L., Bogliolo, A., De Micheli, G.: A survey of design techniques for system-level dynamic power management. IEEE Trans. Very Large Scale Integr. Syst. **8**(3), 299–316 (2000)
6. Pontes, J., Moreira, M., Soares, R., Calazans, N.: Hermes-GLP: a GALS network on chip router with power control techniques. In: Proceedings of the International Symposium on VLSI (ISVLI) (2008)
7. Semeraro, G., Magklis, G., Balasubramonian, R., Albonesi, D.H., Dwarkadas, S., Scott, M.L.: Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In: HPCA (2002)
8. Chen, G., Li, F., Kandemir, M., Irwin, M.: Reducing NoC energy consumption through compiler-directed channel voltage scaling. SIGPLAN Not. 41, 6, June (2006)
9. Li, F., Chen, G., Kandemir, M., Kolcu, I.: Profile-driven energy reduction in network-on-chips. SIGPLAN Not. 42, 6, June (2007)
10. Borkar, S.: Design challenges of technology scaling. IEEE Micro **19**(4), 23–29 (1999)
11. Zarkesh-Ha, P., Davis, J.A., Meindl, J.D.: Prediction of net-length distribution for global interconnects in a heterogeneous system-on-a-chip. IEEE Trans. Very Large Scale Integr. Syst. **8**, 649–659 (2000)
12. Magen, N., Kolodny, A., Weiserm, U., Shamir, N.: Interconnect-power dissipation in a microprocessor. In: Proceedings of the International Workshop on System Level Interconnect Prediction, February (2004)
13. Lackey, D.E., Zuchowski, P.S., Bednar, T.R., Stout, D.W., Gould, S.W., Cohn, J.M.: Managing power and performance for system-on-chip designs using Voltage Islands. In: ICCAD (2002)
14. Semeraro, G., Albonesi, D.H., Dropsho, S.G., Magklis, G., Dwarkadas, S., Scott, M.L.: Dynamic frequency and voltage control for a multiple clock domain microarchitecture. ACM/IEEE International Symposium on Microarchitecture (2003)
15. Magklis, G., Scott, M.L., Semeraro, G., Albonesi, D.H., Dropsho, S.: Profile-based dynamic voltage and frequency scaling for a multiple clock domain microprocessor. In: ISCA (2003)
16. Bakoglu, H.B., Meindl, J.D.: Optimal interconnection circuits for VLSI. IEEE Trans. Electron Devices **ED-32**, 903–909 (1985)
17. Nalamalpu, A., Burleson, W.: A practical approach to DSM repeater insertion: Satisfying delay constraints while minimizing area and power. In: IEEE ASIC/SOC Conference, September (2001)
18. Chen, G., Friedman, E.G.: Low-power repeaters driving RC and RLC interconnects with delay and bandwidth constraints. In: IEEE TVLSI, February (2006)
19. Kaul, H., Sylvester, D., Blaauw, D., Mudge, T., Austin, T.: DVS for on-chip bus designs based on timing error correction. DATE, March (2005)

20. Banerjee, K., Mehrotra, A.: A power-optimal repeater insertion methodology for global interconnects in nanometer designs. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **49**(11), 2001–2007 (2002)
21. Nalamalpu, A., Srinivasan, S., Burleson, W.P.: Boosters for driving long onchip interconnects-design issues, interconnect synthesis, and comparison with repeaters. IEEE Trans Comput.-Aided Des. **21**(1), 50–62 (2002)
22. Alpert, C., Devgan, A., Quay, S.: Buffer insertion for noise and delay optimization. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **18**(11), 1633–1645 (1999)
23. van Ginneken, L.P.P.P.: Buffer placement in distributed RC-tree networks for minimal elmore delay. In: Proceedings of the International Symposium Circuits System (ISCAS), pp. 865–868 (1990)
24. Adler, V., Friedman, E.G.: Repeater design to reduce delay and power in resistive interconnect. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **45**(5), 607–616 (1998)
25. Ismail, Y.I., Friedman, E.G.: Effects of inductance on the propagation delay and repeater insertion in VLSI circuits. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **8**(2), 195–206 (2000)
26. Akl, C.J., Bayoumi, M.A.: Reducing interconnect delay uncertainty via hybrid polarity repeater insertion. IEEE Trans. Very Large Scale Integr. Syst. **16**(9), 1230–1239 (2008)
27. Zhai, B., Blaauw, D., Sylvester, D., Flautner, K.: The limit of dynamic voltage scaling and insomniac dynamic voltage scaling. Trans. VLSI **13**(11), 1239–1252 (2005)
28. Kursun, V., Friedman, E.G.: Multi-Voltage CMOS Circuit Design. Wiley, West Sussex (2006)
29. Sinha, A., Chandrakasan, A.P.: Dynamic power management in wireless sensor networks. IEEE Des. Test **18**(2), 62–74 (2001)
30. Zarrabi, H., Al-Khalili, A.J., Savaria, Y.: An interconnect-aware delay model for dynamic voltage scaling in nm technologies. In: GLSVLSI (2009)
31. Zarrabi, H., Al-Khalili, A.J., Savaria, Y.: Repeater design for power-managed VLSI. Submitted to GLSVLSI (2011)
32. Zarrabi, H., Al-Khalili, A.J., Savaria, Y.: An interconnect-aware dynamic voltage scaling scheme for DSM VLSI. In: ISCAS, May 2010
33. Sakurai, T.: Approximation of wiring delay in MOS-FET LSI. IEEE J. Solid-State Circuits **SC-18**(4), 418–426 (1983)
34. Rabaey, J., Chandrakasan, A., Nikolic, B.: Digital Integrated Circuits: A Design Perspective, 2nd edn. Prentice Hall, Upper Saddle River (2003)
35. Elmore, W.C.: The transient response of damped linear networks with particular regard to wide-band amplifiers. J. Appl. Phys. **19**(1), 55–63 (1948)
36. Cong, J., Pan, D.Z.: Wire width planning for interconnect performance optimization. IEEE Trans. Comput. Aided Des. Integr. Circuits **49**(11), 1671–1677 (2002)
37. Zarrabi, H., Saaied, H., Al-Khalili, A.J., Savaria, Y.: Zero skew differential clock distribution network. In: International Symposium on Circuit and Systems (ISCAS), May 2006
38. Zhang, J., Friedman, E.G.: Decoupling technique and crosstalk analysis of coupled RLC interconnects. In: Proceedings of the IEEE International Symposium on Circuits and Systems II, pp. 521–524 , May 2004
39. Kahng, A.B., Muddu, S., Sarto, E.: On switch factor based analysis of coupled RC interconnects. In: Proceedings of the Design Automation Conference, June 2000
40. Pileggi, L.: Coping with RC(L) interconnect design headaches. IEEE ICCAD Tutorial, November 1995
41. Yee, G., Chandra, R., Ganesan, V., Sechen, C.: Wire delay in the presence of crosstalk. ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, December 1997
42. Duan, C., LaMeres, B.J., Khatri, S.: On and Off-chip Cross-talk Avoidance in VLSI Designs. Springer, New York (2010)
43. Zhao, W., Cao, Y.: New generation of predictive technology model for sub-45nm design exploration. IEEE International Symposium on Quality Electronic Design (ISQED) (2006)
44. International Technology Roadmap for Semiconductors: http://www.itrs.net

45. Berkeley Predictive Technology Model. http://www-device.eecs.berkeley.edu/ptm
46. Li, X.C., Mao, J.F., Huang, H.F., Liu, Y.: Global interconnect width and spacing optimization for latency, bandwidth and power dissipation. IEEE Trans. Electron. Devices **52**, 2272–2279 (2005)
47. Wang, A., Calhoun, B.H., Chandrakasan, A.P.: Sub-Threshold Design for Ultra Low-Power Systems. Springer, New York (2006)
48. Wang, A., Chandrakasan, A.P.: A 180-mV subthreshold FFT processor using a minimum energy design methodology. IEEE J. Solid-State Circuits **40**(1), 310–319 (2005)
49. Calhoun, H.B., Ryan, J., Khanna, S., Putic, M., Lach, J.: Flexible circuits and architectures for ultra low power. Proc. IEEE **98**(2), 267–282 (2010)
50. Chandrakasan, A.P., Daly, D.C., Finchelstein, D.F., Kwong, J., Ramadass, Y.K., Sinangil, M.E., Sze, V., Verma, N.: Technologies for ultradynamic voltage scaling. Proc. IEEE **98**(2), 191–214 (2010)
51. Forestier, A., Stan, M.R.: Limits to voltage scaling from the low power perspective. In: Symposium on Integrated Circuits and Systems Design, September 2000
52. Macchiarulo, L., Macii, E., Poncino, M.: Low-Energy encoding for deep-submicron address buses. In: ISLPED (2001)
53. Kalyan, T.V., Mutyam, M., Rao, P.V.: Exploiting Variable Cycle Transmission for Energy-Efficient On-Chip Interconnect Design. In: International Conference on VLSI Design (2008)
54. Ghoneima, M., Ismail, Y., Khellah, M.M., Tschanz, J., De, V.: Serial-link bus: a low-power on-chip bus architecture. Trans. Circuits Sys. Part I (2009)
55. Sotiriadis,, P.P., Chandrakasan, A.P.: A bus energy model for deep submicron technology, IEEE Trans. Very Large Scale Integr. Syst., June 2002
56. Sotiriadis, P., Chandrakasan, A.P.: Bus Energy Reduction by Transition Pattern Coding Using a Detailed Deep Submicrometer Bus Model, IEEE Trans. Circuits Syst., pp. 1280–1295, October 2003
57. Deogun, H., Rao, R.M., Sylvester, D., Brown, R., Nowka, K.: Dynamically pulsed MTCMOS with bus encoding for total power and crosstalk minimization. In: IEEE International Symposium on Quality Electronic Design, pp. 88–93 (2005)
58. Benini, L., De Micheli, G., Macii, E., Sciuto, D., Silvano, S.: Address bus encoding techniques for system-level power optimization, In: Proceedings of the Conference on Design, Automation and Test in Europe, February 1998
59. Suresh, D.C., Agrawal, B., Yang, J., Najjar, W.A.: Tunable and energy efficient bus encoding techniques, IEEE Trans. Comput. (2009)

Chapter 4
Introduction to Optimization Under Uncertainty Techniques for High-Performance Multicore Embedded Systems Compilation

Oana Stan and Renaud Sirdey

Abstract The compilation process design for massively parallel multicore-embedded architectures requires solving a number of difficult optimization problems, nowadays solved mainly using deterministic approaches. However, one of the main characteristics of these systems is the presence of uncertain data, such as the execution times of the tasks. The authors consider that the embedded systems design is one of the major domains for which applying optimization under uncertainty is legitimate and highly beneficial. This chapter introduces the most suitable techniques from the field of optimization under uncertainty for the design of compilation chains and for the resolution of the associated optimization problems.

4.1 Introduction

At the beginning of the twenty-first century, it has become obvious that the performances of single core architectures reached a plateau, the main reasons being the limits of instruction-level parallelism (ILP) as well as the heat wall for the frequency [39].

Between the only viable solutions left to improve performance was to make use of additional high-level parallelism, i.e., multiply the number of processing elements per chip. As a consequence, the standard mainstream and embedded architectures include, nowadays, at least four generic cores and we are entering into a multicore era in which the updated Moore's law states that *the number of cores doubles every two years*.

Figure 4.1 [53] captures the general exponential evolution trend of the number of individual computing units according to the release years of chips (heterogeneous

O. Stan (✉) · R. Sirdey
Embedded Real Time Systems Laboratory, CEA, LIST,
Point Courrier 172, 91191 Gif-Sur-Yvette, France
e-mail: oana.stan@cea.fr

R. Sirdey
e-mail: renaud.sirdey@cea.fr

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_4

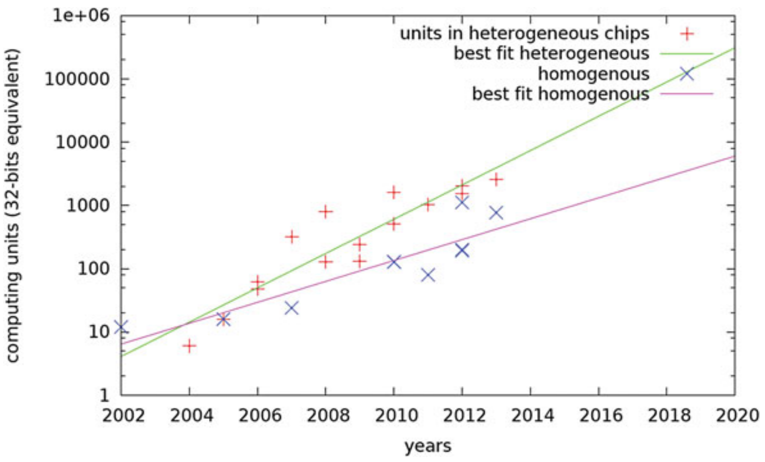


Fig. 4.1 Number of individual processing units in heterogeneous chips (e.g., AMD, NVidia graphics, IBM Cell BE, etc.) and homogeneous chips (e.g., Intel Xeon, IMB Power, STM, Tilera, Kalray, etc.) [53]

and homogeneous). If the generalization of multicore continues according to this trend, at the end of the decade, we will reach at least a thousand of generic cores.

However, according to Gustafson's law [37], as more computing power is available, new and more powerful applications make their appearance in order to benefit from the available capabilities. Already, the latest embedded applications for video and image processing (using complex compression and decompression algorithms), video games, scientific computing, or data security demand a computer power ten to hundred times superior to that of a few years ago.

Therefore, careful attention has to be paid when designing embedded systems solutions since programming applications that fully exploit the computing power and the parallelism is a difficult task.

As we will see further, the design for efficiently embedded manycore systems requires new programming and execution paradigms as well as innovative compilation technologies. One of the common practices is to make use of operation research techniques for the different optimization steps during the compilation process of parallel applications. Since between the main characteristics of the related optimization problems is the presence of intrinsic uncertain parameters, we believe that the overall compilation chain should integrate the latest advances from the optimization field such as stochastic programming methods and robust optimization algorithms.

Thus, this chapter is dedicated to the study of uncertainties associated with the embedded domain and to the analyzis of the most appropriate techniques for optimizing under these uncertainties when designing compilers for parallel embedded applications. The remainder of this chapter is organized as follows: after a short description of manycore architectures in Sect. 4.2.1, we present the existing approaches for programming parallel applications with a particular emphasis on dataflow-based languages (Sect. 4.2.2). The context and motivation for our study are getting refined in Sect. 4.2.3. Afterward, in Sect. 4.3, we describe the main sources of uncertainty affecting the properties of an embedded environment with a particular focus on execution times. In function of this analysis and taking into account the current state of art from stochastic and robust optimization fields, we give more details in Sect. 4.4 about some of the most relevant models and optimization techniques for the compilation of embedded systems. Also, in the last section, we show how these resolution methods can be applied in an operational manner for concrete application case studies, such as the partitioning, placement, and routing of network processes or the dimensioning of communication buffers.

4.2 Massively Parallel Embedded Systems

According to Flynn's macroscopic classification of computing systems, realized in function of the possible interaction patterns between instructions and data, there are four different theoretical classes of machines: Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD), and Multiple Instruction Multiple Data (MIMD). Between these categories, only the last three make parallel execution possible, and thus almost all parallel systems today are either SIMDs, easier to program but for which the parallelism is more difficult to exploit, or MIMDs, for which each processor is executing its own program flow. More flexible than SIMD and allowing nonstructured data and conditional or data-dependent algorithms, the MIMD is a more usual implementation of the multi and manycore concept.

Also, according to the memory organization, we can distinguish between DMM (Distributed Memory Machines) and SMM (Shared Memory Machines) systems.

The massively multicore (manycore) type of architecture is a compromise between the DMM and the SMM solving the problem of scalability of the SMMs for which it is difficult to exceed 32 processors and the performance issues of the DMMs.

Let us now give an overview of the main components of a massively multicore architecture, taking as an example the MPPA chip [30].

4.2.1 Manycore Architectures: MPPA Example

A massively multicore (manycore) processor is a parallel computing system, composed of a number of processing cores (at least a dozen), a mix of local and shared memory, distributed global memory or multilevel cache hierarchy, and an infrastructure for intercores communication.

Some examples of such embedded multicore architectures already available nowadays are [3, 10, 30, 42].

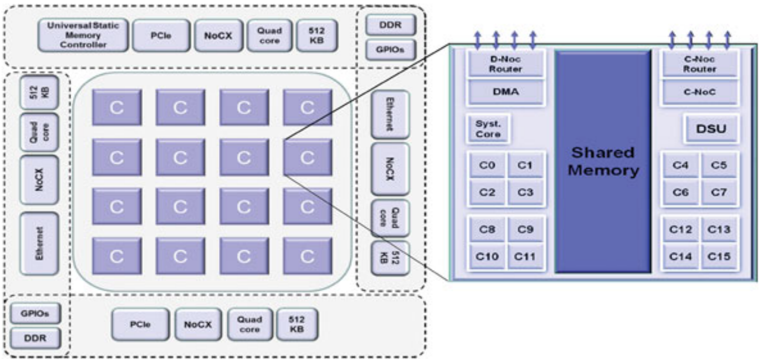


Fig. 4.2 Overview of Kalray's MPPA architecture [30]

The Kalray's MPPA-256 is one of the first homogeneous embedded manycore, released in 2013 and manufactured using 28 nm CMOS technology. As shown in Fig. 4.2, this single-chip manycore processor is organized as 16 (4 × 4) computing clusters and 4 additionally I/O clusters situated at the periphery and providing access to PCi interfaces, DRAM memory, etc. As the basic processing unit of the MPPA chip, each computing cluster integrates 16 processing engines (PE) cores, one resource management (RM) core, a shared memory, and a direct memory access (DMA) engine for transferring data. The 16 computing clusters as well as the 4 I/O subsystems are connected through a bidirectional NoC (Network-on-Chip) with a 2D torus topology and a wormhole route encoding.

4.2.2 Programming for Manycores: Dataflow Oriented Languages

In order to take benefit of the underlying execution infrastructure, when programming parallel applications for manycore, one has to handle several difficulties: dispose of limited and dependent resources (memory, NoC), be able to run correctly large parallel programs and efficiently exploit the parallelism and the computing power.

There is a real urge nowadays for languages permitting to design efficiently and without difficulties parallel applications. As noticed several years ago, the usual imperative programming paradigms (C or Java like) are based on a sequential von Neumann architecture and thus they are inappropriate for writing effective parallel programs. Other modern programming languages like MPI, OpenMP and OpenCL used currently on distributed systems require explicitly managing communications and synchronizations between tasks.

Some paradigms such as agent-based and dataflow programming languages allow to overcome several of the above drawbacks, providing mechanisms to mask low-level communication and intertasks synchronization, assure execution determinism and easily integrate existing code.

In a dataflow model, a program is described as a directed graph, consisting of nodes representing tasks (also named agents or actors) and arcs representing unidirectional communications channels. The exchange of data, quantized into tokens, is realized exclusively through the communications channels and the execution of the program consists in a sequence of firings or evaluations (which correspond to the consumption/production of a certain number of tokens).

With the first models emerging in the early 1970s, there are different formalisms for dataflow-based languages (KPN—Kahn Process Networks [44], DPN—Data Process Networks [47], CSDF—Cyclo-Static Data Flows [15], etc.) different in their expressive power and the guarantees they provide.

In the following, we will focus on the CSDF model of computation and on ΣC [2], a more recent dataflow-based language. In a CSDF model, the number of produced/consumed tokens can vary from an activation to another in a cyclic manner. Since programmers do not have to worry about data synchronization and the computing tasks are only connected through well identified channels, this flexible dataflow model is well suited for efficiently scheduling and mapping applications on manycore platforms. ΣC, a C extension programming language based on CSDF, has been designed for allowing reusability of existing C code, while taking advantage of the properties of a dataflow models such as the ability to verify absence of deadlocks and memory bounded execution. Its ability to specify the production and consumption of each task is used at compile time for different checkings such as buffer sizing, placement, routing, parallelism refinement, etc.

Thus, once the application has been designed and implemented using a parallel programming language, it is the role of the compilation chain to make the connection with the specific execution model for the embedded manycore target. A general difference between a dataflow compiler and a standard one is the fact that the former is handling itself the underlying parallelism, easing the role of the designer.

Let us exemplify the compilation process through the ΣC compilation chain.

4.2.3 Compilation of Applications for Manycore Systems: ΣC Toolchain Example

The ΣC compilation toolchain adapts as best as possible the application code, generic, to the targeted architecture: number of cores, NoC topology, etc. As such, even if the language is platform independent, the compiler will automatically map the parallel program onto a large number of processors, using different architectures and communication types.

There are four passes in which ΣC compilation process is organized:

- **Lexical analysis, parsing and code generation**. This first pass, the ΣC front-end, begins with a lexical, syntactic, and semantic analysis of the code, common to most compilers. Afterward, preliminary C codes are generated from ΣC sources either for offline execution (the instantiation codes of the agents) or for further refinement.

- **Compilation of the parallelism**. The purpose of the second pass, the ΣC middle-end, is to instantiate and connect the agents, by executing at compile time the corresponding codes generated by the first pass.
  Once the construction of the application graph is complete, parallelism reduction techniques by pattern matching [21] are applied and a safe computation of a deadlock-free lowest bound for the buffers sizes of the links is also performed.
- **Resource allocation**. The third pass is in charge of resource allocation (in the larger sense). First, it supports a dimensioning of communication buffers taking into account the execution times of the tasks and the application requirements in terms of bandwidths (nonfunctional constraints). Next, in order to realize a connection with the execution model, it constructs a folded unbounded partial ordering of task occurrences (and thus, finitely representable).
  This pass is also responsible of placement and routing, with the objectives of grouping together (under capacity constraints for each cluster of the architecture) the tasks which communicate the most, mapping these groups of tasks to the clusters and finally, computing routing paths for the data traversing the NoC.
- **Runtime generation and link edition**. The last pass, the ΣC back-end, is responsible of generating the final C code and the runtime tables. Also, during this stage and using C back-end compiler tools, the link edition, and the loadbuild are realized.

4.2.4 Characteristics of Optimization Problems Associated to the Compilation Process for Manycore

As seen in the previous section, the compilation process of an application for a massively parallel architecture is becoming rather complex and requires solving several difficult optimization problems.

Nowadays, the compiler design implies the application of advanced operations research techniques not only to the so-called back-end (by solving optimization problems such as buffer sizing and instruction scheduling, e.g., [14, 38, 49]) but also more upward and all the long of the compilation process, in order to efficiently allocate and exploit the interrelated resources offered by parallel architectures. Between the more recent optimization problems, we can mention the placement/routing for multicores or the construction of a partial order under throughput constraints (e.g., [35, 36]).

Moreover, most of the existing studies treating optimization problems for embedded parallel architectures propose deterministic models and we noticed only a few studies which take into consideration parameter variations and apply the techniques of optimization under uncertainty to the embedded domain (e.g., [22, 48, 51]).

Still, one of characteristics of the manycore systems is the presence of intrinsic uncertain data occurring in the definition of these related optimization problems, such as execution times or latencies. As we will see further, experimental studies from both fields of operations research and program compilation have shown that considering a fixed value for the uncertain parameters, respectively, execution times (usually the mean value), can lead to wrong estimates and optimization solutions not always feasible. As such, developing and testing optimization techniques taking into account uncertainty for this field seem beneficial and even necessary.

In order to conceive and develop methods of optimization under uncertainty which are adequate to the domain of compilation for manycores, a first step consists in identifying, analyzing, and, if possible, modeling the sources of uncertainty specific to this area. As such, we proceed in the next section with a qualitative analysis of the uncertainty sources, with a particular emphasis on the execution times.

4.3 Characterization of the Uncertainties in the Context of Manycores

One of the main sources of uncertainties related to the domain of embedded systems lies in the intrinsic indeterminism of execution times for computing kernels of intermediate granularity.

4.3.1 Overview of the Execution Times

In fact, there are two main sources of uncertainty related to the execution times of embedded systems:

1. intrinsic dependency on the data. Since usually the computation code of an application depends on the input data, there are several treatments which could be executed by the application, translating into different data-dependent paths, with potentially different execution times.
2. extrinsic uncertainty due to architecture characteristics. Variations of execution times are also related to the speculative components (such as caches, pipelines, or branch prediction) of modern hardware architectures on which the application is executed.

These sources of uncertainty are not independent and one must take into account both execution paths and hardware mechanisms.

As described previously, we assume that the embedded application consists of a number of tasks or agents, which work together to achieve the required functionalities. In Fig. 4.3 [71] several relevant properties of the execution time for a task are revealed. The darker upper curve represents the set of all execution times. The shortest execution time is often called best-case execution time (BCET) and the longest is called worst-case execution time (WCET). The other envelope represents a subset of the measures of the execution times. The minimum and maximum of the lower curve

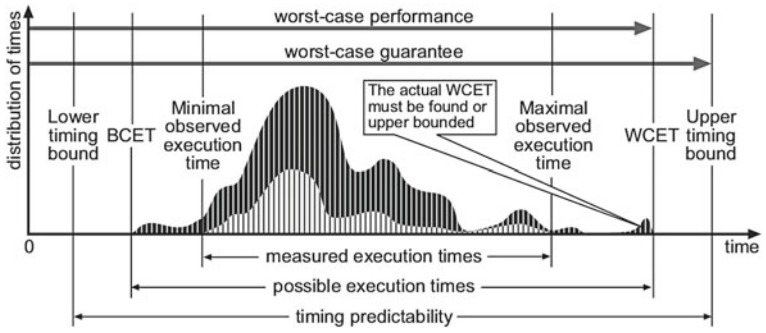


Fig. 4.3 Some properties of the execution times of a real-time task [71]

are the minimal and the maximal observed execution times, respectively. Since, in most cases, the space of all possible executions is too large to fully explore, and also because of the undecidability problem associated to the running of an arbitrary program, it is not possible to determine the exact worst and best case execution times.

Most researches dedicated to the timing analysis of execution times consist in deriving or computing upper bounds for the WCET. For a detailed overview and survey of methods and tools for estimating WCET, we refer the reader to [71].

4.3.2 Estimating Execution Times Distributions

While the methods for estimating bounds for execution times are getting more and more complex, by also taking into account the speculative behavior of the target architecture, they remain justified mainly for hard real-time systems. Instead, for soft real-time systems, there are more and more studies based on probabilistic analysis and approaches for scheduling (e.g., [17, 29, 55]) considering that the execution times of the tasks follow probability distributions.

The problem of estimating the execution times consists in predicting the execution time of a task on a variety of machines in function of the data set and with a high level of accuracy. The existing solutions to this problem can be divided into three main classes: code analysis [63], analytic profiling [33, 45, 73], and statistic prediction [28, 43].

An execution time estimate found by analysis of the source code of a task is typically limited to a specific class of architectures and a particular code type. Consequently, code analysis is not very adapted to treat heterogeneous computing. The profiling technique, first presented by Freund [33], determines the composition of a task in terms of primitive code types. Code profiling data is then combined with benchmark data (obtained on each machine and measuring the performance for each code type). The main disadvantage of this type of methods is that they cannot determine the variations in the input data set. The third category, the statistical prediction algorithms, makes predictions from previous observations. Each time a task executes on a machine, the execution time is measured and added to the set of past observations. The quality of estimation depends on the size of the set of observations. The advantage is that these methods can compensate for parameters of the input data and the machine type without any supplementary information about the internal code or the machine.

A recent work [56] is going further with the analysis, by studying the variations of execution times on multicore architectures. The experimental work is conducted on samples from two benchmarks; SPEC CPU, large sequential applications, and SPEC OMP2001 benchmarks, OpenMP applications. Each program is executed 30 times on a 8 cores Linux machine with the same training input data each time. The normality check (using the standard Shapiro-Wilk test) for both benchmarks proved that the distribution of execution times is not a Gaussian function in almost all cases. Also, contrary to sequential SPEC CPU applications, OpenMP applications have a more important variability of execution times. By executing 30 times, several applications from the SPEC OMP benchmark on different software configurations (sequential and multithreads), the study shows that if the sequential and single threaded versions do not have important variations, when using a larger thread level parallelism (more than 1 thread), the overall execution times decrease with a deeper disparity. More, the mean confidence intervals (obtained with Student's test) are not always tight.

4.3.3 Execution Times: A Qualitative Analysis and Basic Examples

Even if it is reasonable to assume, in embedded computing, that the execution times have probability distributions of bounded support (no infinite loops), we have to cope with the fact that these distributions are intrinsically multimodal.

Let us give some simple examples. For example, for the computing kernel in Table 4.1, there are two modes for the executions times, possible with different variances, corresponding to the two sequences of instructions (see Fig. 4.4).

Instead, for the code in Table 4.2 with n taking values between 1 and N, S_1 and S_2 being two linear sequences of instructions, the distribution has $2N$ modes (the figure Fig. 4.5 showing a possible envelope of the distribution for the case when $N = 4$).

Running a more complicated application like X264 encoder clearly shows that the distribution of execution times is difficult to model and that it is multimodal.

Table 4.1 Example 1—A simple code snippet

```
if condition then
    $S_1$
else
    $S_2$
end if
```

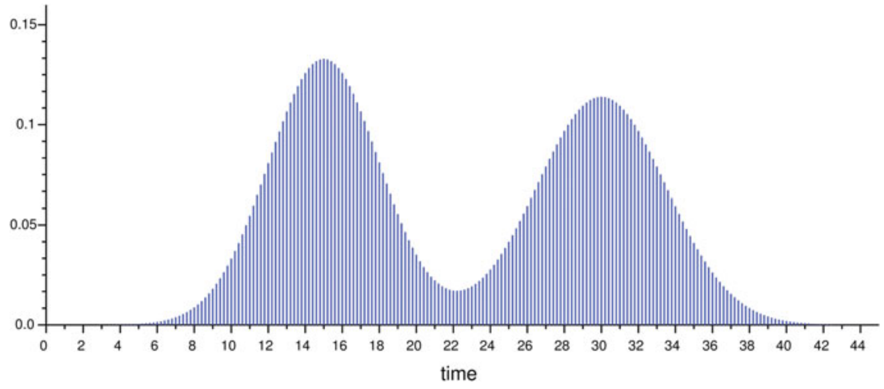


Fig. 4.4 Example 1—2 mode distribution for the execution times

Table 4.2 Example 2—Another code snippet

```
for $i = 1$ to $n$ do
    if condition then
        $S_1$
    else
        $S_2$
    end if
end for
```

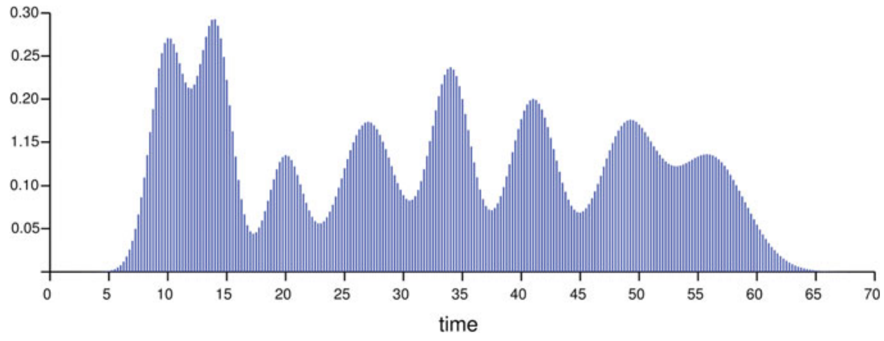


Fig. 4.5 Example 2—Multimodal distribution for the execution times

Figure 4.6 shows the envelope of executions times for each frame when the X264 is executed on a Linux machine, taking as input a video benchmark of size 1280×720, with 24 frames per second.

Hence, it is difficult to model these probabilities laws through usual distributions such as the normal or uniform ones, which are unimodal.

Furthermore, in the case of a process network, we cannot overlook the problem of dependency between these random variables. An easy example consists in the target tracking pipeline for which the execution times of each of the pipeline elementary

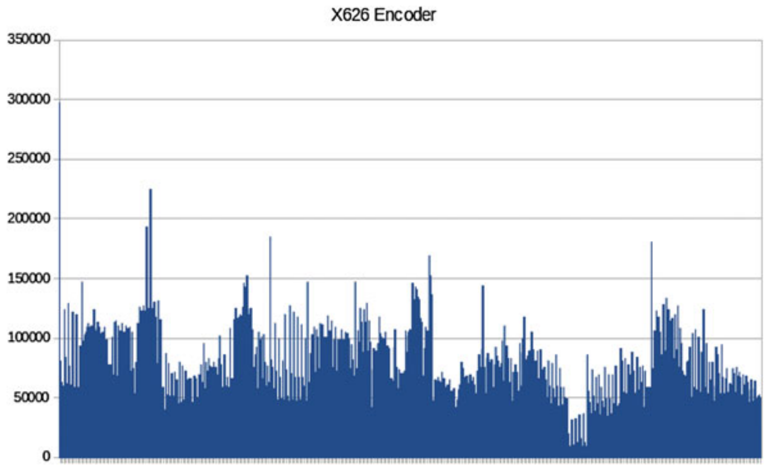


Fig. 4.6 Example 3—Envelope of execution times for frame treatment in a X264 encoder

tasks depend, to a certain degree, on the number of effectively treated targets. In Table 4.3, another example is presented consisting of two elementary tasks both depending on same input data d, difficult to characterized, and each task having two modes for its execution times. As such, as illustrated in Fig. 4.7, the execution time of task $T1$ is dependent to a certain degree of execution time of task $T2$.

To conclude, it is appropriate to assume that the execution times are random variables characterized by complicated multimodal joint distributions, presumably better defined as unions of orthotopes rather than a Gaussian or even a mixture of Gaussians.

However, modeling the underlying distribution of execution times seems delicate and thus, for solving the optimization problems related to compilation for high parallel systems, we do not encourage the use of parametric methods. The main raison is that these parametric optimization methods are making assumptions on the existence of a probability model for the uncertain parameters.

Let us now introduce some general principles about optimizing under uncertainty.

Table 4.3 Exemple 4—Code snippet showing possible tasks dependency

T_1	T_2
if $f(d)$ **then**	**if** $g(d)$ **then**
S_1	S_3
else	**else**
S_2	S_4
end if	**end if**

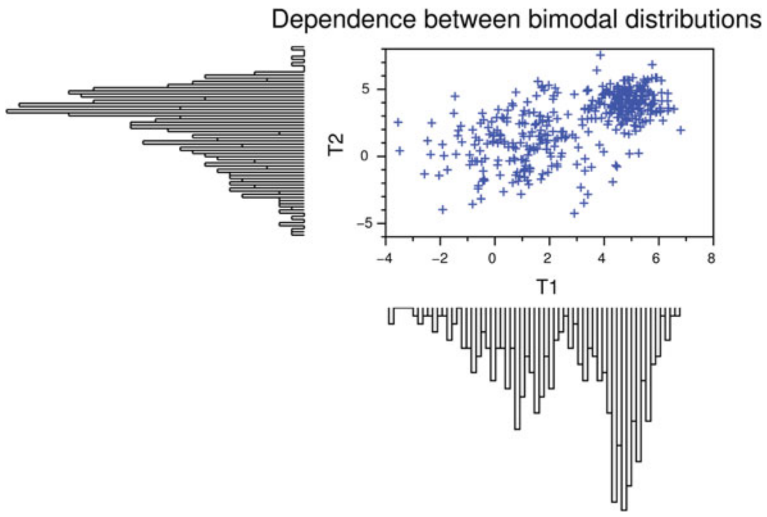


Fig. 4.7 Example 4—Execution times for T_1 and T_2

4.4 Optimization Under Uncertainty

Beginning with the seminal works of Dantzig [25], Charnes and Cooper [23], Miller and Wagner [57], Bellman and Zadeh [4], optimization under uncertainty remains one of the most active domains of research and thanks to recent studies allowing major advances, there is an increased regain of interest for this discipline.

4.4.1 Generalities

An example illustrating the importance of taking into account uncertainty for optimization problems is the recent case study of Ben-Tal and Nemirovski [8] on a collection of 90 problems from NETLIB library [70]. They showed that systems optimized in the classical deterministic sense can be very sensitive to small changes on the parameters values and that only 1 % perturbation of the data can severely affect the feasibility properties of the found solutions.

Therefore, for real-world optimization problems in which data are uncertain and inexact (as, for example, those related to compilation field), in order to find optimal solutions which are feasible in a meaningful sense, one has to deal with the randomness of the variables.

A crucial aspect being the way uncertainty is formalized, we can thus distinguish two major branches of optimization under uncertainty: stochastic programming and robust optimization.

In stochastic combinatorial optimization problems (SCOP), it is assumed that uncertain information can be described by random variables which can be characterized by probability distributions. Static SCOPs are *a priori* optimization problems where the decisions are taken and optimal solutions are found in the presence of randomness, at one single step, *before* the actual realization of the random variables. Dynamic SCOPs consider that decision cannot be made until random variables are revealed and associated random events have happened. As such, decisions are taken *after* random events occur in a single stage, in the case of *simple recourse* problems or in several stages, for *multistage recourse* problems. For both static and dynamic models, there are decisions and there are observations of the random variables, the order of succession being given by different schemes: for static models, first decision, then observation while for dynamic problems, at least one decision is preceded by at least one observation.

Robust optimization does not need to assume any exact knowledge about the probability distribution of random data; instead, uncertain information is set based. As such, uncertain parameters are characterized through a set of possible events and, usually, the decision-making process searches for solutions that are feasible for any realization of the uncertainty in the given set. Indeed, the main criticism of classical robust approaches (e.g., the so-called "max-min" or worst-case approach, the regret maxmin, etc.) is their over-conservatism since they are searching for solutions that are feasible for *all* possible events from the uncertainty set. As such, the obtained solutions are often too conservative, of large cost, being guaranteed even for events with a low probability to occur. Recent approaches (e.g., [7, 13]), more flexible, try to rectify this drawback, by making particular assumptions about the uncertainty set of the parameters and proposing deterministic counterparts to the original robust problem.

Even if the robust methods construct solutions which are immune to data uncertainty, in general, the quality of the solution is not assessed with probabilistic considerations. However, from our perspective, the probability to respect a given reliability target is a more intuitive notion, often easier to set for a decision maker. Additionally, we consider that the problem formulations accompanied by probability guarantees are more appropriate when applying optimizations for the domain we target, the compilation for manycore.

As such, in the following, we are concentrating on the resolution methods for static stochastic programs and (without loss of generality) with uncertainty affecting the constraints, aka *chance-constrained programs*.

4.4.2 Chance-Constrained Programming

The general form of the chance-constrained problem is the following:

$$\begin{aligned} \min_{x} \quad & g(x) \\ \text{s.t.} \quad & \mathbb{P}\left(G(x,\xi) \leq 0\right) \geq 1-\varepsilon \end{aligned} \tag{CCP}$$

where $x \in \mathbb{R}^n$ is the decision variable vector, $\xi \in \Omega \longrightarrow \mathbb{R}^D$ represents a random vector and $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function. We suppose that there exists a probability space $(\Omega, \Sigma, \mathbb{P})$, with Ω, the sample space, Σ, the set of events, i.e., subsets of Ω, and $\mathbb{P}$, the probability distribution on Σ. $G : \mathbb{R}^n \times \mathbb{R}^D \longrightarrow \mathbb{R}^m$ is the function for the m constraints, $0 \leq \varepsilon \leq 1$ is a scalar defining a prescribed probability level and $\mathbb{P}(e)$ of an event e is the probability measure on the set Σ.

This type of problem, where all constraints should be satisfied simultaneously with a probability level of at least $1 - \varepsilon$, is known in the literature as a *joint chance constrained program*. Another variant of optimization problems with uncertainty affecting the constraints is the *separate chance constrained program* in which different probability levels ε_i can be assigned to different constraints. In separate chance constraints the reliability is required for each individual feasible region while in joint chance constraints the reliability is assured on the whole feasible space. Even if appealing for their more simple structure, the separate chance-constrained programs have the important drawback of not properly characterizing safety requirements [62]. As such, while separate chance constraints could be used in the case when some constraints are more critical than others, joint chance constraints seems a more appropriate choice for guaranteeing an overall reliability target for the system.

As one may expect, chance-constrained optimization problems are inherently difficult to address and although this class of problems have been studied for the last 50 years, there is still a lot of work to be made towards practical resolution methods. There is not a general method of resolution for chance-constrained programs, the choice of the algorithm depending on the way random and decision variables interact.

Basically, the major difficulties associated to joint CCP are related to the convexity of chance constraints and the evaluation of the probabilistic constraints. For optimization problems, the convexity is a structural property allowing to use resolution techniques from convex optimization field and thus, finding a global optimal solution. Or, the distribution function of random variables is not in general completely concave or convex. Worse, even if each constraint is convex, the union of all of them may not be convex. As for the evaluation, for a given x, of the probability that $G(x, \xi) \leq 0$, one has to know the probability distribution of the random vector ξ. So, a first problem raises, the one of the modeling random data in practical applications when the involved distributions are not always known exactly and have to be estimated from historical data. The second problem is numerical since typically ξ is multidimensional and there are no ways to compute exactly and efficiently the corresponding probabilities with high accuracy. (At best, if given, the multivariate distribution of ξ, can be approximated by Monte-Carlo simulations or bounding arguments.)

As such, even for simple cases (e.g., Gaussian distributions for random variables), chance-constrained programs can be very difficult to solve. Table 4.4 shows some of the main theoretical and algorithmic resolution methods proposed for solving joint

Table 4.4 Methods for solving chance-constrained programs

Category	Characteristics	Some references
Convexity studies	Theoretical approaches	[23, 57]
	Particular assumptions on the distribution	[40, 62]
Robust optimization	Relatively simple to apply	[5–7, 13, 18–20, 24, 34, 46, 72]
Approximations and sampling	Compute bounds and approximate solutions	[58, 60, 64]
	Usually computationally demanding	[9, 26, 27, 41, 50, 54, 61]
(Meta) Heuristics	Use of precedent techniques for computing distribution	[1, 11, 12, 52, 69]

CCP. Along with the general hypotheses made for each category (e.g., random data in the right-hand side, normal distribution, etc.) (see column "Characteristics") a list of references is provided (in column "Some references").

Concerning the first category, to the best of our knowledge, existing studies determined convexity conditions only for linear probabilistic constraints with normal distributions in left-hand side or log-concave distributions on the right-hand side.

As for the robust approaches proposing probabilistic guarantees, they also need to make particular assumptions, usually quite mild, and they are applicable for specific classes of problems (e.g., linear programs) for an exact resolution.

Other directions of research consist either in discretization and sampling the distribution or in developing convex approximations. Usually, the proposed approximations find feasible but suboptimal and too conservative solutions to the original problem without any guarantees on their quality. The approximation methods based on sampling are replacing the actual distribution by an empirical distribution estimated by simulation when a direct evaluation of the feasibility of chance constraints is not possible and the probability has no available closed form. However, the use of Monte-Carlo simulations is too computationally demanding when ε is small and the assumptions made are restricting their applicability to particular cases (e.g., in order to generate Monte-Carlo samples, these methods require the full joint distribution).

Finally, there are a few approaches that propose heuristics, type genetic algorithms, or tabu search, combined with simulation techniques, in order to propose approximate solutions to chance-constrained programs.

We will not go further on in details concerning each class of methods and we refer the interested reader to the articles mentioned in Table 4.4. Instead, in the next section, we will concentrate on a selection of resolution approaches for chance-constrained programs, the most appropriate (from our perspective) for the field of compilation for high parallel embedded systems.

4.5 Some Suitable Methods of Optimization Under Uncertainty for Compilation of High Parallel Embedded Systems

One of the key aspects when choosing an optimization algorithm consists in analyzing the specificities of the parameters of the problem and of the involved area. Or, most of the previously mentioned approaches for optimizing under uncertainty are making assumptions (e.g., existing analytical form of the distribution, independence of the random vector components) which are either restrictive, or difficult to verify, or not always adequate to represent the uncertainty of real-life applications. Also, most of the approaches for optimization under uncertainty, based on probability models, make assumptions on the underlying distribution or use simulations without making a true connection with the data (i.e., without a through model validation with the available experimental data samples).

However, as illustrated before, the main sources of uncertainty for the compilation of dataflow application on manycore, the execution times, are random variables difficult to fully describe analytically and for which it is difficult to assume a “nice” probability model.

Another aspect to take into account when conceiving optimization methods for dimensioning embedded applications is their response-time requirements.

For safety-critical applications (hard real-time systems), like nuclear power plant control or flight management systems, all the timing constraints have to be met which often goes along with a worst-case approach. Even if they lead to an oversizing of the systems, worst-case approaches [65] are favored since missing any deadline is highly risky and unacceptable.

The methodologies we present in the next section are more suitable for the dimensioning of *soft real-time systems*, such as multimedia applications (video encoding, virtual reality, etc.) for which missing a deadline from time to time is acceptable, resulting only in a decrease of the quality of service. In fact, almost all of the probability-based studies related to real-time systems are intended for this kind of systems. Thus, even if the dimensioning is no longer guaranteed in all cases, acceptable deviations are admitted, and, in consequence, it is possible to avoid oversizing (expensive in terms of hardware) or undersizing (expensive in terms of reliability). Moreover, for a system already dimensioned, it is possible to estimate the level of robustness and specify deviation scenarios for which the system is no feasible or scenarios which could be acceptable.

As such, in the following, we describe two methodologies for optimization under uncertainty adapted for the compilation of soft real-time applications. Since the choice of a proper probability model for the execution times seems difficult, these methods are nonparametric with almost none or only a few assumptions on the distributions of the uncertain data.

4.5.1 The Robust Binomial Approach

The robust binomial approach (RBA) [68] is a simple and pragmatic method using statistical hypothesis testing theory and directly exploiting an available sample, with almost no assumption on the uncertain data. Since it is a specialization of the sample-based approaches, which have the drawback to have a high computational complexity, this generic method is intended to be used within an heuristic algorithm. Moreover, it leads to a generic solution to leverage existing heuristic algorithms for the deterministic case to their chance-constrained counterparts to solve relatively *large size* optimization problems.

4.5.1.1 Basic Ideas and Motivation

In the framework of an iterative compilation, we have at our disposal representative experimental temporal data, obtained during system validation, for example, when performing tests on the target architecture. These observations can be directly employed in order to construct an equivalent optimization problem, more robust and compatible with the variations of the real data, with the condition that the available sample is sufficiently representative of the entire distribution.[1]

The idea is to take advantage of the existing experimental data and revisit the scenario approach in order to provide probabilistic guarantees. The general scenario approach [19, 20] is between the only tractable convex approximations of a chance-constrained program, which does not impose any restrictions on the structure of the uncertain data (in particular with respect to the random vector component independence). Given a sample $\xi^{(1)}, \ldots, \xi^{(NS)}$ of size NS of independent and identically distributed observations of the random vector ξ, the scenario approach in its original form searches a solution satisfying all the realizations and therefore it finds suboptimal solutions, too conservative.

4.5.1.2 Statistical Hypothesis Testing

Before presenting the statistical results on which the robust binomial approach is based, let us introduce the following notation:

[1]An assumption that can be in practice checked, to some extent, using static program analysis techniques. An assumption which also relies reasonably on the expertise of test engineers in terms of designing validation cases representative of real-world system operations.

x	decision vector
ξ	uncertainty vector
p_0	$\mathbb{P}(G(x,\xi) \le 0)$
$\xi^{(1)},\ldots,\xi^{(NS)}$	i.i.d. random variables corresponding to NS observations of ξ
$\tilde{\xi}^{(i)}$	realization of observation $\xi^{(i)}$
χ_i	Bernoulli variable equal to 1 if $G\left(x,\xi^{(i)}\right) \le 0$ and 0 otherwise.

The random variable $\chi = \sum_{i=1}^{NS} \chi_i$ follows, *by definition*, a Binomial distribution with parameters NS and p_0 ($\chi \sim \mathscr{B}(NS, p_0)$). Let us now consider a realization $\tilde{\chi}$ of χ. If $\tilde{\chi}$ (corresponding to the number of times the inequality $G(x,\xi) \le 0$ is satisfied on a sample of size NS) is sufficiently large (for instance, larger than $k(NS, 1-\varepsilon, \alpha)$) we say that the constraint $\mathbb{P}(G(x,\xi) \le 0) \ge 1-\varepsilon$ is *statistically satisfied.*

The value of the threshold $k(NS, 1-\varepsilon, \alpha)$ (to which, for simplicity sake, we will refer, from now on, as k) will be chosen so that the probability we accept the constraint by error is smaller than a fixed α, in which case p_0 is strictly smaller than $1-\varepsilon$:

$$\mathbb{P}(\chi \ge k) \le \alpha \tag{1}$$

For any fixed $p_0 < 1-\varepsilon$, $\mathbb{P}(\chi \ge k)$ is smaller than $\mathbb{P}(\chi' \ge k)$ when $\chi' \sim \mathscr{B}(NS, 1-\varepsilon)$. So we can choose k such that $\mathbb{P}(\chi' \ge k) \le \alpha$.

Given x and ε, the parameter α can be interpreted as the type I error of the statistical hypothesis test with the following composite hypothesis:

$$\begin{cases} \mathrm{H}_0 : \mathbb{P}\left(G(x,\xi) \le 0\right) < 1-\varepsilon \\ \mathrm{H}_1 : \mathbb{P}\left(G(x,\xi) \le 0\right) \ge 1-\varepsilon \end{cases}$$

H_0 corresponds to the null hypothesis made by caution, which is (intuitively) the hypothesis we wish to reject only if we have statistically significant reasons to do so.

Hence, we can conclude, with a high *confidence level* of at least $1-\alpha$, that $p_0 \ge 1-\varepsilon$.

4.5.1.3 Sensitivity Analysis of the Parameters in RBA

Table 4.5 shows some minimal values for k in function of the sample size NS, of $\varepsilon = 0.10$ and of $\alpha = 0.05$. For example, for establishing that an inequality holds with a preset probability level of $1-\varepsilon = 0.90$ and with a confidence level of $1-\alpha = 0.95$, for a sample of size 50, the threshold k needed is at 48 and $P(\chi \ge 48) \approx 0.034$. It should also be noted that, for a practical use, the parameters ε and α should be of the same order of magnitude.

We can also establish in advance the minimal size of the sample required for a fixed level of the probability $1-\varepsilon$ (with $\varepsilon \in\ ]0,1[$) and a prespecified confidence level $1-\alpha$ (with $\alpha \in\ ]0,1[$).

Table 4.5 Examples values for $k(NS, 0.90, 0.05)$ in function of NS

NS	$k(NS, 0.90, 0.05)$
30	29
40	38
50	48
100	94
1000	915
10,000	9528

In particular, if $p_0 = 1 - \varepsilon$ and $\mathbb{P}(\chi = NS) > \alpha$ then we can affirm that the sampling size is insufficient (which is true for $NS = 10$ and $NS = 20$). This formula provides an easy way to determine the minimal number of realizations that need to be drawn in order to statistically significantly (α) achieve the desired probability level $(1 - \varepsilon)$.

A further analysis consists in studying the effect of variations of $1-\alpha$ and of ε on the threshold k. For an acceptable risk error α (less than 10 %) and a fixed probability level $1 - \varepsilon$, the variation of k in function of α does not look so important (in average a difference of 7 additional realizations for respecting the constrains and accept a smaller risk of 0.01 instead of 0.1 for a sample size of 1000). Instead, the value of the initial reliability level $1 - \varepsilon$ has a greater impact on the threshold k for same sample size. As expected, for an important probability guarantee, the number of realizations satisfying the constraints has to be higher. For example, for a sample of size 1000 and different levels of $1 - \alpha$, we remark an augmentation in the number of realizations to hold the constraints (i.e., the value of k) of 85, in average, for $\varepsilon = 0.01$ than for $\varepsilon = 0.1$.

4.5.1.4 Chance Constraints and Sampling

The statistical theory explained above can be applied for obtaining a statistically significant approximation model to the initial program (CCP). In order to obtain a relevant equivalent program to (CCP) model, the following assumptions about the random vector ξ represented by a sample of size NS are made:

Assumption 1 NS, the size of the sample for the uncertain vector ξ, is finite and sufficiently representative.

Assumption 2 The sample for ξ is composed of independent and identically distributed (i.i.d.) observations: $\xi^{(1)}, \ldots, \xi^{(NS)}$.

The first assumption is not very restrictive, since even if the number of initial observations is not sufficient, we can resort to statistical methods for resampling, such as bootstrapping [31]. However, it is important that the initial sample is representative of the distribution. The second assumption, of independence, is on the different observations of the random vector and not on its components which (as already stated) can be dependent. Additionally, the assumptions above remain quite general. As many previous studies do not mention, these assumptions are also necessary in the case of methods using a probability model, as the model itself must be validated e.g., on a Kolmogorov–Smirnov hypothesis test *using an i.i.d. sample of experimental data.*

Let us now define the binary variable $\tilde{\chi}_i$ for realization $\tilde{\xi}^i$:

$$\tilde{\chi}_i = \begin{cases} 1 & \text{if } G\left(x, \tilde{\xi}^{(i)}\right) \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Since the sum $\sum_{i=1}^{NS} \chi_i$ follows a Binomial distribution of parameters NS and p_0 (again, by construction), it is possible to determine $k(NS, 1-\varepsilon, \alpha)$. Therefore, $\tilde{\chi}_i$, the realization of the variables χ_i, can be used to replace the probability constraint

$$\mathbb{P}(G(x,\xi) \leq 0) \geq 1 - \varepsilon$$

and to obtain the (RBP) formulation, equivalent to (CCP):

$$\min_x \quad g(x) \tag{RBP}$$

$$\text{s.t.} \quad \sum_{i=1}^{NS} \tilde{\chi}_i \geq k(NS, 1-\varepsilon, \alpha) \tag{2}$$

$$G(x, \tilde{\xi}^{(i)}) \leq (1 - \tilde{\chi}_i)L; \qquad i = 1, \ldots, NS$$

$$\tilde{\chi}_i \in \{0, 1\}; \qquad i = 1, \ldots, NS$$

The first constraint assures that the number of constraints which are satisfied for the given sample are superior to the threshold k, fixed in advance in function of NS, ε and α. Constraints of type 2 verify the respect of the initial constraint for each realization i, making the link between x, $\tilde{\xi}^{(i)}$ and $\tilde{\chi}_i$, with L a constant of large size, depending on the problem structure but generally easy to find. For example, for a knapsack constraint $\sum_{i=1}^{m} w_i x_i \leq C$ with $w_i \geq 0$ the weights of the items to be placed, supposed uncertain, x_i binary variables and C the maximal capacity allowed, L is equal to $\sum_{i=1}^{m} w_i$.

Minimizing the objective function $g(x)$ for (RBP) model is equivalent to solving the initial program (CCP) with a confidence level of at least $1 - \alpha$. Additionally, we emphasize once more that the validity of this approximation is independent of any particular assumption on the joint distribution of the random vector ξ, in particular with respect to inter-component dependencies.

Although it is well illustrated on the mathematical formulation (RBP), it should be stressed out that RBA approach is not really appropriate in a mathematical programming setting, since, for example, the reformulation of an original linear problem contains many binary variables and it is more complex to deal with. However, the method can be easily and efficiently adapted to heuristic approaches.

4.5.1.5 Adapting (meha) Heuristics with RBA

When disposing of a sample verifying assumptions 1 and 2, by making use of RBA method, any constructive algorithm relying on an oracle for testing solution admissibility can be turned into an algorithm for the stochastic case. This can be done by modifying the said oracle so as to count the number of constraint violations for the given realizations and take an admissibility decision based on the threshold k.

Table 4.6 shows, as an example, the general structure of a greedy algorithm for the deterministic case as well as its adaptation for the stochastic case. The input is problem specific and consists, for the deterministic case, in giving the structure of the objective g, the constraint function G, the parameter vector ξ as well as the domain of definition for the decision variables. For the chance-constrained version, in which we consider ξ as random, we also specify a sample of size NS for ξ, the probability level ε, and, in order to apply the robust binomial approach, the confidence level α. In both cases, R represents the set of decisions not yet made (or residual), D the set of admissible decisions, $g(S)$ the solution value for solution S, d^* the current optimal decision, and S^* the optimal overall solution, build in a greedy fashion. While there are residual decisions to be made, an oracle is evaluating them for deciding the admissible decisions. Between the admissible decisions, only the one with the greatest improvement on the optimal solution value is kept and the overall solution S^* is updated. If no admissible decision is found by the oracle, the algorithm stops. As seen, the only major difference when considering chance constraints is in establishing the set of admissible solutions, **using a stochastic oracle $\mathscr{O}_s$ instead of the original one $\mathscr{O}$** (line 4). The deterministic oracle is establishing the admissibility of a residual decision by verifying the respect of the constraints, while the stochastic oracle is applying the robust binomial approach and it verifies if a residual decision is stochastically significant with a confidence level of $1 - \alpha$. For the given sample, it compares the number of constraints respected by the sample with the threshold k, established in advance in function of NS, ε and α (see the procedures for $\mathscr{O}$ and $\mathscr{O}_s$ in Table 4.7).

Of course, any optimization algorithm relying on an oracle to determine whether or not a solution is admissible (e.g., a neighboring method) can be turned into an algorithm solving the stochastic case using the same method.

4.5.2 Bertsimas and Sim-Like Robust Models

Robust optimization (RO) has gained increasing attention in the last years as another more simple framework for immunizing against parametric uncertainties in an optimization problem. With very few assumptions on the underlying uncertainty, robust methods are designed to solve special classes of programs (e.g., linear models, quadratic programs, etc.) in a mathematical programming setting. Moreover, recent models such as the ones of Ben-Tal and Nemirovski [7] and Bertsimas and Sim [13] are also providing probability guarantees.

Table 4.6 General schema for a constructive algorithm

Deterministic	Stochastic
Require: g and G functions, ξ	**Require:** g and G functions
	Require: $\tilde{\xi}_1, \dots, \tilde{\xi}_{NS}, \varepsilon, \alpha$
1: $R = \{r :$ residual decisions$\}$	1: $R = \{r :$ residual decisions$\}$
2: $S^* = \emptyset$	2: $S^* = \emptyset$
3: **while** $R \neq \emptyset$ **do**	3: **while** $R \neq \emptyset$ **do**
4: $D = \{r \in R : \mathscr{O}(r) = True\}$	4: $D = \{r \in R : \mathscr{O}_s(r) = True\}$
5: **if** $D \neq \emptyset$ **then**	5: **if** $D \neq \emptyset$ **then**
6: $d^* = \underset{d \in D}{\text{argmin}}\, g(S^* \cup \{d\})$	6: $d^* = \underset{d \in D}{\text{argmin}}\, g(S^* \cup \{d\})$
7: $S^* = S^* \cup \{d^*\}$	7: $S^* = S^* \cup \{d^*\}$
8: $R = R \setminus \{d^*\}$	8: $R = R \setminus \{d^*\}$
9: **else**	9: **else**
10: break;	10: break;
11: **end if**	11: **end if**
12: **end while**	12: **end while**
Ensure: S^*	**Ensure:** S^*

Table 4.7 Deterministic oracle versus stochastic oracle

Deterministic oracle $\mathscr{O}$	Stochastic oracle $\mathscr{O}_s$
Require: $r \in R, G, \xi$	**Require:** $r \in R, G, \tilde{\xi}_1, \dots, \tilde{\xi}_{NS}$
1: **if** $G(r, \xi) < 0$ **then**	**Require:** $k(NS, \varepsilon, \alpha)$
2: **return** $True$	1: $nbRespConstr = 0$
3: **end if**	2: **for** $i = 1$ **to** NS **do**
4: **return** $False$	3: **if** $G(r, \tilde{\xi}_i) < 0$ **then**
Ensure: $True, False$	4: $nbRespConstr{+}{+}$
	5: **end if**
	6: **if** $nbRespConstr \geq k$ **then**
	7: **return** $True$
	8: **end if**
	9: **end for**
	10: **return** $False$
	Ensure: $True, False$

4.5.2.1 Basic Ideas and Motivation

There are two important properties of robust optimization methods that make them appealing in practice [24]. First at all, robust linear models are polynomial in size and can be formalized as linear programs or second-order cone programs. In this way, one can use state-of-the art powerful and efficient LP and SOCP solvers (e.g., CPLEX 9.1) in order to solve small and medium-sized linear problems. Second, the robust methods are not making assumptions on the underlying probability distributions for the stochastic parameters, which, as seen before, there are not always available. Instead, they are applicable for cases in which only some modest distribution information is available (e.g., known mean and support). We can however remark that there are situations in which, even if the probability distribution is available, it is easier to solve the RO-based models than the exact probabilistic formulation (robust solutions obtained with several orders of magnitude faster than the exact solution).

The robust approaches are based on uncertainty sets and, in function of the assumptions made on the properties of these sets, there are different formulations, more or less tractable. With the right class of the uncertainty set, RO tractable models have been found for many well-known classes of optimization problems such as linear, quadratic, semidefinite, or even some cases of discrete problems.

The first robust models constructed feasible solutions for any realization of the uncertainty in the given set, and thus, they were too conservative. Meanwhile, the recent robust approaches permitting to choose a level of probabilistic guarantee allow more flexibility for choosing a trade-off between robustness of their solutions and performance. In contrast to the sensitivity analysis, a post-optimization tool, the probabilistic protection levels for the robust solutions are calculated a priori, in function of the structure and of the size of the uncertainty set.

4.5.2.2 Some Popular Robust Models

As expected, in general, the robust counterpart to an arbitrary optimization problem is of increased computational complexity. However, there are special classes of initial problems and types of uncertainty sets for which the robust version can be handled efficiently. In the following, we will present robust models for linear optimization problems and some general results about other different formulations.

Without loss of generality, the robust counterpart of a linear optimization program (LP) can be written as:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \quad \forall A \in \mathscr{U} \end{aligned} \tag{RLP}$$

with A constraint data matrix $m \times n$ and $\mathscr{U}$ the uncertainty set.

Uncertain constraints in LP program were first discussed by Soyster [65] which considered the case when uncertainty is "column-wise," i.e., the columns of the matrix A are data known to belong to convex sets $A_j \in \mathscr{K}_j$ and the constraints are of the form: $\sum_{j=1}^{n} A_j x_j \leq b,\ x \geq 0,\ \forall (A_j \in \mathscr{K}_j,\ j = 1, \ldots, n)$. Soyster showed that, under these constraints, the initial problem is equivalent to

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & \sum_{j=1}^{n} \bar{A}_j x_j \leq b \\ & x \geq 0 \end{aligned}$$

where $\bar{a}_{ij} = \sup_{A_j \in \mathscr{K}_j}(A_{ij})$. This model is too "pessimistic" since corresponds to the case when every entry of the constraint matrix is as large as possible and no violations of the constraints are allowed. As such, it is a worst-case approach more suitable to solve optimization problems in the context of hard real-time systems since it assures the highest protection against data variations.

In fact, the general case consists in a "row-wise" uncertainty, i.e., the one in which the rows of the constraint matrix are known to belong to convex sets: $a_i \in \mathscr{U}_i,\ i = 1, \ldots, m$. In this case, one has to well specify the properties of uncertainty sets in order to obtain formulations which can be efficiently solved.

Ben-Tal and Nemirovski [7] and El-Ghaoui [32] consider ellipsoidal uncertainty sets and, thus, obtain less conservative models. The assumption made by Ben-Tal and Nemirovski [7] is that the uncertainty set is "ellipsoidal," i.e., an intersection of a finite number of many "ellipsoids"—sets corresponding to convex quadratic inequalities. This leads to an optimization problem over a quadratic constraint[2] and a resulting dual of type second-order cone program (SOCP).

As we will see further, this study is between the first one to propose, under some restrictions, probabilistic guarantees for robust linear programs.

Also, under the same assumption of simple ellipsoidal uncertainty, one can obtain robust counterparts in the form of semidefinite optimization problems (SDF) for other classes of initial programs: quadratically constrained quadratic programs and second-order cone programs.

Les us now present another interesting robust model with different assumptions on the uncertainty set.

4.5.2.3 Bertsimas and Sim Robust Formulation

Let consider a particular row i in the constraint matrix A and J_i the set of coefficients a_{ij} in row i that vary. The model of data uncertainty proposed by Bertsimas and Sim [13] suppose that each uncertain coefficient is a symmetric and bounded random variable $\tilde{a}_{ij}$ taking values in $\left[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}\right]$.

The novelty of this approach is the introduction of parameter $\Gamma_i \in [0, |J_i|]$, not necessarily integer, allowing, for each row i, to adjust the robustness of the solutions (against the level of conservatism). As such, a feasible solution is obtained for all

[2]A quadratic constraint is a constraint of type: $\alpha_i^T x + \alpha_i \geq \|B_i x + b_i\|,\ \forall i = 1, \ldots, M$ with α_i fixed reals, a_i and b_i fixed vectors, B_i fixed matrices and $\|.\|$ standing for the usual Euclidean norm.

cases in which up to $\lfloor \Gamma_i \rfloor$ are allowed to vary in the intervals $\left[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}\right]$ and one coefficient a_{it} changes by $(\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it}$.

Under this assumption, for the initial linear program (3), one can obtain the equivalent linear robust formulation (4). This program which can then be solved using standard linear solvers (e.g., CPLEX, COIN-BC) permits to find approximate robust solutions for problems of small and medium size.

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax \le b \\ & 1 \le x \le u \end{aligned} \tag{3}$$

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & \sum_j a_{ij}x_j + z_i\Gamma_i + \sum_{j\in J_i} p_{ij} \le b_i \quad \forall i \\ & z_i + p_{ij} \ge \hat{a}_{ij}y_j \quad \forall i, j \in J_i \\ & -y_j \le x_j \le y_j \quad \forall j \\ & l_j \le x_j \le u_j \quad \forall j \\ & p_{ij} \ge 0 \quad \forall i, j \in J_i \\ & y_j \ge 0 \quad \forall j \\ & z_i \ge 0 \quad \forall i \end{aligned} \tag{4}$$

Of course, when $\Gamma_i = 0$, the constraints are the same as for the nominal problem (no uncertainty taken into account) and, if $\Gamma_i = |J_i|$ for each i, we obtain Soyster's model.

Another interesting feature for the Bertsimas and Sim method is that it can also be applied, under same hypothesis as before, to some discrete optimization problems, proposing a robust mixed integer linear equivalent to an initial mixed integer program.

4.5.2.4 Probability Guarantees

For linear optimization problems, Bertsimas and Sim [13] as well as Ben-Tal and Nemirovski [7] obtain several probability bounds against constraint violations for different uncertainty formulations.

As such, for constraints of type $\sum_j \tilde{a}_{ij}x_j \le b_i$, Ben-Tal and Nemirovski [7] assume that the uncertain data are of the form $\tilde{a}_{ij} = (1 + \varepsilon\xi_{ij})a_{ij}$ with a_{ij}, the nominal value for the coefficient, $\varepsilon \ge 0$ the given uncertainty level, $\xi_{ij} = 0$ for $j \notin J_i$ and $\{\xi_{ij}\}$ random variables independent and symmetrically distributed in $[-1, 1]$.

They then show that, for every i, the probability that constraint:

$$\sum_j a_{ij}x_j + \varepsilon\Omega\sqrt{\sum_j a_{ij}^2 x_j^2} \le b_i + \delta max\,[1, |b_i|]$$

(where $\delta > 0$ is a given feasibility tolerance and $\Omega > 0$ a reliability parameter) is violated is, at most, $exp(-\Omega^2/2)$.

As for Bertsimas and Sim model [13], the same parameter Γ controls the "price of robustness", i.e., the trade-off between the probability of constraint violation and the feasibility of the solution. So, if more than $\lfloor \Gamma_i \rfloor$ coefficients $a_{ij}, j \in J_i$ are varying, the solution remains feasible with a high probability. Several bounds for probability that the constraints are guaranteed are established for the case when the variables $\tilde{a}_{ij}$ are independent and symmetrically distributed random variables on $\left[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}\right]$. Let x^* be the optimal solution of formulation (4). Then, a first bound B_1 for the probability that each constraint i is violated is the following:

$$Pr\left(\sum_j \tilde{a}_{ij}x^* \ge b_i\right) \le exp\left(-\frac{\Gamma_i^2}{2|J_i|}\right).$$

Under same assumption of independence and symmetry for the random variables, a more tight bound B_2 for violation of constraints is established:

$$Pr\left(\sum_j \tilde{a}_{ij}x^* \ge b_i\right) \le \frac{1}{2^n}\left\{(1-\mu)\binom{n}{\lfloor \nu \rfloor} + \sum_{l=\lfloor \nu \rfloor + 1}^{n}\binom{n}{l}\right\}$$

with $n = |J_i|$, $\nu = (\Gamma_i + n)/2$ and $\mu = \nu - \lfloor \nu \rfloor$.

As for the case when the uncertain data is interdependent, Bertsimas and Sim consider a model in which only $|K_i|$ sources affect the data in the row i and each entry $a_{ij}, j \in J_i$ can be modeled as $\tilde{a}_{ij} = a_{ij} + \sum_{k \in K_i} \tilde{\eta}_{ik} g_{kj}$ where η_{ik} are independent and symmetrically distributed random variables in $[-1, 1]$. Using this model, one can also find a robust linear equivalent formulation for which the probability that the solution remains feasible is guaranteed with high probability (i.e., bound B_1 still holds).

Also, since the assumption of distributional symmetry is too limiting in many real-time situations, Chen et al. [24] propose a generalized framework for robust linear optimization with asymmetric distributions. Using some other deviation measures for random variables such as the forward and the backward deviations, they obtain an equivalent which is a second-order cone program, for which the probability of constraint violation is again being guaranteed to a requested level.

Table 4.8 Example for the values of Γ_i in function of $n = |J_i|$ and a probability of constraint violation of less than 1 % [13]

J_i	Γ_i in function of bound B_1	Γ_i in function of bound B_2
5	5	5
10	9.6	8.2
100	30.3	24.3
200	42.9	33.9
2000	135.7	105

4.5.2.5 Sensitivity Analysis on the Model's Parameters

Table 4.8 shows the choice of Γ_i as a function of $n = |J_i|$ so that the probability that a constraint is violated is less than 1 % and bounds B_1 and, respectively, B_2 are being used. It can be easily seen that the second bound dominates bound B_1 which gives unnecessarily higher values for Γ_i. For example, for $|J_i| = 2000$, in order to have a violation probability of less than 1 %, with B_2, we need to use $\Gamma = 105$ which is only $\approx$17 % of the number of uncertain coefficients. When a lower number of uncertain data is available, (e.g., $|J| = 5$), the model is equivalent to Soyster's formulation since a full protection is necessary. As such, the model of Bertsimas and Sim seems a better choice for finding less conservative solutions for problems with constraints containing a large number of uncertain data.

Moreover, as shown in [12], the Bertsimas's robust model seems to be quite robust when handling deviations in the underlying data: for a portofolio selection problem, for perturbations at 5 % level, the solution frontier is relatively unchanged while the solution frontier for Ben-Tal and Nemirovski approach [7] is severely affected.

4.6 Case Studies

4.6.1 The Partitioning, Placement and Routing of Dataflow Networks

As mentioned in Sect. 4.2.3, one important step in the compilation of dataflow applications for massively parallel systems is the resource allocation, with the associated optimization problems of partitioning, placement, and routing.

In the problem of partitioning of networks of processes, the objective is to assign the tasks to a fixed number of processors in order to minimize communications between processors while respecting the capacity of each processor in terms of resources (memory footprint, core occupancy, etc.). The chance-constrained case considered in [66] consists in taking into account the uncertainty coming from the execution times on the resources defining the weights of the tasks (e.g., core occupancy) and, thus, have probabilistic capacity constraints for the sum of resources affected to a node. Since a multistart constructive algorithm was already available for solving the nominal case, the authors took advantage of the existing implementation and adapt it, using the robust binomial approach to transform the deterministic admissibility oracle into a stochastic one. The accent is put more on the design-for-use methodology and, for the experimental results, on the price of robustness compared with the deterministic version for different thresholds on the involved parameters—ξ, the probability guarantee, α, the confidence level and N, the size of samples. The importance of taking into account the variations on the weights of the tasks is shown: for half of the stochastic instances, the solutions of the corresponding deterministic problems are not feasible.

Another possible application of the robust binomial approach is for the stochastic problem of joint placement and routing [67] the purpose of it being to map dataflow applications on a clusterized parallel architecture by making sure that the capacities of the clusters are respected and that, for the found placement, there exists a routing through the links of the underlying Network-On-Chip, respecting the maximal available bandwidth. If, again, the resources of the tasks depending on the uncertainty times are uncertain, one has to be able to solve the chance-constrained problem with probability capacity constraints for the nodes. The robust binomial approach was applied in the framework of a GRASP (greedy randomized adaptive search procedure) method and extensive computational tests were performed on 1920 synthetic instances as well as on samples from a real application of motion detection. Different configurations for the model parameters were tested, with $\varepsilon, \alpha \in \{0.01, 0.1\}$. Under same configuration, for more than 70 % of instances, the quality of the stochastic solutions is within 5 % from the quality of deterministic ones. As for the computation time, as expected, it depends on the size of the initial problem: in this case, on the number of clusters and of the size of the available sample.

4.6.2 The Dimensioning of Communications Buffers

Another crucial step in the compilation of an application ΣC is the memory dimensioning for the communication buffers. Bodin et al. [16] treats thus one fundamental problem for the compiler to solve in order to achieve performance, the minimization of the buffer sizes under a throughput constraint. Let us explain more in detail the formulation and point out where the uncertainty is ignored.

As said previously, a CSDF application can be seen as a directed graph $G = (T, A)$ with T, the set of actors (tasks) and A, the set of buffers (arcs or channels). Every actor $t \in T$ is decomposed into $\phi(t) \in \mathbb{N}^*$ phases and each k^{th} phase ($\forall k \in \{1, \dots, \phi(t)\}$), denoted t_k, has a duration $d(t_k) \in \mathbb{R}$. Also, each actor is executed several times: for every phase k, the n^{th} execution of this phase of task $t \in T$ is $\langle t_k, n\rangle$.

For every couple $(k, n) \in \{1, \dots, \phi(t)\} \times \mathbb{N}^*$, the preceding execution phase of $\langle t_k, n\rangle$ is:

$$Pred\langle t_k, n\rangle = \begin{cases} \langle t_{k-1}, n\rangle & \text{if } k > 1 \\ \langle t_{\phi(t)}, n-1\rangle & \text{if } k = 1 \end{cases}$$

Every arc $a = (t, t') \in A$ has an associated buffer $b(a)$ from actor t to actor t', containing initially $M_0(a) \in \mathbb{N}$ tokens of stored data. $\forall k \in \{1, \dots, \phi(t)\}, in_a(k) \geq 0$ data are produced at the end of execution of t_k and, similarly, $\forall k' \in \{1, \dots, \phi(t')\}$, $out_a(k') \geq 0$ data are consumed before $t'_{k'}$ starts its execution.

Let us also define $I_a\langle t_k, n\rangle$ the total number of data produced by t in buffer $b(a)$ at the completion of $\langle t_k, n\rangle$ which can be computed recursively as: $I_a\langle t_k, n\rangle = I_a Pred\langle t_k, n\rangle + in_a(k)$. Similarly, $O_a\langle t'_{k'}, n'\rangle$ is the number of data, computed recursively as $O_a Pred\langle t'_{k'}, n'\rangle + out_a(k')$, consumed by t' in buffer $b(a)$ at the completion of $\langle t'_{k'}, n'\rangle$. The amount of tokens, respectively, produced and consumed in $b(a)$ during the entire iteration of actors t and t' are noted $i_a = I_a\langle t_{\phi(t)}, 1\rangle$ and $o_a = O_a\langle t_{\phi(t')}, 1\rangle$.

Let also suppose that each buffer $b(a)$ associated with arc $a = (t, t')$ is bounded. This constraint can be modeled using a feedback arc $a' = (t', t) \in Fb(A)$ for each arch $a = (t, t')$ in A. As such, the initial size of the buffer is $b(a) = M_0(a) + M_0(a')$ and, if $\theta(a) = \theta(a')$ is the size of data stored in $b(a)$, the size of the buffer will be exactly $M_0(a)\theta(a) + M_0(a')\theta(a')$. The whole size of G is then $\sum_{a \in \{A \cup Fb(A)\}} \theta(a) M_0(a)$.

If Th^*_G is the minimal required throughout, the problem consists in finding integer values for each $M_0(a')$ such that the throughput is at least Th^*_G and the whole buffer size is minimal.

Using a periodic schedule, it is possible to model this problem as a linear integer program less general and easier to define. Let S be a function defining a valid schedule that, for each (t, k, n) with $t \in T$, $\forall k \in \{1, \dots, \phi(t)\}$ and $n \in \mathbb{N}^*$, associates a starting time $S\langle t_k, n\rangle \in \mathbb{R}$ for the nth execution of t_k such that no data is read before it is produced (the number of data in each buffer is positive). The throughput of a periodic actor t is $Th^S_t = \frac{1}{\mu^S_t}$, where μ^S_t is the period of t for schedule S and then the throughout of a schedule is equal to $Th^S_G = \frac{1}{\mu^S_t q_t}$ for any actor $t \in T$ with q_t the repetition vector of G. The inverse of this throughout is the period of the periodic schedule S, defined as $\Omega^S_G = \mu^S_t q_t$.

With this periodic schedule, S are defined precedence constraints on the phase executions of tasks t and t'. As such, for two executions $\langle t_k, n\rangle$ and $\langle t'_{k'}, n'\rangle$ with n and n' in $\mathbb{N}^*$, the arc $a = (t, t')$ and the couple $(k, k') \in \{1, \dots, \phi(t)\} \times \{1, \dots, \phi(t')\}$ we define the variables:

$$\alpha^{min}_a(k, k') = \lceil max\{0, in_a(k) - out_a(k')\} + O_a\langle t'_{k'}, 1\rangle - I_a\langle t_k, 1\rangle - M_0(a)\rfloor^{gcd_a}$$
$$\alpha^{max}_a(k, k') = \lfloor O_a\langle t'_{k'}, 1\rangle - I_a Pred\langle t_k, 1\rangle - M_0(a) - 1\rfloor^{gcd_a}$$

where gcd_a is the greatest common divisor between i_a and o_a for arc a, $\lceil\alpha\rceil^\gamma = \lceil\frac{\alpha}{\gamma}\rceil \times \gamma$ and $\lfloor\alpha\rfloor^\gamma = \lfloor\frac{\alpha}{\gamma}\rfloor \times \gamma$.

Finally, for every arc a, let define the set of couples $\mathscr{Y}_A = \{(k, k') \in \{1, \dots, \phi(t)\} \times \{1, \dots, \phi'(t)\}, \alpha^{min}_a(k, k') \leq \alpha^{max}_a(k, k')\}$.

Let also assume that the period Ω^S_G is fixed in advance. Using the above defined variables, the optimization problem considered can be formulated as the integer linear system (5) [16].

$$\min_{a \in Fb(A)} \quad \theta(a) M_0(a) \tag{5}$$

$$\forall a \in A, \forall (k,k') \in \mathscr{Y}(a), \quad S\langle t'_{k'}, 1\rangle - S\langle t_k, 1\rangle \geq d(t_k) + \Omega_G^S \times \frac{\alpha_a^{max}(k,k')}{i_a \times q_t}$$

$$\forall a \in Fb(A), \forall k \in \{1, \dots, \phi(t)\}, \forall k' \in \{1, \dots, \phi(t)\},$$

$$S\langle t'_{k'}, 1\rangle - S\langle t_k, 1\rangle \geq d(t_k) + \Omega_G^S \times \frac{f_a(k,k') \times gcd_a}{i_a \times q_t}$$

$$u_a(k,k') = O_a\langle t'_{k'}, 1\rangle - I_a Pred\langle t_k, 1\rangle - M_0(a) - 1$$

$$f_a(k,k') \times gcd_a \geq u_a(k,k') - gcd_a + 1$$

$$f_a(k,k') \in \mathbb{N}, u_a(k,k') \in \mathbb{N}$$

$$\forall a \in Fb(A), \quad M_0(a) \in \mathbb{N}$$

$$\forall t \in T, \forall k \in \{1, \dots, \phi(t)\}, \quad S\langle t_k, 1\rangle \in \mathbb{R}^+$$

This mixed integer program is then solved for the deterministic case using the commercial solver Gurobi optimizer tool [59].

One can analyze this model and see that the uncertainties associated with the successive durations of the different phases for tasks executions are localized in the coefficients $d(t_k)$ of the first and second sets of constraints. Therefore, after reformulating the program in a convenient way (i.e., a model similar to (3)) and under the hypothesis of a bounded and symmetric support, one could easily apply Bertsimas and Sim approach [13]. We can then control the robustness of solutions with the parameter Γ which size depends on T, the number of actors, $\phi(t)$ the number of phases for each task t and n, the number of executions for a task.

4.7 Conclusion

The multicore and manycore systems are between the future architectures for server acceleration and embedded devices. In order to fully exploit the huge potential of these architectures, one has to be able to write parallel applications correctly and efficiently. Dataflow paradigms seems a good choice for designing embedded applications without worrying about data synchronization and about the underlying parallelism, left in the "hands" of the compiler.

As such, the high parallel embedded architectures need also innovative compilation techniques, making use of advanced operation research methods for better optimizations. Recent advances in optimization under uncertainty approaches make them appealing in an operational manner for domains where data are uncertain, as it is the case of embedded systems where execution times are subject to variations due to the application inputs and to certain modern hardware characteristics.

Since the execution times are random variables difficult to analytically characterize, the most suitable methods of optimization under uncertainty for embedded applications are nonparametric, with few assumptions on the distribution of the parameters involved in the optimization model. In this chapter, we have presented methods from two distinct classes of approaches: the robust binomial approach [68], an extension of the sample-based approaches grounded in statistical testing theory, and the model of Bertsimas and Sim [13], from the robust optimization field. The former, more adapted for a setting where the random variables are dependent and where it is hazardous to make any particular assumption on the distribution, provides approximation solutions for medium and large-sized difficult optimization problems. The latter is more adapted for small and medium optimization problems in order to find exact solutions, when one can make minor assumptions on the support and on the variance of the uncertain parameters. We have also illustrated two possible user cases, appearing in the compilation of cyclo-static dataflow applications.

While this work is only an introduction to the optimization under uncertainty techniques, we hope it is a good starting point for those interested in the design of high performing embedded manycore systems using advanced and robust operation research methods.

References

1. Aringhieri, R.: Solving Chance-constrained Programs Combining Tabu Search and Simulation, pp. 30–41. Springer, Berlin (2004)
2. Aubry, P., Beaucamps, P.E., Blanc, F., Bodin, B., Carpov, S., Cudennec, L., David, V., Dore, P., Dubrulle, P., Dupont, B.d.D., Galea, F., Goubier, T., Harrand, M., Jones, S., Lesage, J., Louise, S., Chaisemartin, N., Nguyen, T., Raynaud, H., Sirdey, R.: Extended cyclostatic dataflow program compilation and execution for an integrated manycore processor. In: Proceedings of the First International Workshop on Architecture, Languages, Compilation and Hardware Support for Emerging Manycore Systems (ALCHEMY 2013), Barcelona, Spain, pp. 1624–1633 (2013)
3. Bell, S., Edwards, B., Amann, J., Conlin, R., Joyce, K., Leung, V., MacKay, J., Reif, M., Bao, L., Brown, J., Mattina, M., Miao, C.C., Ramey, C., Wentzlaff, D., Anderson, W., Berger, E., Fairbanks, N., Khan, D., Montenegro, F., Stickney, J., Zook, J.: TILE64 - processor: a 64-core soc with mesh interconnect. In: IEEE International Solid-State Circuits Conference, ISSCC 2008. Digestof Technical Papers, pp. 88–598 (2008)
4. Bellman, R.E., Zadeh, L.A.: Decision-making in a fuzzy environment. Manag. Sci. **17**(4), B-141–B-164 (1970)
5. Ben-Ameur, W., Kerivin, H.: Routing of uncertain traffic demands. Optim. Eng. **6**(3), 283–313 (2005)
6. Ben-Ameur, W., Zotkiewicz, M.: Robust routing and optimal partitioning of a traffic demand polytope. Int. Trans. Oper. Res. **18**(3), 307–333 (2011)
7. Ben-Tal, A., Nemirovski, A.: Robust solutions of linear programming problems contaminated with uncertain data. Math. Program. **88**, 411–424 (2000)
8. Ben-Tal, A., Nemirovski, A.: Robust optimization: methodology and applications. Math. Program. **92**(3), 453–480 (2002)
9. Ben-Tal, A., Nemirovski, A.: On safe tractable approximations of chance-constrained linear matrix inequalities. Math. Oper. Res. **34**, 1–25 (2009)
10. Benini, L., Flamand, E., Fuin, D., Melpignano, D.: P2012: building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2012, pp. 983–987 (2012)
11. Beraldi, P., Ruszczynski, A.: Beam search heuristic to solve stochastic integer problems under probabilistic constraints. Eur. J. Oper. Res. **167**(1), 35–47 (2005)

12. Bertsimas, D., Nohadani, O.: Robust optimization with simulated annealing. J. Glob. Optim. **48**, 323–334 (2010). doi:10.1007/s10898-009-9496-x
13. Bertsimas, D., Sim, M.: The price of robustness. Oper. Res. **52**(1), 35–53 (2004). doi:10.1287/opre.1030.0065
14. Bilsen, G., Engels, M., Lauwereins, R., Peperstraete, J.: Cyclo-static data flow. In: 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95), vol. 5, pp. 3255–3258 (1995)
15. Bilsen, G., Engels, M., Lauwereins, R., Peperstraete, J.: Cycle-static dataflow. IEEE Trans. Signal Process. **44**(2), 397–408 (1996)
16. Bodin, B., Munier Kordon, A., Dupont de Dinechin, B.: Periodic schedules for cyclo-static dataflow. In: ESTImedia, pp. 105–114 (2013)
17. Burns, A., Bernat, G., Broster, I.: A probabilistic framework for schedulability analysis. In: Proceedings of the Third International Conference on Embedded Software (EMSOFT 2003), pp. 1–15 (2003)
18. Calafiore, G., Campi, M.: Uncertain convex programs: randomized solutions and confidence levels. Math. Program. **102**, 25–46 (2005)
19. Calafiore, G., Campi, M.: The scenario approach to robust control design. IEEE Trans. Autom. Control **51**(5), 742–753 (2006)
20. Campi, M., Garatti, S.: A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. J. Optim. Theory Appl. **148**(2), 257–280 (2011)
21. Carpov, S., Cudennec, L., Sirdey, R.: Throughput constrained parallelism reduction in cyclo-static dataflow applications. Procedia Comput. Sci. **18**, 30–39 (2013)
22. Carpov, S., Sirdey, R., Carlier, J., Nace, D.: Memory bandwidth-constrained parallelism dimensioning for embedded many-core microprocessors. In: CPAIOR10 Workshop on Combinatorial Optimization for Embedded System Design, Bologna, Italy (2010)
23. Charnes, A., Cooper, W.: Chance-constrained programming. Manag. Sci. **6**, 73–89 (1959)
24. Chen, X., Sim, M., Sun, P.: A robust optimization perspective on stochastic programming. Oper. Res. **55**(6), 1058–1071 (2007)
25. Dantzig, G.: Linear programming under uncertainty. Manag. Sci. **1**(3–4), 197–206 (1955)
26. Dentcheva, D., Prékopa, A., Ruszczynski, A.: Concavity and efficient points of discrete distributions in probabilistic programming. Math. Program. **89**, 55–77 (2000)
27. Dentcheva, D., Prékopa, A., Ruszczynski, A.: Bounds for probabilistic integer programming problems. Discret. Appl. Math. **124**(1–3), 55–65 (2002)
28. Devarakonda, M., Iyer, R.: Predictability of process resource usage: a measurement-based study on unix. Softw. Eng. IEEE Trans. **15**(12), 1579–1586 (1989)
29. Diaz, J., Garcia, D., Kanghee, K., Chang-Gun, L., Lo Bello, L., Lopez, J., Sang Lyul, M., Mirabella, O.: Stochastic analysis of periodic real-time systems. In: Real-Time Systems Symposium (RTSS 2002), pp. 289–300 (2002)
30. Dupont de Dinechin, B., Ayrignac, R., Beaucamps, P., Couvert, P., Ganne, B., Guironnet de Massas, P., Jacquet, F., Jones, S., Morey Chaisemartin, N., Riss, F., Strudel, T.: A clustered manycore processor architecture for embedded and accelerated applications. In: HPEC, pp. 1–6 (2013)
31. Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Taylor and Francis, Oxford (1994)
32. El Ghaoui, L., Oustry, F., Lebret, H.: Robust solutions to uncertain semidefinite programs. Siam J. Optim. **9**(1), 33–52 (1998)
33. Freund, R.F.: Optimal selection theory for superconcurrency. In: Proceedings of the 1989 ACM/IEEE Conference on Supercomputing, Supercomputing'89, pp. 699–703. ACM, New York (1989)
34. Gaivoronski, A., Lisser, A., Lopez, R., Xu, H.: Knapsack problem with probability constraints. J. Global Optim. **49**, 397–413 (2011)
35. Galea, F., Sirdey, R.: Méthode de cadencement d'applications flot de données cyclostatiques. Technical report, CEA LIST/DACLE/10-070 (2010)
36. Galea, F., Sirdey, R.: A parallel simulated annealing approach for the mapping of large process networks. In: IPDPS Workshop, pp. 1787–1792 (2012)

37. Gustafson, J.: Reevaluating Amdahl's law. Commun. ACM **31**(5), 532–533 (1988)
38. Hanen, C., Munier, A.: Cyclic scheduling on parallel processors: an overview. In: Chrétienne, P., Coffman, E.G., Lenstra, J.K., Liu, Z. (eds.) Scheduling Theory and Its Applications. Wiley, New York (1994)
39. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco (2003)
40. Henrion, R., Strugarek, C.: Convexity of chance constraints with independent random variables. Comput. Optim. Appl. **41**(2), 263–276 (2008)
41. Hong, L., Yang, Y., Zhang, L.: Sequential convex approximations to joint chance constrained programs: a Monte Carlo approach. Oper. Res. **59**(3), 617–630 (2011)
42. Howard, J., Dighe, S., Hoskote, Y., Vangal, S., Finan, D., Ruhl, G., Jenkins, D., Wilson, H., Borkar, N., Schrom, G., Pailet, F., Jain, S., Jacob, T., Yada, S., Marella, S., Salihundam, P., Erraguntla, V., Konow, M., Riepen, M., Droege, G., Lindemann, J., Gries, M., Apel, T., Henriss, K., Lund-Larsen, T., Steibl, S., Borkar, S., De, V., Van Der Wijngaart, R., Mattson, T.: A 48-core ia-32 message-passing processor with dvfs in 45 nm cmos. In: IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp. 108–109 (2010)
43. Iverson, M., Ozguner, F., Follen, G.: Run-time statistical estimation of task execution times for heterogeneous distributed computing. In: Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing, pp. 263–270 (1996)
44. Kahn, G.: The semantics of simple language for parallel programming. In: IFIP Congress, pp. 471–475 (1974)
45. Khokhar, A.A., Prasanna, V.K., Shaaban, M.E., Wang, C.L.: Heterogeneous computing: challenges and opportunities. Computer **26**(6), 18–27 (1993)
46. Klopfenstein, O.: Optimisation robuste de réseaux de télécommunications. Ph.D. thesis, Orange Labs, Laboratoire Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne (2008)
47. Lee, E., Parks, T.: Dataflow process networks. Proc. IEEE **83**(5), 773–801 (1995)
48. Lee, J., Shin, I., Easwaran, A.: Online robust optimization framework for QoS guarantees in distributed soft real-time systems. In: Proceedings of the tenth ACM International Conference on Embedded Software, EMSOFT'10, pp. 89–98. New York (2010)
49. Lemerre, M., David, V., Aussagues, C., Vidal-Naquet, G.: Equivalence between schedule representations: theory and applications. In: Real-Time and Embedded Technology and Applications Symposium, RTAS'08, pp. 237–247. IEEE (2008). doi:10.1109/RTAS.2008.17
50. Li, P., Wendt, M., Wozny, G.: Robust model predictive control under chance constraints. Comput. Chem. Eng. **24**(2–7), 829–834 (2000)
51. Lombardi, M., Milano, M., Ruggiero, M., Benini, L.: Stochastic allocation and scheduling for conditional task graphs in multiprocessor systems-on-chip. J. Sched. **13**, 315–345 (2010)
52. Loughlin, D.H., Ranjithan, S.: Chance-constrained genetic algorithms. In: GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 369–376 (1999)
53. Louise, S.: The future of programming embedded systems: methods for timing or for performance support. HDR, EDIPS, Paris-Sud, France (2014)
54. Luedtke, J., Ahmed, S.: A sample approximation approach for optimization with probabilistic constraints. SIAM J. Optim. **19**(2), 674–699 (2008)
55. Manolache, S., Eles, P., Peng, Z.: Memory and time-efficient schedulability analysis of task sets with stochastic execution time. In: 24th Euromicro Conference on Real-Time Systems, pp. 0019 (2001)
56. Mazouz, A., Touati, S.A.A., Barthou, D.: Study of variations of native program execution times on multi-core architectures. In: CISIS, pp. 919–924 (2010)
57. Miller, B., Wagner, H.: Chance constrained programming with joint constraints. Oper. Res. **13**(6), 930–945 (1965)
58. Nemirovski, A., Shapiro, A.: Convex approximations of chance constrained programs. SIAM J. Optim. **17**(4), 969–996 (2006)
59. Optimization, G.: Gurobi—state of art mathematical programming solver (2014). http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview

60. Pagnoncelli, B.K., Ahmed, S., Shapiro, A., Pardalos, P.M.: Sample average approximation method for chance constrained programming: theory and applications. J. Optim. Theory Appl. **142**, 399–416 (2009)
61. Pal, B., Gupta, S., Chakraborti, D.: A genetic algorithm based stochastic simulation approach to chance constrained interval valued multiobjective decision making problems. In: 2010 International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1–7 (2010)
62. Prékopa, A.: Stochastic Programming. Kluwer Academic, Dordrecht (1995)
63. Reistad, B., Gifford, D.K.: Static dependent costs for estimating execution time. SIGPLAN Lisp Pointers VI **VII**(3), 65–78 (1994)
64. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. J. Risk **2**, 21–41 (2000)
65. Soyster, A.L.: Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper. Res. **21**(5), 1154–1157 (1973)
66. Stan, O., Sirdey, R., Carlier, J., Nace, D.: A heuristic algorithm for stochastic partitioning of process networks. In: Proceedings of the 16th IEEE International Conference on System Theory, Control and Computing (ICSTCC), pp. 1–6 (2012)
67. Stan, O., Sirdey, R., Carlier, J., Nace, D.: A GRASP for placement and routing of dataflow process networks on manycore architectures. In: 3PGCIC, pp. 219–226 (2013)
68. Stan, O., Sirdey, R., Carlier, J., Nace, D.: The robust binomial approach to chance-constrained optimization problems with application to stochastic partitioning of large process networks. J. Heuristics **20**(3), 261–290 (2014)
69. Tanner, M.W., Beier, E.B.: A general heuristic method for joint chance-constrained stochastic programs with discretely distributed parameters (2007). http://www.optimization-online.org/DB_FILE/2007/08/1755.pdf
70. UTK, ORNL: Netlib repository (2014). http://www.netlib.org/
71. Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat, J., Stenström, P.: The worst-case execution-time problem: overview of methods and survey of tools. ACM Trans. Embed. Comput. Syst. **7**(3), 36:1–36:53 (2008)
72. Xu, H., Caramanis, C., Mannor, S.: Optimization under probabilistic envelope constraints. Oper. Res. **60**(3), 682–699 (2012)
73. Yang, J., Ahmad, I., Ghafoor, A.: Estimation of execution times on heterogeneous supercomputer architectures. In: International Conference on Parallel Processing, (ICPP 1993), vol. 1, pp. 219–226 (1993)

Chapter 5
Digital IIR Filter Design with Fix-Point Representation Using Effective Evolutionary Local Search Enhanced Differential Evolution

Yu Wang, Weishan Dong, Junchi Yan, Li Li, Chunhua Tian, Chao Zhang, Zhihu Wang and Chunyang Ma

Abstract Previously, the parameters of digital IIR filters were encoded with floating-point representations. It is known that a fixed-point representation can effectively save computational resources and is more convenient for direct realization on hardware. Inherently, compared with floating-point representation, fixed-point representation may make the search space miss much useful gradient information and, therefore, raises new challenges. In this chapter, the universality of DE-based MA is improved by implementing more efficient evolutionary algorithms (EAs) as the local search techniques. The performance of the newly designed algorithm is experimentally verified in both function optimization tasks and digital IIR filter design problems.

Y. Wang (✉)
Alibaba Group, Hangzhou, China
e-mail: dilun.wy@alibaba-inc.com, wybarca@gmail.com

Y. Wang · W. Dong · J. Yan · L. Li · C. Tian · C. Zhang · Z. Wang · C. Ma
IBM Research-China, Beijing, China

W. Dong
e-mail: dongweis@cn.ibm.com

J. Yan
e-mail: yanjc@cn.ibm.com

L. Li
e-mail: lili.crl.cn@cn.ibm.com

C. Tian
e-mail: chtian@cn.ibm.com

C. Zhang
e-mail: bjzchao@cn.ibm.com

Z. Wang
e-mail: zhwang@cn.ibm.com

C. Ma
e-mail: machybj@cn.ibm.com

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_5

5.1 Introduction

Considered as an important but very hard task in digital signal processing, digital IIR filter design has attracted much research attention in recent decades [5, 6, 19, 24, 27, 68–70, 94]. Digital IIR filter is one of the most frequently used computation tools in digital signal processing systems. In many applications, such as high-speed and low-power communication transceivers, it is routinely employed as a custom designed digital block [6].

The cascade form of an infinite impulse response (IIR) filter can be described as follows [5, 6, 19, 24, 27, 68–70, 94]:

$$H(z) = K \prod_{k=1}^{n} \frac{1 + b_k z^{-1}}{1 + a_k z^{-1}} \prod_{i=1}^{m} \frac{1 + d_{i1} z^{-1} + d_{i2} z^{-2}}{1 + c_{i1} z^{-1} + c_{i2} z^{-2}} \tag{5.1}$$

where K is the gain, a_k and b_k for $k = 1, 2, \ldots, n$ are the first-order coefficients, and c_{i1}, c_{i2}, d_{i1}, and d_{i2} for $i = 1, 2, \ldots, m$ are the second-order coefficients.

5.1.1 Problem Statement

As described in [70], the important task of the designer is to find values of $[a_0 \ldots a_n, b_0 \ldots b_n, c_{11} \ldots c_{1m}, c_{21} \ldots c_{2m}, d_{11} \ldots d_{1m}, d_{21} \ldots d_{2m}]$ that produce the desired response. In EAs, the designing objective is represented in a fitness function, which can be formulated as minimization of the magnitude response error. This situation can be simulated as the difference to the boundary of design requirement. The fitness value can be calculated as follows, which have been used by [68, 94]:

$$H_p(\omega_p) = \begin{cases} 1 - \delta_1 - |H(e^{j\omega_p})|, & |H(e^{j\omega_p})| < 1 - \delta_1 \\ 0, & |H(e^{j\omega_p})| \geq 1 - \delta_1, \end{cases} \tag{5.2}$$

where ω is in the passband and the $H_p(\omega)$ is the passband magnitude response error at ω.

$$H_s(\omega_s) = \begin{cases} |H(e^{j\omega_s})| - \delta_2, & |H(e^{j\omega_s})| \geq \delta_2 \\ 0, & |H(e^{j\omega_s})| \leq \delta_2, \end{cases} \tag{5.3}$$

where ω is in the stopband and the $H_s(\omega)$ is the stopband magnitude response error at ω.

$$\min f_1 = \frac{1}{P_n} \sum_{i=1}^{P_n} H_p(\omega_i) + \frac{1}{S_n} \sum_{j=1}^{S_n} H_s(\omega_j) \tag{5.4}$$

where P_n and S_n are the sampling frequency in the passband and stopband, respectively.

In order to make the designed filter feasible and implementable, the stability of the candidates should be guaranteed. The stability requirement of digital IIR filter has been summarized by [64]. The constraints used in this chapter can be expressed as follows:

$$-1 < a_{j2} < 1, \tag{5.5}$$

$$-1 - c_{j2} < c_{j1} < 1 + c_{j2}. \tag{5.6}$$

where Eq. (5.5) represents the stability requirement for first-order blocks and Eq. (5.6) denotes that for the second-order blocks.

5.1.2 Literature Review

In previous works, some classical methods have been proposed to tackle digital IIR filter design. The bilinear transformation approach, illustrated by [5], is one of the early techniques and has been widely adopted. Via this approach, a digital filter is transformed to a corresponding analog low-pass (LP) filter. Then, well-known LP filter design methods, such as Butterworth, Chebyshev Type I, and Chebyshev Type II, can be used to accomplish the design of the analog LP filter. Finally, the analog LP filter is transformed back to the digital filter by again using a bilinear transformation. However, this procedure requires much pre-knowledge and shows poor performance in most cases [70]. Therefore, this stimulated the research on more effective optimization approaches with less prior knowledge and higher accuracy to obtain good digital IIR filter designs, which has been experimentally proven by [68, 94].

Since the seminal work of [19], diverse evolutionary algorithms (EAs) have been developed for digital IIR filter design. The major advantages of these EAs over other methods were summarized by [76, 79]: (1) pre-knowledge of the problems is not necessary for the application of EAs, while the highly nonlinear characteristic must be approximated first for transformation methods and other mathematical optimization approaches; (2) EAs usually work with a population of candidate solutions and can handle constraints adaptively under the strategy set beforehand in a single run.

One of the design ways is parameter estimation, whose process can be described as follows: given the settings of the digital IIR filter, the method is expected to provide a final digital IIR filter to meet all requirements. Important contributions of this class are hybrid genetic algorithm proposed by [24], the hierarchical genetic algorithm (HGA) introduced by [68], and the Taguchi-strategy enhanced GA (HTGA) designed by [70]. Due to the fact that the fitness landscape of digital IIR filter design problems contains too many local optima, [79] presents that the motivation of these algorithms is to develop specific operators to strengthen the exploration ability. Two recently published works of [79, 94] implemented multi-objective evolutionary algorithms to simultaneously optimize three objectives: the magnitude response error, the linear phase response error, and the order. Especially in [79], without pre-knowledge, the three objectives are considered equally important during the optimization procedure. Based on the experimental results on four types of low order digital IIR filter design, the multi-objective digital IIR filter design is very promising.

Another important design way is system identification. Compared with the above class, the major difference is that the fitness value of one individual changes from time to time. The reason can be ascribed to the different input test sequences, which are always randomly or probabilistically generated. Therefore, it is apparent that this problem belongs to the noise-induced optimization domain. The typical methods include ant colony optimization (ACO) [32], seeker optimization [7], artificial immune systems [30], tabu search [31], and structured stochastic optimization [36]. Since these optimization algorithms work in uncertain environments, the accuracy cannot be effectively guaranteed. Hence, all the above algorithms are only applicable to very low order digital IIR filter designs (not higher than seven orders) and furthermore, all algorithms in comparison in [7] cannot obtain satisfactory solutions for some hard problems.

Although there have been a number of successful EA-based digital IIR filter design works as summarized above, most of them only consider parameters that take on value from a continuous domain. As presented in [27, 54, 71], very little research takes fixed-point representations into consideration. However, in the development and application of embedded systems, fixed-point implementations of digital IIR filters are more and more common in real applications. Therefore, optimal design based on fixed-point representations quickly gains significance. The previous limited works on fixed-point digital IIR filter design just tackled some relatively simple problems and are lack of necessary comparison analysis. In this chaper, we try to bridge this gap by applying our proposed MA(DE-LS) to higher order and more difficult digital IIR filter design problems.

5.1.3 Chapter Structure

The remainder of this chapter is structured as follows: In Sect. 5.2, the related optimization methods, especially memetic algorithms are reviewed. In Sect. 5.3,the framework MA(DE-LS) is first introduced, followed by the suboptimizers, including global search method and local search techniques. Section 5.4 experimentally investigates the impacts of different suboptimizers. Section 5.5 provides a further experimental evaluation of the MA(DE-LS). Several state-of-the-art continuous EAs and MAs, are utilized to provide comparisons. In Sect. 5.6, MA(DE-LS) is compared with multiple techniques on four types of digital IIR filter design tasks. In Sect. 5.7, the conclusion is given and the future work is outlined.

5.2 Related Works in Optimization Algorithm

Over the past two decades, memetic algorithms (MAs) have attracted increasing attention from the research community [29, 35, 45, 51]. MAs have been successfully implemented in various scientific and engineering applications, and have gained better efficiency in many cases [26, 72]. As presented by [51], MAs is generally considered as a framework that combines population based global search algorithms and local search techniques. The global search algorithms usually use modern meta-heuristic search techniques such as evolutionary algorithms (EAs) [11, 12, 17], to explore the search space [18, 20–23, 34, 40, 47, 58, 63, 67, 95, 96]. The search procedure is usually a computational expensive process to find the optimal solutions. On the other hand, local search techniques, such as the steepest descent method [9], the conjugate gradient method [28] and quadratic programming [49], perform an iterative search for the optimal solution within the neighborhood of a given candidate. In comparison with the global search, an individual using local search strategy can reach a near optimal solution (usually one local optimum in the multimodal problems) more easily. Therefore, by using a hybrid of global search and local search, MA framework can benefit from the merits of these two aspects.

In many previous MA variants, differential evolution (DE), first proposed by [65], has been frequently used as the global search method. DE-based MAs are commonly used in engineering optimization problems. For example, [62] combined a chaotic control DE with sequential quadratic programming (DEC-SQP) for economic dispatch optimization of power system. Recently, [41] proposed an adaptive DE-based MA, which hybridizes parameters adaptive DE and local random search together to solve a dynamic economic dispatch task. Using a similar idea, [42] designed another MA using an adaptive chaotic DE and chaotic local search operation for short-term hydrothermal generation scheduling problem. Besides the mathematical local search operators, [39] introduced that some other effective heuristic methods can be applied to strengthen the exploitation capability. For example, random walk with direction exploitation and harmony search were embedded into DE to form two DE-based MAs for engineering design problems. For more comprehensive surveys on DE-based MAs, please refer to the following tutorials: [4, 8, 35, 46, 52, 53]. As most existing MAs were used for specific applications, [48] proposed a probabilistic memetic framework (PrMF) for general use, in which DE and GA can be used as the global search method. In PrMF, the global search and local search are balanced by governing the learning intensity of each individual according to the theoretical upper bound derived based on the feedback during the optimization procedure. The function optimization results have effectively demonstrated the superiorities of PrMF compared with classical MAs and fast EAs. However, the universality of PrMA still needs to be improved. In order to accelerate DE for general use, [50] designed several DE-based MAs by altering the local search technique. The influences of different local search techniques in terms of efficiency and effectiveness have been experimentally investigated on a test function suit.

As presented above, most existing DE-based MAs implement mathematical exact methods as the local search techniques. Many other methods always embed some heuristic methods with good efficiency to strengthen the exploitation ability of DE. This is not surprising, because the diverse performance of different optimization methods has been theoretically illustrated in no free lunch theorems proposed by [84]. As presented by [55, 59, 60, 66, 73, 77, 78, 80–83], some newly proposed EAs consider using more than one evolutionary new offspring generating operators. This also implied the different strengths of different EAs. In this chapter, we investigate how more robust local search techniques can be used to improve both efficiency and effectiveness of DE in an adaptive MA framework MA(DE-LS). The highlights are listed as follows:

- Two EAs are used as local search technique for purposive comparative study, including covariance matrix adaptation evolution strategy (CMAES) [25] and self-adaptive mixed distribution-based univariate estimation of distribution algorithm (MUEDA) [76]. Based on the experiments, some important experiences of designing MA(DE-LS) are obtained.
- To illustrate the advantages of MA(DE-LS) over EAs, we compare our method with several effective global search EAs on 26 functions with diverse characteristics, such as unimodality, multimodality, rotation, ill-condition, and mis-scalability. The experimental results MA(DE-LS) can perform better.
- Compared with traditional DE-based MAs and the state-of-the-art MAs, the progresses of MA(DE-LS) are experimentally verified on function optimization tasks.
- A real-world application, the digital IIR filter design with fixed-point representation, is used to evaluate the practical applicability of MA(DE-LS). Compared with several state-of-the-art EAs, MA(DE-LS) shows better generality, robustness, and reliability.

5.3 Algorithm

MAs can be uniformly formulated in a fixed algorithmic framework, of which components can be easily altered or modified in different variants [48, 53]:

- Initialization: population, external parameters;
- Loop:
  - Global search: the new population is generated by global search method;
  - Local search: update chosen individuals using local search technique within adaptive computational budget;
- Output: best solution found and search information.

5.3.1 Framework of MA(DE-LS)

In our proposed MA(DE-LS), the main structure follows closely with the general MA framework. Our contribution lies in the implementation of the efficient EA local search operators. MA(DE-LS) uses an effective gradual learning strategy to assign computational budget to both global and local search phases. The framework of MA(DE-LS) is shown in Table 5.1.

The ideal resource allocation between global search and local search in MAs can be described as follows: when the population is distributed widely or most individuals are located far away from the global optimum, the global search should receive more computational resources. When the population converge close to the global optimum, the most suitable local search technique for the local fitness landscape should be selected to play a leading role. In MA(DE-LS), the computational budget of local search technique can be adaptively tuned during the optimization procedure. At each iteration, the best solution found by global search is recorded as f'_{Gbest}, and the global computational budget is $B_{GS} = NP$. After each iteration, the mean effect of global search ξ_{meanGS} is calculated as:

$$\xi_{meanGS} = \frac{(f'_{Gbest} - f_{best})}{B_{GS}} \tag{5.7}$$

Similarly, the mean effect of local search technique ξ_{meanLS} can by obtained by:

$$\xi_{meanLS} = \frac{(f'_{Lbest} - f'_{Gbest})}{B_{LS}} \tag{5.8}$$

Table 5.1 Procedure of MA(DE-LS)

MA(DE-LS)
Input
• Optimization task (including the criterion of determining the fitness values and the dimensionality (D)) • a termination condition
Output The best solution found
Step (0) Initialization Randomly initialize a population X_0 of population size NP. Set $t = 0$. Set local search computational budget as B_{LS} and global search computational budget $B_{GS} = NP$
Step (1) Optimization procedure • **Step (1.1) Global search** Apply DE to update X_t to X_{t+1}. Pick up the best solution x_{gbest} with fitness value f'_{Gbest} • **Step (1.2) Local search** Apply local search technique to the top individual x_{gbest} with computational budget B_{LS}. Pick up the best fitness value f'_{Lbest} • **Step (1.3)** If $f'_{Lbest} < f'_{Gbest}$, update population • **Step (1.4) Update** B_{LS}: $(f'_{Lbest} - f'_{Lbest})/B_{LS} > (f'_{Gbest} - f_{best})/B_{GS}$? $B_{LS} = B_{LS} + 30$:$B_{LS} = B_{LS} - 30$ • **Step (1.5)** Update f_{best}. Set $t = t + 1$ • **Step (1.6)** If the termination criterion is met, go to step 1.1; else go to step 2
Step (2) Terminate and output

where f'_{Lbest} is the best fitness found by local search technique. In Step 1.4, the computational budget of local search technique B_{LS} can be adaptively tuned based on the comparison of ξ_{meanGS} and ξ_{meanLS}.

5.3.2 Suboptimizers

In traditional MAs, GA serves as a common and effective choice for global search [53]. In general, any EA with good exploration ability can be used as the global search method. Since the performance of DE is highly improved on continuous optimization tasks, more and more MAs employ DE as the global search technique. This is especially true, when the MAs are designed for general purpose [48, 50]. In this chapter, we mainly focus on investigating the feasibility of utilizing the efficient EA techniques as the local search operators. For the selection of global search method, we implement an effective DE version Self-adaptive Differential Evolution (SaDE), whose good exploration ability has been experimentally verified by [60].

The popular traditional mathematical local search techniques include the steepest descent methods, the conjugate gradient methods [9], the conjugate gradient method [28, 85–92], quadratic programming [13–16, 49, 74]. These approaches accomplish a neighborhood search within very limited computational cost. Recently, an adaptive PrMA (APrMA) proposed by [48] adopts a simple evolution strategy (ES) based search technique as its local search operator, which belongs to the class of EAs. [50] developed and embedded a new crossover-based local search into an MA. Compared with the traditional local search techniques, the EA-based local search optimizers have better robustness, although they usually require more computational cost. In order to meet the requirement of local search, the adopted EAs should start working on one single individual and search its neighborhood by using mutation or sampling operators. Therefore, the ESs and EDAs with high convergence speed are the good choices for local search. For example, the covariance matrix adaptation evolution strategy (CMA-ES) designed by [25] shows particularly reliable and excellent performance for local optimization. In CMA-ES, it is stated that its convergence speed is ten times slower than the Davidon, Fletcher, and Powell Strategy (DFP) [57], but its robustness for difficult problems is far more excellent. Given an initial individual, CMA-ES performs iterative self-adaptive Gaussian-based mutation and recombination. CMA-ES has been successfully used as an advanced local search EA in [1]. Moreover, as experimentally verified by [1, 2], its global search ability can be effectively strengthened by incorporating a restart strategy. Differently, our designed algorithm uses the global search EA to guarantee a good exploration.

Wang and Li [76] proposed a self-adaptive mixed distribution based univariate EDA (MUEDA) for large scale global optimization. The fast convergence and good scalability of MUEDA has been experimentally proven in [75, 76]. However, sharing the similar idea of [1, 10], Wang and Li [75] has to reduce MUEDA's risk of being trapped by local optima by incorporating a restart strategy. This shows its potentiality of being adopted as a local search technique. Compared with CMA-ES, the execution cost of MUEDA is lower, since it avoids the complex processing needed for adaption of the covariance matrix. However, its performance on rotated problems is relatively poor. Therefore, based on different required computational costs, we can flexibly apply purposive candidate local search techniques in MA(DE-LS).

5.4 Experimental Study of Different Local Search Techniques

In this experiment, we attempt to obtain general rules of designing MA(DE-LS). To avoid the influence of global search method, we use SaDE in different MA(DE-LS) variants. The strengths and weaknesses of different local search techniques can be summarized from the experiments on diverse kinds of optimization problems.

5.4.1 Experimental Settings

The algorithms under comparison are shown as follows:

- SaDE: self-adaptive differential evolution [60];
- CMAES: covariance matrix adaptation evolution strategy [25];
- MA(DE-LS)(SaDE-CMAES): global search method: SaDE, local search techniques: CMA-ES;
- MUEDA: self-adaptive mixed distribution based univariate estimation of distribution algorithm [76].
- MA(DE-LS)(SaDE-MUEDA): global search method: SaDE, local search techniques: MUEDA;

In order to fully reveal the impacts of local search techniques, we adopt a test function suite containing 26 numerical optimization problems with different characteristics, such as unimodality, multimodality, rotation, ill-condition, mis-scale, and noise. The mathematical definitions of the test functions are summarized in Table 5.2. The first three groups contain separable problems. Test functions of group 2 and 3 have mis-scaled variables and noisy landscapes, which are much more challenging for optimization. For group 4 problems, the classical test functions are rotated by $z = M(x - o)$, where M is an orthogonal rotation matrix, to avoid local optima lying along the coordinate axes while retaining the properties of the test functions. We set the dimensions (Ds) of all problems to be 30, and the fitness evaluation size to be 10,000 $\times$ D. A run is considered to be successful if at least one solution was found during its course whose fitness value is not worse than $(fit(x^*) + 1e - 5)$.

The ranking of the algorithms is based on the success performance SP [1], which is defined as follows:

Table 5.2 Classical benchmark problems to be minimized

Group 1: Classical test functions (nonlinear, separable, scalable) ($z = x - o$)

Num	Shifted	Rotated	Problems	Objective function
fun1	$\surd$		**Shifted Sphere**	$f_1(x) = \sum_{i=1}^{n} z_i^2 + f_{bias}$
fun2	$\surd$		**Shifted Schwefel 1.2**	$f_2(x) = \sum_{i=1}^{n}(\sum_{j=1}^{i} z_j^2) + f_{bias}$
fun3	$\surd$		**Shifted Schwefel 2.22**	$f_3(x) = \sum_{i=1}^{n} \lvert z_i \rvert + \prod_{i=1}^{n} \lvert z_i \rvert + f_{bias}$
fun4	$\surd$		**Shifted Schwefel 2.21**	$f_4(x) = \max \lvert z_i \rvert + f_{bias}$
fun5	$\surd$		**Shifted Rochenbrock**	$f_5(x) = \sum_{i=1}^{D}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$
fun6	$\surd$		**Shifted Ackley**	$f_6(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}) + e - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)) + 20 + f_{bias}$
fun7	$\surd$		**Shifted Griewank**	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} cos(\frac{z_i}{\sqrt{i}}) + f_{bias}$
fun8	$\surd$		**Shifted Rastrigin**	$f_8(x) = \sum_{i=1}^{n}(z_i^2) - 10cos(2\pi z_i) + 10 + f_{bias}$
fun9	$\surd$		**Noncontinues Rastrigin**	$f_9(x) = \sum_{i=1}^{n}(y_i^2) - 10cos(2\pi y_i) + 10$, $y_{i+1} = \begin{cases} z_i, if\ \lvert z_i \rvert < 1/2 \\ round(2 * z_i)/2, otherwise, \end{cases}$
fun10	$\surd$		**Shifted Penalized 1**	$f_{10} = \frac{\pi}{30}\{10\sin^2 + \sum_{i=1}^{2} 9(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1} + (y_n - 1)^2)]$ $+ \sum_{i=1}^{30} u(x_i, 10, 100, 4)\}$, u and y are shown in [93]
fun11	$\surd$		**Shifted Penalized 2**	$f_{11} = 0.1\{\sin^2(\pi 3x_1) + \sum_{i=1}^{2} 9(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2\}$ $\cdot[1 + \sin^2(2\pi x_30)] + \sum_{i=1}^{30} u(x_i, 5, 100, 4)\}$, u and y is shown in [93]
Group 2: Test functions that are mis-scaled ($z = x - o$)				
fun12	$\surd$		**Shifted Rochenbrock100**	$f_{12}(x) = \sum_{i=1}^{D}(100((a_i z_i)^2 - (a_{i+1} z_{i+1}))^2 + ((a_i z_i) - 1)^2) + f_{bias}$, $a_i = 10^{\frac{i-1}{D-1}}$
fun13	$\surd$		**Shifted Rastrigin10**	$f_{13}(x) = \sum_{i=1}^{n}((a_i z_i)^2) - 10cos(2\pi(a_i z_i)) + 10 + f_{bias}$, $a_i = 10^{\frac{i-1}{D-1}}$
fun14	$\surd$		**Shifted Rastrigin1000**	$f_{14}(x) = \sum_{i=1}^{n}((a_i z_i)^2) - 10cos(2\pi(a_i z_i)) + 10 + f_{bias}$, $a_i = 1000^{\frac{i-1}{D-1}}$

(continued)

Table 5.2 (continued)

Num	Shifted	Rotated	Problems	Objective function
Group 1: Classical test functions (nonlinear, separable, scalable) ($z = x - o$)				
Group 3: Test function with noise ($z = x - o$)				
fun15	√		**Noise Schwefel 1.2**	$f_{15}(x) = (\sum_{i=1}^{n}(\sum_{j=1}^{i} z_j^2))(1 + 0.4\|N(0,1)\|) + f_{bias}$
Group 4: Rotated test functions (nonlinear, nonseparable, scalable) ($z = M(x - o)$)				
Num	Shifted	Rotated	Problems	Objective function
fun16	√	√	**Rotated Sphere**	$f_{16}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}$
fun17	√	√	**Rotated Tablet**	$f_{17}(x) = (1000x_1)^2 + \sum_{i=2}^{D} z_i^2 + f_{bias}$
fun18	√	√	**Rotated Ellipse**	$f_{18}(x) = \sum_{i=1}^{D}(20^{\frac{i-1}{D-1}} z_i)^2 + f_{bias}$
fun19	√	√	**Rotated diff pow**	$f_{19}(x) = \sum_{i=1}^{D} z_i^{2+a_i} + f_{bias}$, $a_i = 10^{\frac{i-1}{D-1}}$
fun20	√	√	**Rotated Schwefel 2.21**	$f_{20}(x) = \max \|z_i\| + f_{bias}$
fun21	√	√	**Rotated Rochenbrock**	$f_{21}(x) = \sum_{i=1}^{D}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$
fun22	√	√	**Rotated Ackley**	$f_{22}(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}) + e - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)) + 20 + f_{bias}$
fun23	√	√	**Rotated Griewank**	$f_{23}(x) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} cos(\frac{z_i}{\sqrt{i}}) + f_{bias}$
fun24	√	√	**Rotated Rastrigin**	$f_{24}(x) = \sum_{i=1}^{n}(z_i^2) - 10cos(2\pi z_i) + 10 + f_{bias}$
fun25	√	√	**Noise Schwefel 1.2**	$f_{25}(x) = (\sum_{i=1}^{n}(\sum_{j=1}^{i} z_j^2))(1 + 0.4\|N(0,1)\|) + f_{bias}$
fun26	√	√	**Noise Quadric**	$f_{26}(x) = \sum_{i=1}^{n} i z_i^4 + random[0,1) + f_{bias}$

$$SP = mean(\text{function evaluations of successful runs}) \times \frac{\text{all runs}}{\text{number of successful runs}} \quad (5.9)$$

Small values of SP are preferable. As shown in [60, 73, 80], the empirical cumulative distribution of normalized SP can strongly benchmark the performance of multiple algorithms. The overall performance on a suite of test problems can be described in the empirical distribution of the normalized success performance picture, in which small values of normalized SP and large values of the empirical distribution are preferable. As indicated by [60], the first one that reaches the top of the graph will be considered as the best algorithm.

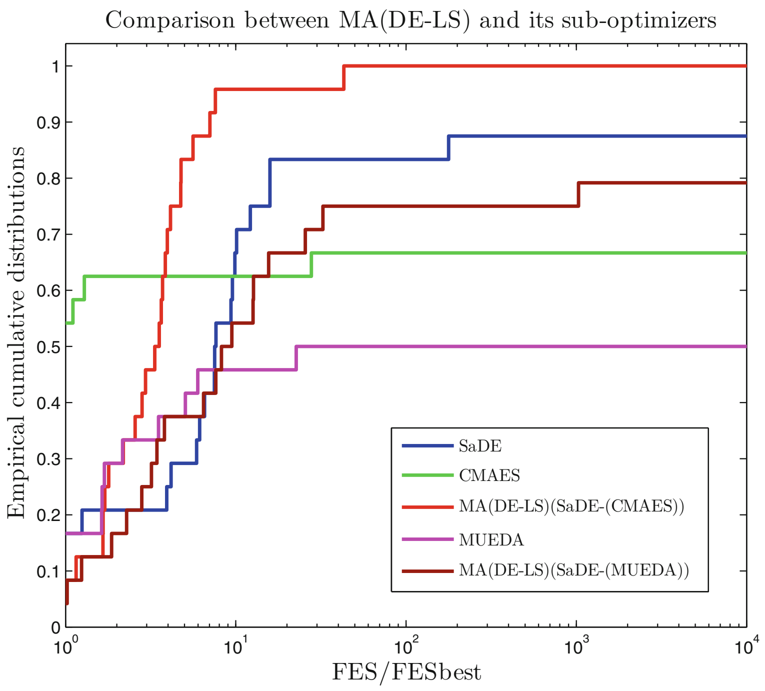


Fig. 5.1 Empirical distribution of normalized success performance of MA(DE-LS) and its suboptimizers on 26 function optimization tasks

5.4.2 Experimental Result and Discussion

The empirical distribution of the normalized success performance picture is shown in Fig. 5.1. It can be easily observed that CMA-ES provides top convergence speed for over 50 % problems. However, for over 30 % problems, it fails completely. The other local search technique MUEDA, which is mainly for large scale global optimization, shows similar properties, but poorer performance than CMA-ES. The reason is that MUEDA treats the variables separately, which inevitably results in poor performance on rotated problems.

The global search method SaDE used in MA(DE-LS) shows relatively reliable performance, which can solve almost 90 % problems. However, its efficiency is not satisfactory, see Table 5.1. The performance of two MA(DE-LS) versions is distinctly different. MA(DE-LS)(SaDE-CMAES) shows top overall performance. With the help of local search technique CMAES, it even can solve more problems than using SaDE only. Contrary, the performance of MA(DE-LS)(SaDE-MUEDA) is slightly worse than SaDE. It is not surprising, because MA(DE-LS) has to use some computational cost to accomplish the local search computational budget adaption. This is especially true for the rotated problems.

Based on this comparison study, we have the following observations:

- Using local search method alone is not sufficient for reliably solving diverse tasks. This is also the main reason for applying restart strategy in [1, 76].
- Using the same global search method, different local search techniques may result in remarkably different performance. A bad choice of the local search technique may even result in poorer performance in both effectiveness and efficiency than the global search.
- With the help of suitable local search technique, MA(DE-LS) can not only enhance the efficiency like the other MAs, but strengthen the effectiveness of the global search method.

5.5 Experimental Comparison with State-of-the-Art EAs & MAs

In order to comprehensively benchmark MA(DE-LS), we design three experimental studies in this section. The merits and demerits of MA(DE-LS) can be clearly shown via comparison with diverse State-of-the-art EAs, DE-based MAs and other MAs.

5.5.1 Comparison with State-of-the-Art Global Search EAs

In order to show the superiority of MA(DE-LS) over the effective global search methods, we compared it with six recently proposed state-of-the-art EAs:

- SaDE: self-adaptive differential evolution [60];
- jDE: differential evolution with parameter adaption [3].
- SLPSO: self-adaptive learning based particle swarm optimization [80];
- CLPSO: comprehensive learning particle swarm optimizer [38];
- FIPS-PSO: fully informed PSO [43];
- FDR-PSO: Fitness-distance-ratio based PSO [56];

Specifically, the SLPSO and SaDE self-adaptively utilize offspring generating strategies from PSO and DE respectively to strengthen the robustness. CLPSO has exhibited the top capability of handling multimodal problems [38] among the PSO variants.

Figure 5.2 depicts the comparison between MA(DE-LS) and six global search EAs. The superior capacity in universality of SLPSO and SaDE compared the other PSO and DE variants has been experimentally confirmed in [60, 80]. It is observed that they have similar empirical cumulative distribution curves and significantly outperform jDE and the other PSOs. Compared with SLPSO and SaDE, MA(DE-LS) has significantly better convergence speed and universality. For almost 60 % problems, MA(DE-LS) provides the fastest optimization speed. Furthermore, when its empirical cumulative distribution curve reaches the top, the normalized *SP* value

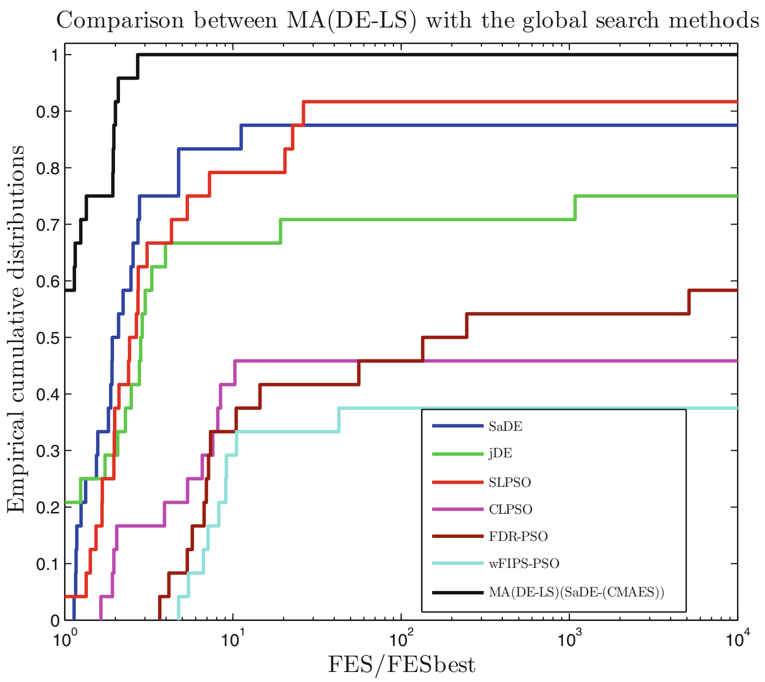


Fig. 5.2 Compared with state-of-the-art global EAs on 26 function optimization tasks

is still very small (≤2). In Fig. 5.3, the comparison of optimization curves on six representative problems again confirms this viewpoint. Therefore, the advantages of the MA(DE-LS) compared with effective global search EAs are definitely verified.

5.5.2 Comparison with State-of-the-Art DE-Based MAs

In order to show the superiorities of MA(DE-LS) over the other DE-based MAs, we compare MA(DE-LS) with four DE-based MAs. Besides, the classical DE is adopted to provided comparative results. The algorithms under comparison are as follows:

- DE: mutation parameter $F = 0.9$ crossover parameter $Cr = 0.1$ for separate problems and $Cr = 0.9$ for separate problems [61].
- DEahcSPX: DE with the adaptive hill-climbing local search and simplex crossover (SPX) operation [50];
- DEfirSPX: DE with the adaptive hill-climbing local search with length adaption and (SPX) operation [50];
- DExhcSPX: DE with the adaptive crossover-based local search and simplex crossover (SPX) operation [50];
- SaDE-L: self-adaptive differential evolution with local search DFP [59].

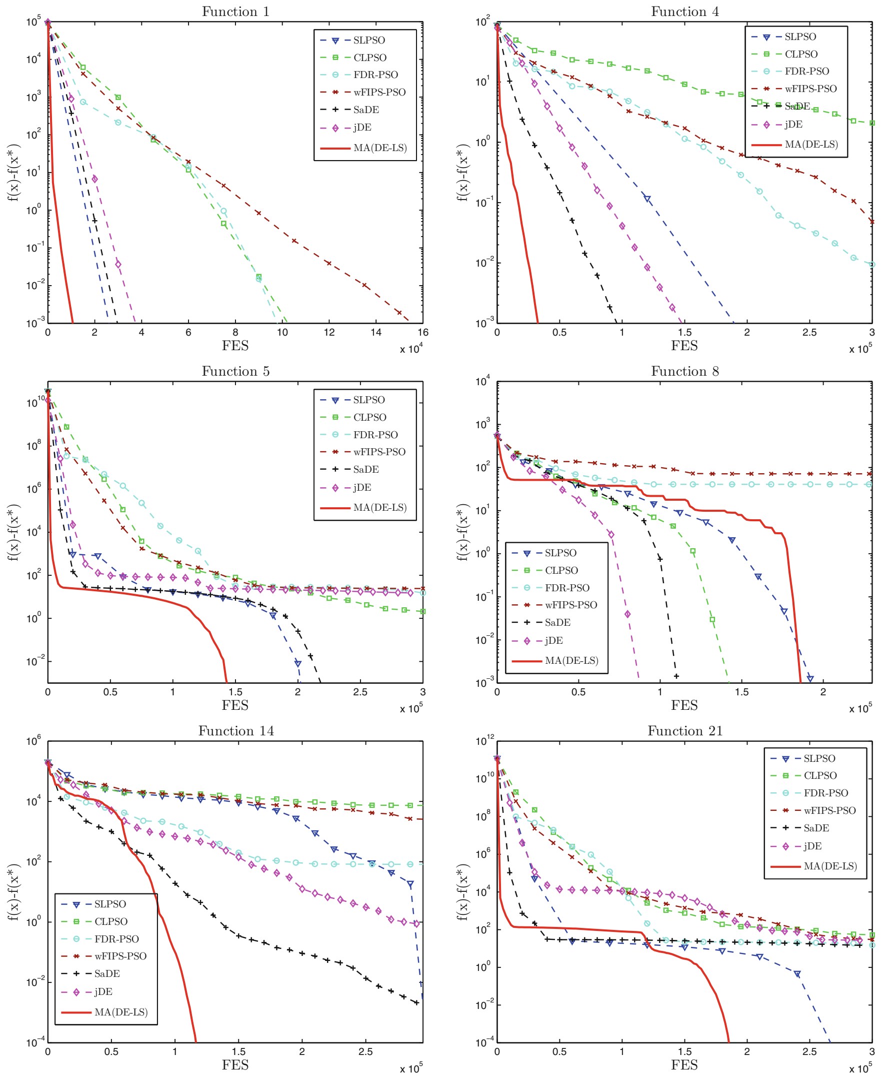


Fig. 5.3 Optimization curves of function optimization with 30 D. (Sphere, Schwefel 2.21, Rochenbrock, Rastrigin, Noise Schewefel 1.2, and Rotated Rochenbrock)

The reasons of selecting these algorithms are: (1) the convergence speed of three SPX-based DEs is accelerated by different kinds of local search techniques. (2) SaDE-L performs DFP, a mathematical exact search technique, after a fixed interval.

Due to diverse characteristics, the CEC05 competition function suite has been widely used to benchmark different optimization algorithms [73]. As required in [2], we set the maximum number of function evaluations is set to be $10{,}000 \times D$. For all

Table 5.3 Comparison with State-of-the-art DE-based MAs on CEC05 functions

	Fcec1	Fcec2
DE	0.00E+00 ± 0.00E+00	5.49E−08 ± 1.20E−07
DEahcSPX	0.00E+00 ± 0.00E+00	6.52E−05 ± 4.84E−05
DEfirSPX	0.00E+00 ± 0.00E+00	1.05E−03 ± 1.29E−03
DExhcSPX	0.00E+00 ± 0.00E+00	9.40E−04 ± 1.80E−03
SaDE−L	0.00E+00 ± 0.00E+00	9.71E−08 ± 4.86E−07
SaDE−(CMAES)	0.00E+00 ± 0.00E+00	**0.00E+00 ± 0.00E+00**
	Fcec3	Fcec4
DE	2.89E+05 ± 1.93E+05	**5.04E−01 ± 8.58E−01**
DEahcSPX	1.29E+06 ± 9.22E+05	4.62E+00 ± 8.78E+00
DEfirSPX	1.73E+06 ± 1.22E+06	1.04E+01 ± 1.75E+01
DExhcSPX	1.54E+06 ± 1.15E+06	6.69E+00 ± 1.06E+01
SaDE−L	5.05E+04 ± 1.58E+05	5.82E−06 ± 1.45E−05
SaDE−(CMAES)	**0.00E+00 ± 0.00E+00**	3.79E+00 ± 9.11E+00
	Fcec5	Fcec6
DE	2.35E+02 ± 1.83E+02	3.77E+00 ± 2.71E+00
DEahcSPX	9.00E+02 ± 4.79E+02	3.84E+00 ± 3.75E+00
DEfirSPX	1.15E+03 ± 6.68E+02	1.65E+01 ± 4.72E+01
DExhcSPX	1.01E+03 ± 4.31E+02	1.41E+01 ± 1.86E+01
SaDE−L	7.88E+02 ± 1.24E+03	2.12E+01 ± 1.34E+01
SaDE−(CMAES)	**2.78E+01 ± 2.11E|+01**	**4.86E−01 ± 1.32E+00**
	Fcec7	Fcec8
DE	9.65E−01 ± 9.14E−02	2.09E+01 ± 6.25E−02
DEahcSPX	7.39E−03 ± 6.32E−03	2.09E+01 ± 1.12E−01
DEfirSPX	**4.53E−03 ± 6.92E−03**	2.10E+01 ± 4.61E−02
DExhcSPX	7.98E−03 ± 9.48E−03	2.09E+01 ± 7.41E−02
SaDE−L	8.27E−03 ± 1.14E−02	**2.01E+01 ± 5.73E−02**
SaDE-(CMAES)	9.09E−03 ± 3.54E−03	2.04E+01 ± 4.65E−01
	Fcec9	Fcec10
DE	**0.00E+00 ± 0.00E+00**	6.16E+01 ± 4.56E+01
DEahcSPX	2.04E+01 ± 8.19E+00	5.27E+01 ± 4.84E+01
DEfirSPX	2.47E+01 ± 7.72E+01	6.96E+01 ± 5.39E+01
DExhcSPX	2.80E+01 ± 7.75E+00	6.79E+01 ± 4.80E+01
SaDE-L	2.27E−15 ± 1.14E−14	3.58E+01 ± 6.08E+00
SaDE-(CMAES)	**0.00E+00 ± 0.00E+00**	**3.56E+01 ± 1.11E+01**

test functions, the algorithms carry out 25 independent runs on 30*D* problems. The mean values and standard deviation values are summarized in Table 5.3.

For the unimodal problems Fcec1–5, MA(DE-LS) can provide top performance except for Fcec4, SaDE-L is better. It is apparent that the effective local search technique used in MA(DE-LS) plays a leading role. For the other multimodal problems, it is interesting to see that the performance of MA(DE-LS) still keeps on a high level. Only for Fcec7 and Fcec8, MA(DE-LS) is slightly worse than the best results. In summary of these, we can conclude that MA(DE-LS) shows better reliability than the other DE-based MAs.

5.5.3 Comparison with State-of-the-art MAs

To demonstrate the advantages of MA(DE-LS) over the other MAs, we compare MA(DE-LS) with three MA versions from the IEEE Congress on Evolutionary Computation 2005 (CEC05) function optimization competition:

- SaDE-L: self-adaptive differential evolution with local search DFP [59];
- DMS-L-PSO: dynamic multi-swarm particle swarm optimizer with local search [37];
- BLX-MA: real-coded memetic algorithm with adaptive local search probability and local search length [44].

Besides these three MAs, another recently proposed MA for general problems and a related EA are also adopted:

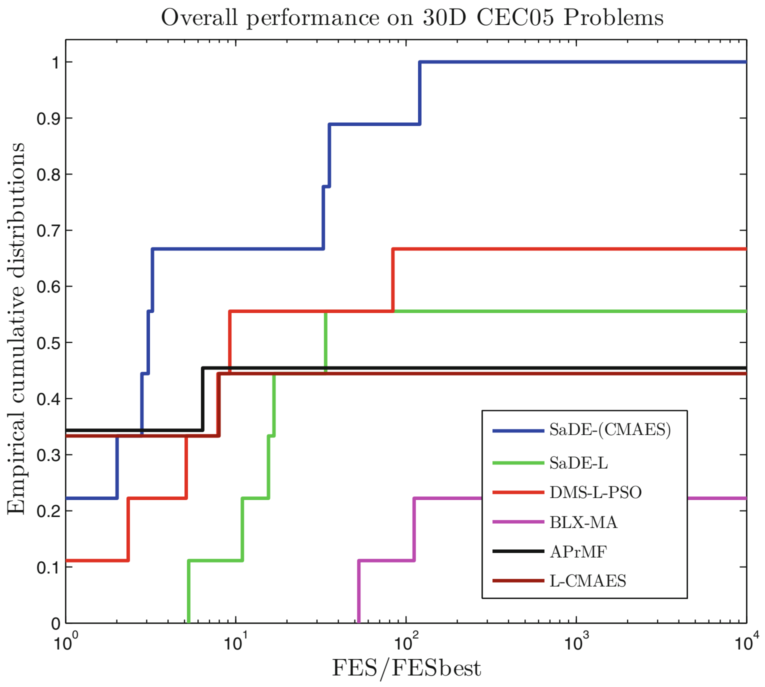


Fig. 5.4 Compared with state-of-the-art MAs on 16 CEC05 problems

- APrMA: adaptive probabilistic memetic algorithm [48];
- L-CMA-ES: restart CMA-ES [1];

In general, they have different strengths. APrMA is proposed for general problems, and has shown distinct progress compared with other MAs. The essential of L-CMA-ES is to restart local CMA-ES versions when the search fails. Therefore, L-CMA-ES is made of multiple runs of CMA-ES on many problems.

In this experiment, a run is considered to be successful if at least one solution was discovered during its course whose fitness value is not worse than ($\mathit{fit}(x^*) + 1e - 6$) for the functions 1–5 and ($\mathit{fit}(x^*) + 1e - 2$) for the other functions. The overall performance on the first 16 problems in empirical accumulative distribution of normalized SP is described in Fig. 5.4.

From Fig. 5.4, we can observe that MA(DE-LS) is definitely the best, because of the higher optimization speed and the higher number of solved problems. For the other algorithms, the performance of APrMA, SaDE-L, DMS-L-PSO and L-CMA-ES is similar and cannot solve more than 70 % of the problems that can be solved by MA(DE-LS).

5.6 Experimental Study on Digital IIR Filter Design with Fixed-Point Representation

In order to provide comprehensive impression of the utility of our MA(DE-LS), we adopt twelve test problems belonging to four types of digital IIR filters with different settings, including low-pass (LP), high-pass (HP), band-pass (BP), and band-stop (BS). The detailed settings of the test problems are summarized in Table 5.4.

Table 5.4 Settings of the test problems

Problem	Type	Passband	Stopband	δ_1 (dB)	δ_2 (dB)	Order
1	LP	$[0\ 0.5\pi]$	$[0.6\pi\ \pi]$	1	80	9
2	LP	$[0\ 0.5\pi]$	$[0.6\pi\ \pi]$	1	110	11
3	LP	$[0\ 0.5\pi]$	$[0.6\pi\ \pi]$	1	140	13
4	HP	$[0.6\pi\ \pi]$	$[0\ 0.5\pi]$	1	80	8
5	HP	$[0.6\pi\ \pi]$	$[0\ 0.5\pi]$	1	110	11
6	HP	$[0.6\pi\ \pi]$	$[0\ 0.5\pi]$	1	140	13
7	BP	$[0.4\pi\ 0.6\pi]$	$[0\ 0.3\pi] \bigcup [0.7\pi]$	1	55	12
8	BP	$[0.4\pi\ 0.6\pi]$	$[0\ 0.3\pi] \bigcup [0.7\pi]$	1	70	14
9	BP	$[0.4\pi\ 0.6\pi]$	$[0\ 0.3\pi] \bigcup [0.7\pi]$	1	90	16
10	BS	$[0\ 0.25\pi] \bigcup [0.75\pi\ \pi]$	$[0.4\pi\ 0.6\pi]$	1	55	9
11	BS	$[0\ 0.25\pi] \bigcup [0.75\pi\ \pi]$	$[0.4\pi\ 0.6\pi]$	1	70	13
12	BS	$[0\ 0.25\pi] \bigcup [0.75\pi\ \pi]$	$[0.4\pi\ 0.6\pi]$	1	90	14

The number of variables to be optimized is $2\times$order

For the digital IIR filters represented by Eq. (5.1), we restrict all of the parameters in $[-2, 2]$, but most of the parameters locate in $[-1, 1]$. Therefore, for a word length of n_L, when the parameter is in $[-1, 1)$, we use $n_L - 1$ bits to represent the decimal part and 1 sign bit. For the parameters with values in $[-2, -1)\bigcup[1, 2)$, we use 1 bit to represent the integer part, $(n_L - 2)$ bits to represent the decimal part and 1 sign bit. In this experiment, the coefficients of all filters are quantized with a word length of 16 bits. To verify the superiors of MA(DE-LS), we compare its performance with those of five state-of-the-art EAs. The algorithms for comparison are listed as follows:

- SaDE: self-adaptive differential evolution [60].
- jDE: self-adaptive control parameters in differential evolution [3].
- PSO-cf-local: local version of particle swarm optimization with constriction factor [33].
- CLPSO: comprehensive learning particle swarm optimizer [38].
- MUEDA: self-adaptive mixed distribution based univariate estimation of distribution algorithm [76].

These algorithms have different strengths: CLPSO proposed by [38] has shown remarkable capability in handling multimodal problems amongst various PSO variants; The convergence speed of PSO-cf and MUEDA are very high, and they perform well on unimodal problems; SaDE and jDE have very good capability and generality of handling diverse problems.

5.6.1 Experimental Results

We set the maximum number of function evaluations to be $10{,}000 \times D$, where D is number of variables to be optimized. The parameter settings for all algorithms are inherited from the referenced papers. For all test problems, we carry out 30 independent runs for each algorithm. Statistics gathered from the results are summarized in Table 5.5. The mean and standard deviation (std) and the number of successful runs (Sruns) are recorded. A run is considered to be successful if at least one solution was discovered during its course whose fitness value is 0. Moreover, when all 30 runs are successful, the average number of function evaluations (NFE) required to find the global optima are also recorded.

5.6.2 Discussion

The problems 1–3 belong to digital LP IIR filter design. These three problems are relatively easy, compared with the other kinds of problems. On the first two problems, MA(DE-LS) can provide 30 successful runs, followed by SaDE, which uses more computational cost. On the third problem, none of algorithms can provide 100 % succeed rate, but MA(DE-LS) also performs the best in terms of NFE. For the other

Table 5.5 Filter performance comparison on four types of fixed-point digital IIR filters

Problem 1	mean	std	Sruns	NFE	Problem 2	mean	std	Sruns	NFE
MA(DE-LS)	0.00E+00	0.00E+00	30	18,333	MA(DE-LS)	0.00E+00	0.00E+00	30	470,133
SaDE	0.00E+00	0.00E+00	30	50,142	SaDE	0.00E+00	0.00E+00	30	103,441
jDE	3.34E−01	9.06E−01	24	–	jDE	6.91E+00	7.48E+00	5	–
PSO-cf	1.23E+02	1.22E+02	0	–	PSO-cf	3.00E+02	1.89E+02	0	–
CLPSO	8.14E+00	4.85E+00	1	–	CLPSO	3.61E+01	9.84E+00	0	–
MUEDA	4.42E−02	2.42E−01	29	–	MUEDA	4.38E−01	2.32E+00	27	–
Problem 3	mean	std	Sruns	NFE	Problem 4	mean	std	Sruns	NFE
MA(DE-LS)	1.84E+00	5.51E−00	25	–	MA(DE-LS)	0.00E+00	0.00E+00	30	17,819
SaDE	9.38E+00	2.00E+01	24	–	SaDE	1.86E+00	2.99E+00	18	–
jDE	3.66E+01	9.63E+00	0	–	jDE	4.30E+00	3.11E+00	4	–
PSO-cf	4.32E+02	2.25E+02	0	–	PSO-cf	1.23E+02	1.14E+02	1	–
CLPSO	7.81E+01	2.02E+01	0	–	CLPSO	1.02E+01	4.23E+00	0	–
MUEDA	3.88E+00	6.46E+00	14	–	MUEDA	2.15E+00	3.86E+00	11	–
Problem 5	mean	std	Sruns	NFE	Problem 6	mean	std	Sruns	NFE
MA(DE-LS)	0.00E+00	0.00E+00	30	54,211	MA(DE-LS)	0.00E+00	0.00E+00	30	162,880
SaDE	1.57E−01	5.19E−01	23	–	SaDE	7.93E+00	9.05E+00	13	–
jDE	1.66E+00	1.82E+00	5	–	jDE	1.14E+01	6.55E+00	1	–
PSO-cf	2.27E+02	1.58E+02	1	–	PSO-cf	4.38E+02	2.19E+02	0	–
CLPSO	2.08E+00	7.77E+00	3	–	CLPSO	14.35697	3.819005	0	–
MUEDA	0.00E+00	0.00E+00	30	18,213	MUEDA	0.00E+00	0.00E+00	30	37,414

(continued)

Table 5.5 (continued)

Problem 7	mean	std	Sruns	NFE	Problem 8	mean	std	Sruns	NFE
MA(DE-LS)	0.00E+00	0.00E+00	30	45,620	MA(DE-LS)	1.25E−01	2.17E−02	27	87,169
SaDE	9.17E−01	1.27E+00	12	–	SaDE	2.29E+00	3.05E+00	11	–
jDE	2.59E+00	1.56E+00	1	–	jDE	8.33E+00	2.45E+00	0	–
PSO-cf	1.84E+01	3.52E+01	15	–	PSO-cf	5.31E+01	8.32E+01	9	–
CLPSO	4.57E−01	2.93E+00	6	–	CLPSO	8.229372	1.888515	0	–
MUEDA	1.21E+01	1.70E+00	0	–	MUEDA	2.88E+01	3.46E+00	0	–
Problem 9	mean	std	Sruns	NFE	Problem 10	mean	std	Sruns	NFE
MA(DE-LS)	0.00E+00	0.00E+00	30	71,690	MA(DE-LS)	0.00E+00	0.00E+00	30	295,546
SaDE	1.47E+01	7.92E+00	5	–	SaDE	5.05E+00	6.18E+00	13	–
jDE	2.23E+01	4.40E+00	0	–	jDE	3.50E+01	5.51E+01	4	–
PSO-cf	7.56E+01	8.60E+01	3	–	PSO-cf	1.99E+02	1.59E+02	4	–
CLPSO	9.574261	7.411346	0	–	CLPSO	8.53E+00	9.09E+00	0	–
MUEDA	7.10E+01	5.11E+00	0	–	MUEDA	2.19E+02	1.03E+02	2	–
Problem 11	mean	std	Sruns	NFE	Problem 12	mean	std	Sruns	NFE
MA(DE-LS)	4.26E+00	6.14E+01	24	–	MA(DE-LS)	1.87E+01	3.81E+01	23	–
SaDE	2.35E+01	2.40E+01	9	–	SaDE	4.43E+01	2.77E+01	3	–
jDE	5.03E+01	2.61E+01	0	–	jDE	6.13E+01	2.38E+01	0	–
PSO-cf	1.59E+02	1.60E+02	7	–	PSO-cf	2.75E+02	2.64E+02	1	–
CLPSO	7.11E+00	7.20E+00	2	–	CLPSO	2.41E+01	1.33E+01	0	–
MUEDA	2.83E+01	6.13E+01	23	–	MUEDA	5.59E+01	1.23E+02	24	–

three algorithms, the success rate is very limited. The experimental results on the HP problems 4–6 is similar to the first three ones, on which MA(DE-LS) also provide the top performance. Instead of SaDE, MUEDA now obtains the second place. The reason may be that the HP problems require more exploitation ability, which is the most important strength of MUEDA. It is presented by [33] that the convergence speed of PSO-cf is very high, but its performance on these problems is very poor (see Table 5.5). This is because the landscapes of these problems are full of local optima, which inevitably impede the search of the particles, while the mixed distribution used in MUEDA is able to effectively avoid this ([76]).

On the BP problems 7–9, only MA(DE-LS) can succeed in finding the desirable filters in every run, while the other algorithms show poor performance. The reason is MA(DE-LS) can balance the exploration and exploitation well, and provide effective and efficient performance on different kinds problems. The three BS problems are the most difficult in this test suite. On these problems, only MA(DE-LS) can provide satisfactory performance, although it requires significantly more NFS than in the above problems. On the BP problems, MUEDA, which shows effective performance on the LP and HP problems, completely fails. Therefore, the effect of the two-stage framework is clearly verified. Besides, the success rates of SaDE on BP problems decrease to lower than 50 %. The difficulty level of BS problems is also very high. Even MA(DE-LS) fails in some runs on problems 11 and 12, on which MUEDA provides comparable results.

In summary, MA(DE-LS) achieves the best performance on all test problems in terms of both effectiveness and efficiency.

5.7 Conclusion

In the previous research, DE has been frequently used as the global search method in MAs. However, because of the limited performance of the conventional local search operators, the performance of previous DE-related MAs still needs further improvement. In this chapter, we investigate the feasibility of using efficient EAs as the local search techniques in an adaptive MA framework, which can fully extract the strengths of both global and local search techniques. Two MA(DE-LS) variants are generated by applying different local search techniques, including CMAES and MUEDA. The impacts of local search techniques have been experimentally revealed. Some interesting observations are obtained, which is very useful for designing DE-related algorithms.

In order to comprehensively show the effectiveness and efficiency of MA(DE-LS), we experimentally compare MA(DE-LS) with state-of-the-art EAs, DE-based MAs, and other MAs. The superior performance of MA(DE-LS) in terms of efficiency, effectiveness, and reliability have been clearly verified. In order to show its practical applicability, we apply MA(DE-LS) to digital IIR filter design with fixed-point representation. The experimental results definitely verify the promising performance of MA(DE-LS).

Due to MA(DE-LS)'s promising performance, the research direction of applying efficient EAs as the local search operators deserves more attention from both academic and engineering communities.

References

1. Auger, A., Hansen, N.: Performance evaluation of an advanced local search evolutionary algorithm. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1777–1784 (2005)
2. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1769–1776 (2005)
3. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)
4. Caponio, A., Kononova, A.V., Neri, F.: Differential evolution with scale factor local search for large scale problems. In: Tenne, Y., Goh, C.-K. (eds.) Computational Intelligence in Expensive Optimization Problems. Studies in Evolutionary Learning and Optimization, vol. 2, pp. 297–323. Springer, Heidelberg (2010)
5. Chen, C.T.: One-Dimensional Digital Signal Processing. Marcel Dekker, New York (1979)
6. Choo, H., Muhammad, K., Roy, K.: Complexity reduction of digital filters using shift inclusive differential coefficients. IEEE Trans. Signal Process. **52**(6), 1760–1772 (2004)
7. Dai, C.H., Chen, W.R., Zhu, Y.F.: Seeker optimization algorithm for digital IIR filter design. IEEE Trans. Ind. Electron **57**(5), 1710–1718 (2010)
8. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
9. Debye, P.: Näherungsformeln für die Zylinderfunktionen für groβe Werte des Arguments und unbeschränkt veränderliche Werte des Index. Mathematische Annalen **67**(4), 535–558 (1909)
10. Dong, W., Yao, X.: Covariance matrix repairing in Gaussian based EDAs. In: IEEE Congress on Evolutionary Computation (CEC07), pp. 415–422 (2007)
11. Dong, W., Yao, X.: Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms. Inf. Sci. **178**(15), 3000–3023 (2008)
12. Dong, W., Yao, X.: NichingEDA: utilizing the diversity inside a population of EDAs for continuous optimization. In: IEEE Congress on Evolutionary Computation, pp. 1260–1267 (2008)
13. Dong, W., Zhang, X., Jiang, Z., Sun, W., Xie, L., Hampapur, A.: Detect irregularly shaped spatio-temporal clusters for decision support. In: IEEE International Conference on Service Operations, Logistics, and Informatics, pp. 231–236 (2011)
14. Dong, W., Zhang, X., Li, L., Sun, C., Shi, L., Sun, W.: Detecting irregularly shaped significant spatial and spatio-temporal clusters. SIAM Data Mining, pp. 732–743 (2012)
15. Dong, W., Li, L., Zhou, C., Wang, Y., Li, M., Tian, C., Sun, W.: Discovery of generalized spatial association rules. In: IEEE International Conference on Service Operations, Logistics, and Informatics, pp. 60–65 (2012)
16. Dong, W., Fan, W., Shi, L., Zhou, C., Yan, X.: A general framework to encode heterogeneous information sources for contextual pattern mining. In: ACM International Conference on Information and Knowledge Management, pp. 65–74 (2012)
17. Dong, W., Chen, T., Tino, P., Yao, X.: Scaling up estimation of distribution algorithms for continuous optimization. IEEE Trans. Evol. Comput. **17**(6), 797–822 (2013)
18. Elloumi, S., Fortemps, P.: A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. Eur. J. Oper. Res. **205**, 31–41 (2010)
19. Etter, D.M., Hicks, M.J., Cho, K.H.: Recursive adaptive filter design using an adaptive genetic algorithm. In: Proceedings of IEEE International Conference on ASSP, pp. 635–638 (1982)

20. Florios, K., Mavrotas, G., Diakoulaki, D.: Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. Eur. J. Oper. Res. **203**, 14–21 (2010)
21. Garcia-Najera, A., Bullinaria, J.A.: An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. **38**(1), 287–300 (2011)
22. Hanne, T.: A multiobjective evolutionary algorithm for approximating the efficient set. Eur. J. Oper. Res. **176**, 1723–1734 (2007)
23. Hanne, T., Nickel, S.: A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. Eur. J. Oper. Res. **167**, 663–678 (2005)
24. Harris, S.P., Ifeachor, E.C.: Automatic design of frequency sampling filters by hybrid genetic algorithm techniques. IEEE Trans. Signal Process. **46**(12), 3304–3314 (1998)
25. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
26. Hasan, S., Sarker, R., Essam, D., Cornforth, D.: Memetic algorithms for solving job-shop scheduling problems. Memet. Comput. **1**(1), 69–83 (2009)
27. Haseyama, M., Matsuura, D.: A filter coefficient quantization method with genetic algorithm, including simulated annealing. IEEE Signal Process. Lett. **13**(4), 189–192 (2006)
28. Hestenes, R.M., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl Bur. Stand. **49**(6), 409–436 (1952)
29. Houck, C.: Joines, J., Kay, M.: Utilizing Lamarckian evolution and the Baldwin effect in hybrid genetic algorithms. NCSU-IE Technical Report 96-01, Meta-Heuristic Research and Applications Group, Department of Industrial Engineering, North Carolina State University (1996)
30. Kalinli, A., Karaboga, N.: Artificial immune algorithm for IIR filter design. J. Eng. Appl. Artif. Intell. **18**(5), 919–929 (2005)
31. Kalinli, A., Karaboga, N.: A new method for adaptive IIR filter design based on Tabu search algorithm. Int. J. Electron. Commun. **59**(2), 111–117 (2005)
32. Karaboga, N., Kalinli, A., Karaboga, D.: Designing IIR filters using ant colony optimisation algorithm. J. Eng. Appl. Artif. Intell. **17**(3), 301–309 (2004)
33. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1671–1676 (2002)
34. Kim, Y.K., Park, K., Ko, J.: A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput. Oper. Res. **30**(8), 1151–1171 (2003)
35. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans. Evol. Comput. **9**(5), 474–488 (2005)
36. Krusienski, D.J., Jenkins, W.K.: Design and performance of adaptive systems based on structured stochastic optimization. IEEE Circuits Syst. Mag. **5**(1), 8–20 (2005)
37. Liang, J.J., Suganthan, P.N.: Dynamic multi-swarm particle swarm optimizer with local search. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 522–528 (2005)
38. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
39. Liao, T.W.: Two hybrid differential evolution algorithms for engineering design optimization. Appl. Soft Comput. **10**(4), 1188–1199 (2010)
40. Lozano, M., García-Martínez, C.: Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. Comput. Oper. Res. **37**(3), 481–497 (2010)
41. Lu, Y.L., Zhou, J.Z., Qin, H., Li, Y.H., Zhang, Y.C.: An adaptive hybrid differential evolution algorithm for dynamic economic dispatch with valve-point effects. Expert Syst. Appl. **37**(7), 4842–4849 (2010)
42. Lu, Y.L., Zhou, J.Z., Qin, H., Li, Y.H., Zhang, Y.C.: An adaptive chaotic differential evolution for the short-term hydrothermal generation scheduling problem. Energy Convers. Manag. **51**(7), 1481–1490 (2010)

43. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Trans. Evol. Comput. **8**(3), 204–210 (2004)
44. Molina, D., Herrera, F., Lozano, M.: Adaptive local search parameters for real-coded memetic algorithms. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 888–895 (2005)
45. Moscato, P. : Genetic algorithms and martial arts: towards memetic algorithms. Publication Report 790, Caltech Concurrent Computation Program (1989)
46. Neri, F., Tirronen, V.: Recent advances in differential evolution: a review and experimental analysis. Artif. Intell. Rev. **33**(1), 61–106 (2010)
47. Neumann, F.: Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. Comput. Oper. Res. **35**(9), 2750–2759 (2008)
48. Nguyen, Q.H., Ong, Y.S., Lim, M.H.: A probabilistic memetic framework. IEEE Trans. Evol. Comput. **13**(3), 604–623 (2009)
49. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Berlin (2006)
50. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Trans. Evol. Comput. **12**(1), 107–125 (2008)
51. Ong, Y.S., Keane, A.J.: Meta-Lamarckian learning in memetic algorithms. IEEE Trans. Evol. Comput. **8**(2), 99–110 (2004)
52. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Trans. Syst. Man Cybern. Part B **36**(1), 141–152 (2006)
53. Ong, Y.S., Lim, M.H., Chen, X.S.: Research frontier: memetic computation—past, present & future. IEEE Comput. Intell. Mag. **5**(2), 24–36 (2010)
54. Pan, S.T.: A canonic-signed-digit coded genetic algorithm for designing finite impulse response digital filter. Digit. Signal Process. **20**(314–327), 2010 (2010)
55. Pena, J.M., Robles, V., Larranaga, P., Herves, V., Rosales, F., Perez, M.S.: GA-EDA: hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. Proc. Lect. Notes Comput. Sci. **3029**, 361–371 (2004)
56. Peram, T., Veeramachaneni, K., Mohan, C.K.: Fitness-distance-ratio based particle swarm optimization. In: Proceedings of Swarm Intelligence Symposium, p. 174 (2003)
57. Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. Comput. J. **7**(4), 303–307 (1964)
58. Prodhon, C.: A hybrid evolutionary algorithm for the periodic location-routing problem. Eur. J. Oper. Res. **210**, 204–212 (2011)
59. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1785–1791 (2005)
60. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
61. Rökkönen, J., Kukkonen, S., Price, K.V.: Real-parameter optimization with differential evolution. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), pp. 506–513 (2005)
62. dos Santos Coelho, L., Cocco Mariani, V.: Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect. IEEE Trans. Power Syst. **21**(2), 989–996 (2006)
63. Shah, R., Reed, P.: Comparative analysis of multiobjective evolutionary algorithms for random and correlated instances of multiobjective d-dimensional knapsack problems. Eur. J. Oper. Res. **211**, 466–479 (2011)
64. Shynk, J.J.: Adaptive IIR filtering. IEEE ASSP Mag. **6**(2), 4–21 (1989)
65. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic strategy for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
66. Sun, J., Zhang, Q.F., Tsang, E.: DE/EDA: a new evolutionary algorithm for global optimization. Inf. Sci. **169**, 249–262 (2005)
67. Tan, K.C., Chew, Y.H., Lee, L.H.: A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. Eur. J. Oper. Res. **172**, 855–885 (2006)

68. Tang, K.S., Man, K.F., Kwong, S., Liu, Z.F.: Design and optimization of IIR filter structure using hierarchical genetic algorithms. IEEE Trans. Ind. Electron **45**(3), 481–487 (1998)
69. Tarczynski, A., Cain, G.D., Hermanowicz, E., Rojewski, M.: A WISE method for designing IIR filters. IEEE Trans. Signal Process. **49**(7), 1421–1432 (2001)
70. Tsai, J.T., Chou, J.H., Liu, T.K.: Optimal design of digital IIR filters by using hybrid Taguchi genetic algorithm. IEEE Trans. Ind. Electron **53**(3), 867–879 (2006)
71. Vanuytsel, G., Boets, P., Van Biesen, L., Temmerman, S.: Efficient hybrid optimization of fixed-point cascaded IIR filter coefficients. In: Proceedings of IEEE Instrumentation and Measurement, pp. 793–797 (2002)
72. Vicini, A., Quagliarella, D.: Airfoil and wing design using hybrid optimization strategies. Am. Inst. Aeronaut. Astronaut. J. **37**(5), 634–641 (1999)
73. Vrugt, J.A., Robinson, B.A., Hyman, J.M.: Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Trans. Evol. Comput. **13**(2), 243–259 (2009)
74. Wang, R., Dong, W., Wang, Y., Tang, K., Yao, X.: Pipe failure prediction: a data mining method. In: IEEE International Conference on Data Engineering, pp. 1208–1218 (2013)
75. Wang, Y., Li, B.: A restart univariate estimation of distribution algorithm: sampling under mixed Gaussian and Levy probability distribution. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2008), pp. 3218–3925 (2008)
76. Wang, Y., Li, B.: A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimization. Studies in Computational Intelligence, pp. 171–198. Springer, New York (2009)
77. Wang, Y., Li, B., Weise, T.: Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. Inf. Sci. **180**(12), 2405–2420 (2011)
78. Wang, Y., Li, B.: Two-stage based ensemble optimization for large-scale global optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2010), pp. 1–8 (2011)
79. Wang, Y., Li, B., Chen, Y.B.: Digital IIR filter design using multi-objective optimization evolutionary algorithm. Apply Soft Comput. **11**(2), 1851–1857 (2011)
80. Wang, Y., Li, B., Weise, T., Wang, J.Y., Yuan, B., Tian, Q.J.: Self-adaptive learning based particle swarm optimization. Inf. Sci. **181**(20), 4515–4538 (2011)
81. Wang, Y., Li, B., Zhang, K.B.: Estimation of distribution and differential evolution cooperation for real-world numerical optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2011), pp. 1315–1321 (2011)
82. Wang, Y., Li, B., Weise, T.: Two-Stage ensemble memetic algorithm: function optimization and digital IIR filter design. Inf. Sci. **220**(20), 408–424 (2013)
83. Wang, Y., Huang, J., Dong, W., Yan, J., Tian, C., Li, M., Mo, W.: Two-stage based ensemble optimization framework for large-scale global optimization. Eur. J. Oper. Res. **228**(2), 308–320 (2013)
84. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
85. Yan, J., Li, Y., Zheng, E., Liu, Y.: An accelerated human motion tracking system based on voxel reconstruction under complex environments. In: Asian Conference on Computer Vision (ACCV), pp. 313–324 (2009)
86. Yan, J., Zhu, M., Liu, H., Liu, Y.: Visual saliency detection via sparsity pursuit. IEEE Signal Process. Lett. **17**(8), 739–742 (2010)
87. Yan, J., Shen, S., Li, Y., Liu, Y.: An optimization based framework for human pose estimation. IEEE Signal Process. Lett. **17**(8), 766–769 (2010)
88. Yan, J., Zhu, M., Liu, H., Liu, Y.: Visual saliency detection via rank-sparsity decomposition. In: 2010 17th IEEE International Conference on Image Processing (ICIP), pp. 1089–1092 (2010)
89. Yan, J., Song, J., Wang, L., Liu, Y.: Model-based 3D human motion tracking and voxel reconstruction from sparse views. In: 2010 17th IEEE International Conference on Image Processing (ICIP), pp. 3265–3268 (2010)

90. Yan, J., Tong, M.: Weighted sparse coding residual minimization for visual tracking. In: 2011 IEEE Visual Communications and Image Processing (VCIP), pp. 1–4 (2011)
91. Yan, J., Tian, C., Huang, J., Albertao, F.: Incremental dictionary learning for fault detection with applications to oil pipeline leakage detection. Electron. Lett., IET **47**(21), 1198–1199 (2011)
92. Yan, J., Wang, Y., Zhou, K., Huang, J., Tian, C., Zha, H.: Towards effective prioritizing water pipe replacement and rehabilitation. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, pp. 2931–2937. AAAI Press (2013)
93. Yao, X., Liu, Y.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)
94. Yu, Y., Yu, X.J.: Cooperative coevolutionary genetic algorithm for digital IIR filter design. IEEE Trans. Ind. Electron **54**(3), 1811–1819 (2007)
95. Zhang, C., Ruan, X., Zhao, Y.M., Yang, M.H.: Contour detection via random forest. Proc. Int. Conf. Pattern Recogn. **2012**, 2772–2775 (2012)
96. Zhang, C., Li, X., Ruan, X., Zhao, Y.M., Yang, M.H.: Discriminative generative contour detection. In: Proceedings of the British Machine Vision Conference 2013, pp. 74.1–74.11 (2013)

Chapter 6
Applying Operations Research to Design for Test Insertion Problems

Yann Kieffer and Lilia Zaourar

Abstract Enhancing electronic circuits with ad hoc testing circuitry—so-called Design for Test (DFT)—is a technique that enables one to thoroughly test circuits after production. But this insertion of new elements itself may sometimes be a challenge, for bad choices could lead to unacceptable degradations of features of the circuit, while good choices may help reduce testing costs and circuit production costs. This chapter demonstrates how methods from Operations Research—a scientific discipline rooted in both mathematics and computer science, leaning strongly on the formal modeling of optimization issues—help us adress such challenges and build efficient solutions leading to real-world solutions that may be integrated into electronic design software tools.

6.1 Introduction

This chapter presents an overview of how optimization techniques from Operations Research can help improve the performance and reduce the cost of electronic chip testing. Operations Research allows one to design efficient optimization algorithms to solve optimization issues arising in the design of electronic design software to facilitate chip testing.

6.1.1 Electronic Chip Testing

The semiconductor industry owes its success to the ever-growing performance and integration of electronic chips. This is usually summarized under the term "Moore's

Y. Kieffer (✉)
LCIS, University Grenoble Alpes, 26900 Valence, France
e-mail: yann.kieffer@lcis.grenoble-inp.fr

L. Zaourar
CEA List, 91191 Gif-sur-Yvette, France
e-mail: lilia.zaourar@cea.fr

law", which is an empirical observation of the pace of improvement of electronic circuit density.

But this race to higher and higher density means on one side more and more complex design and fabrication methodologies; and on the other an increased difficulty for circuit testing.

Let us give just two figures. Some electronic chips production rate as low as 30 % yield, meaning that out of 100 chips produced, only 30 are functional and will be shipped. Trying to push the limits of the technology accounts for such a low figure.

Our second figure is the number of nonfunctional chips that may be shipped in a batch of chips to the customer. A typical figure would be, for example, 50 ppm (parts per million), meaning that no more than 50 nonfunctional chips can be shipped in a batch of a million chips.

To achieve such low rates, one needs to use specific testing techniques. This is the reason why Design For Test (DFT) emerged as a set of techniques. We will present DFT in Sect. 6.2.1.

6.1.2 Electronic Design Automation

With chips numbering well into the hundreds of millions of transistors, sometimes even billions of transistors, it is quite clear that computers and specific software are needed both to handle chip designs and also to carry out some operations for their implementation. Electronic Design Automation is the generic name for all the effort to provide software tools for electronic design, circuit implementation, and circuit fabrication preparation.

No design methodology can exist without support from EDA tools. In the case of DFT, some techniques are supported by specific tools tailored to that particular technique. Other techniques may be supported as libraries of ready-to-use electronic components.

6.1.3 Operations Research

The attempt to optimize some product or process can be conducted in two ways.

One of them is to improve one's knowledge of the application field, and bring in new ideas or new methods, leading to a more complex and more efficient system or service.

The other has a particular application context. It can be used only when choices have to be made depending on the input data for the particular process or design task, and those choices have an impact on the overall quality or cost.

The best example of such a case in electronic design is the so-called place&route step. It takes as input an abstract circuit presented as gates and their connections; it produces a geometrical placement in space of the gates and the metallic connections inside the chip.

Obviously, all the placement information of individual gates and wires are choices that need to be made during this step; and unwise choices can lead to a dramatic increase of the overall size of the resulting chip.

This kind of optimization problem calls for an optimization algorithm, meaning an algorithm taking input data, and computing from that data a solution or combination of good value, according to some numeric measure of the values of possible solutions.

The whole discipline dealing with optimization algorithms design, validation, and implementation is called Operations Research. Its emphasis is on optimization problem identification (also called the modeling step), and optimization problem resolution. It is a set of tools and techniques belonging both to mathematics and computer science, and helps one to design algorithms that compute good solutions within reasonable computation times.

Some electronic design automation (EDA) tools face optimization challenges for which Operations Research is the right answer. We exemplify this statement in this chapter in the case of Design For Test insertion.

6.1.4 Chapter Contents

This chapter has three main parts. The first part presents Design For Test, how it raises optimization challenges, and gives a general outline of how to tackle these challenges.

The second and third part are two case studies that illustrate the general statements from the first part. The first case study is about the insertion of scan chains at the Register-Transfer Level (RTL). The second case study handles the optimization of on-chip memory testing components.

6.2 Design for Test Optimization Challenges

Before describing the main optimization challenges related to Design For Test (DFT), we first present what is DFT and why it is important. Key techniques of Design For Test are then introduced, followed by key optimization criteria related to DFT insertion. Finally, we give hints at a general method for optimizing DFT insertion.

6.2.1 Design for Test: What Is It?

Once a circuit has been produced, its functioning is fixed, and cannot be changed. Whatever use can be made of the circuit depends only on the design decisions before production. This is a major factor influencing all aspects of the design of electronic circuits (so-called ASICs, application-specific integrated circuits).

Another important fact is that with advanced fabrication processes, not all circuits produced are functional. Actually, the fallout rate can be quite high. Since detecting faulty circuits after integration is too costly, circuit producers do their best to deliver only functional circuits. This means testing each circuit one by one before delivery to the customer.

But testing has to be thorough, unless the circuit's malfunction only shows at use time. The complexity of the internal part of the circuit, (up to hundreds of millions of transistors) as compared to the low number of access ports (usually less than one thousand) makes testing a very challenging step. Actually, testing time for one chip is a key parameter, since chips get produced by batches, but tested one by one.

Additionally, testing is done using an ad hoc device called a tester, which is a very high precision machine, costing millions of dollars. Series of chips get tested in parallel over several testers.

The key parameter for the test phase is its cost. This cost is directly proportional to the testing time of an individual chip. Hence minimizing this time is a key problem for integrated circuits fabrication.

It is a well-known fact that efficient testing of a complex circuit cannot be carried out quickly without enhancing the design of the circuit. Adding testing circuitry ("testing logic" is the jargon term) enables one to increase observability and controllability of the circuit. These terms refer respectively to the ability to set, and to read, at the time of testing, the state of any particular part of the circuit. Ideally, one would like to have 100 % observability and controllability of the chip, to actually be able to test thoroughly all parts of the chip.

This additional design phase is what is called Design For Test, DFT for short. It is the final design test: DFT has to be done before the transformation of the circuit description to layout, masks, and actual circuits. But it is usually done after the chip designer has delivered a functional design: it is seldom undertaken by the designer itself, and is usually left to specialized teams.

To sum up, testing is mandatory to sort out functioning chips from dysfunctional ones after production. But testing is very costly, and it is important to reduce its time. Testing time and testability are the two main criteria. Testability has to be near 100 %, and testing time should be minimized. The only known way to reach these objectives is to add testing logic to the design in a phase known as DFT, after the functional design is produced, but before the chip gets to the implementation and production phases.

6.2.2 DFT Techniques

DFT is a set of practices, technologies, and methodologies enabling the chip designer to enhance the testability of the circuit and enable a timewise efficient testing of the circuit.

DFT has core techniques for actually instrumenting the circuit part to be tested; and integrating techniques and technology to put together and orchestrate the functioning of the testing part of the circuit, and also offer an interface to the outside world through pins of the circuit. DFT techniques is a very comprehensive topic; the reader is referred to reference texts such as [10] for more details.

The present chapter focuses on the optimization challenges of the DFT phase of chip design. Optimization issues arise when implementing a test technique requires many choices, and these choices may vary greatly in terms of performance or cost.

The instrumentation of a part of a circuit in order to test it generates such optimization challenges. We are not aware of optimization problems arising from the integration of test systems, at least from the point of view of chip design. We will touch on a test integration challenge in the perspectives part at the end of this chapter.

We now present briefly the two main testing techniques: the so-called scan chain insertion, and the BIST methodology.

Scan chain insertion is a technique for testing irregular parts of the circuits. We say that a circuit or part of circuit is regular when it contains small structures that are repeated a big number of times. Two good examples of regular circuits are memories and CCDs. What one usually thinks of as a digital circuit—a piece of hardware that somehow computes something—is usually irregular.

The irregular case is the most difficult to handle. Scan chain insertion is a very costly technique in terms of chip area, but it is also very efficient. The difficulty in testing an irregular circuit is to access its internal state. Scan chain insertion removes this difficulty by actually transforming each and every storage element of a circuit into one that is freely readable and writable at testing time. Hence, the state can be completely known and set, and testing is reduced to testing the right function of the computational part of the circuit (combinatorial part in electronic parlance).

How this is achieved is simply stated as: all memory elements of the circuit are chained together into a shift register, together with the necessary logic to either enable this register (test mode) or leave it aside (functional mode). The technique will be presented in more detail in the first case study, which presents work on the optimization of scan chain insertion in a special case.

Before presenting the methodology of BIST, let us explain how any circuit is tested, be it instrumented through DFT or not.

The circuit to be tested (DUT, device under test) is placed on an industrial tester. A test program, which has been designed before the chip production, is run on the chip. Testing involves some analog part (so-called parametric testing) and some digital part (test vectors application). Test vectors are just arrays of 0's and 1's, or low and high voltage values, that are fed in sequence to the testing input pins of the circuit, while 0's and 1's get read from the output pins. Output values are tested against expected values. One bad value means that the chip is dysfunctional; dysfunctional chips are disposed off.

In the case of scan chain testing, the test vectors are actually states of the chip that get uploaded to the chip. Then the chip is activated for one step, and the state is again read out.

Regular circuits do not usually need such expansive means of testing. Memories, for example, can be tested directly on an industrial tester. Testing is done using regular functions of the memory, the main difference being that the testing patterns that are applied are regular, and are meant to thoroughly ascertain the function of the memory, while regular usage accesses are random and not tailored for testing limit cases.

The ease with which regular structures are tested justifies the ability to embed most of the test methodology on the chip, reducing the tester's work to that of an orchestra conductor: requiring test to begin, and getting back the test result. Such reduction of the job on the tester's part is meaningful if more than one part of the circuit get tested at the same time.

Embedding the testing logic in the circuit is the main idea behind the techniques collectively known as BIST, for Built-In-Self-Test. BIST has other benefits, like allowing the chip to test itself once it has come out of the industrial production environment.

BIST is mainly used in practice for regular structures, although it could be used also for irregular ones. Our second case study presents work on optimizing the testing architecture in the case where many small memories are embedded in a chip.

6.2.3 DFT Insertion Key Parameters

For most circuits, DFT is a necessity, for testability of the circuit would be much too low without it. Adopting DFT brings one new parameter into the picture—namely the fault coverage rate. Since DFT is meant to increase fault coverage, the fault coverage rate should be closely looked after.

Inserting DFT also brings a couple of challenges and constraints.

Since DFT is added circuitry, its addition will increase the area of the final chip. This increase is not always just a fixed cost in area, as will be clearly illustrated in the case of scan insertion (Sect. 6.3).

Another cost is the added complexity of the design flow when DFT is used. This has no incidence on the product itself, only on project management. We will thus disregard it in this work.

Testing a chip is an actual step after production of the chip. Chips are put on costly machines called testers. After the test, the machine emits a PASS or FAIL result. During the test, the chip is powered by the tester. But the power needed for the test may be unrelated to the power needed for normal use. It is common practice to design the power budget according to functional use of the chip, and to adapt the testing step so that it works within this power limit. Hence testing power should be watched, lest the chip be burnt and destroyed during the test.

Finally, as already mentioned in Sect. 6.2.1, the testing costs of a series of chips are proportional to the testing time of an individual chip of the series. Decreasing the testing time of one chip by 20 % means reducing the whole testing costs by 20 %.

To sum up, testing involves four key parameters: testability as measured by fault coverage, area increase, power use during test, and testing time.

6.2.4 DFT Optimization: Some Hints at a General Method

We will first set a general context for our discussion, and then give an outline of a method that encompasses the work of both the case studies presented below.

6.2.4.1 DFT Insertion Challenges

Our concern is DFT, meaning the design of additional circuitry that facilitates or even enables one to do chip testing after chip production. The objective while resorting to DFT is to enhance the testability of the circuit—this parameter should never be forgotten.

Another important part of the context is the setting for the optimization. Usually, our input will be a circuit that is to be enhanced with DFT. Sometimes, the input may be greatly simplified, as in case study B where only the memories of the circuit are considered.

As will be illustrated in both case studies, the insertion of this circuitry is based on two factors. First, a testing technology has to be chosen or designed. Second, the way this technology is added to the circuit should be considered wisely, for wrong choices could lead to unacceptable results.

The first part is the job of the DFT engineers, and has not much to do with mathematics or computer science. It is when the second factor exists that Operations Research may really help with DFT insertion.

Some additional input may be parameters that restrict the possible design choices for DFT elements; such will be found out by a close analysis of what DFT structures may be included in the circuit.

At least three factors will have to be considered when inserting DFT. The first factor is the increase of area of the circuit. Since no function can be added without an addition of circuitry, inserting DFT means increasing the area of the chip. Chip production costs are directly correlated to chip area. This area increase may be negligible, or not, depending on the DFT technology used.

The two other factors are related to the actual testing phase. The cost of chip testing is in direct relation to the time of testing. This is because testing uses costly resources (namely industrial testers). In fact, the cost of testing is becoming an increasing proportion of the total cost of chip production.

The time for testing all the chips in a series is roughly proportional to the time for testing one of the chips. For large series of chips, many testers are used in parallel to allow for not-too-long testing phases, meaning sometimes weeks. Time and money limits put real pressure on the testing phase. Hence, any technique that may help reduce testing time is appreciated.

One such general principle is that of testing several functions of the chip at the same time—it is some kind of parallel test at chip level. Testing functions 2 by 2 may reduce the testing time by a factor of 2, meaning financial gains of a factor of 2 for the testing part.

But this idea has a limit in that the testing phase consumes power, and the power source for testing is actually the same as the power source for the function of the chip. Testing several functions at once increases the power needed for testing, and this power has to be kept within certain bounds, lest the chip be burnt during testing.

The way chips are designed and produced today leads to the following context: chip power sources are calculated to accommodate the necessary power for normal function of the chip; and testing power has to stay within this power range. Hence from the point of view of a testing scheme, what is important is the testing peak power, meaning the instant power value at the time of testing when it is greatest.

As will be illustrated in our second case study, testing time and testing peak power are competing parameters. If one accepts to increase one of them, it does help to reduce the other. Higher peak power means more testing may happen in parallel, and this may enable testing time reduction.

Other parameters may show up as directly related to DFT insertion choices. One such example is the number of testing pins in case study A. One has to be careful though not to confuse actual optimization parameters, which are quantities that one would like to see minimized or maximized in the resulting circuit, with other quantities that one thinks would be helpful reducing or increasing in order to optimize those aforementioned quantities.

Finally, as in any optimization phase, one should keep in mind that the time needed for the optimization computations should be kept within reasonable bounds. Most DFT insertion tools need to give an answer within 15 min; overnight computations are usually not considered a viable option. Those time limits are consequences of the way electronic design is carried out: many tools have to be used in succession, and some step in the whole flow may fail, in which case one has to go back up to some previous step in the design flow. Finally, the time for traversing the whole flow is limited also, because of time-to-market considerations.

6.2.4.2 How to Approach DFT Insertion Optimization Challenges

At the onset, any DFT insertion requiring optimization would seem to optimize the following five criteria:

1. maximize testability
2. minimize area expansion
3. minimize individual testing time
4. minimize testing peak power
5. minimize optimization computation times

This is a reduced list: some parameters may be added after thorough analysis of the DFT technique used.

This would seem to place all DFT insertion issues into the field of multi-criteria optimization. Our experience show it is not always the case. This will be illustrated in case study A, where only one criterion is optimized. Our recommendation is to do the utmost to reduce the number of objectives before working on any optimization solution (method or algorithm).

Let us illustrate our argument with a real life example. Say you want to buy a new car. You like fast cars; but you would also like to spend as little as possible. The car dealer will make it clear to you that the fastest car is not the cheapest, and the cheapest car is not among the fastest ones.

In the case of a car purchase, you may be able to compromise between speed and price. This is difficult to automatize. In the case of DFT insertion, it may be that some optimization parameters may be left aside during the optimization phase.

The foremost reason to not care about an optimization parameter is that your optimization choices do not impact the value of the parameter. In both case studies below, testability was in no way impacted by DFT architecture considerations.

A weaker form of the same is when a parameter increase or decrease is negligible. For example, it may be that the area expansion in the worst case is less than 1/1000th area of the circuit. Then optimizing area increase may have no real industrial benefit.

Finally, some parameter may need to be kept within some limits, for example below some threshold, without any added value if it is further reduced. In that case, this apparent optimization criterion is better considered as an optimization constraint, therefore reducing the number of optimization parameters.

Minimizing optimization computation time is not to be considered a real optimization parameter; it is rather a modality of how optimization should be conducted. It is an ever-present concern, but is never explicitly mentioned in Operations Research studies.

If after all these simplifications, one there is still more than one parameter to optimize, then it is advisable to do the following analysis.

Let us first present two extreme cases:

1. Two parameters are clearly in competition, one with the other. In the case of DFT, it will usually be the case between testing peak power and testing time—if they are still to be considered at this step of the analysis.
2. All parameters improve when one particular parameter is improved. This parameter may or may not be one of those already considered.

All pairs of parameters have to be taken and assessed with regards to thier interaction to each other and try to estimate whether they are in competition, or may improve together, or are indifferent one to the other.

In case one parameter is indifferent to all others, optimization can be separated into several phases: one for this parameter, and one for the rest.

If two parameters are found to be in competition, then there must be a compromise to decide how to choose between competing parameters? Is it the responsibility of the chip designer, or may the DFT insertion tool provider already offer a satisfactory solution?

If it is preferable to leave the choice to the designer, then we are really facing a multi-objective optimization problem.

In the extreme case when all parameters increase with the increase of only one of them, we fall back to single criterion optimization, which is the most comfortable scenario to tackle optimization problems.

Intermediate cases may happen when one parameter is more important to optimize than another; in that case, optimization may again be decomposed into several phases, where each successive phase takes as input the result of the optimization of the previous phase.

Once this analysis of optimization criteria is done, it is a matter of going back to the usual method of work in Operations Research, which is:

1. Give a precise, mathematical formulation of the optimization problem, separating input data, output data, constraints that the output should satisfy, and a list of the optimization criteria together with optimization direction (maximize or minimize);
2. Simplify or reformulate problem into an equivalent problem, possibly simpler, or closer to classical optimization problems from the literature;
3. Adapt or design a solution method for the problem;
4. Validate on numerical data, and iterate the solution step if necessary.

6.3 Case Study A: RTL Scan Insertion

In this case study, we present work on the optimization of the process of inserting scan chains when it is done at the Register-Transfer Level (RTL). We first review the electronic design context of RTL scan insertion; we then apply our methodology to find out which key parameters should be optimized when inserting scan; and finally present our resolution method together with numerical results.

6.3.1 RTL Scan Insertion: Electronic Context

To review the electronic context of RTL scan insertion, we first give an overview of the scan chain technique. A presentation of the implementation flow for circuits follows. We end up with a discussion of the value of scan insertion at the RT level.

6.3.1.1 The Scan Chain Technique

Testing the so-called logical part of the design, that is the part that actually "computes", is quite a challenge if the design is not properly instrumented. The internal state of the design is one of 2^N if N is the number of memory elements of the design.

Putting the design into one of those states just through inputting determined values as input is not a realistic endeavor. Besides, a complete test would involve 2^{N+i} cases, where i is the number of inputs, since all input stimuli (2^i) should be tried against all internal states. Current designs, even small parts of designs, have N bigger than 1000, so this is definitely not a feasible approach.

The state of the art in testing is to model circuit faults on one hand, and diagnose them with efficient tools on the other. It is for this second part that scan chaining was introduced. With the help of a scan chain, the design can be put in any required state; and the state of the design can also be fully read. Between two such operations, running the design for the length of one clock cycle allows the detection of nearly all faults of the simplest types.

The scan chain technique aimed at enriching the functional design with another mode called test mode. In order to provide it, the memory elements of the circuits are chained into a unique long shift register, with the help of multiplexer in front of the inputs of the memory elements. Outputs are sent to the next element of the chain, and multiplexers indicate whether the input is to be taken from the previous memory element (test mode) or from the circuit (functional mode).

This technique is usually implemented on Netlists, which are abstract representations of circuits by way of gates, nets ("wires") and their connections. But the same can also be carried out directly in the RTL code, producing an enriched RTL of the design, providing a test mode alongside the functional mode of the design. We will explain the reason for doing such a thing after a short presentation of the implementation flow.

6.3.1.2 The Implementation Flow

As can be seen in both parts of Fig. 6.1, the implementation flow is the part of the design process that starts from a design description in RTL (handed over by a design engineer), and ends with the layout description of the circuit. The layout, as its name suggests, describes how the circuit is to be laid out as parts of a semiconductor substrate, metal layer, and insulating layers: it is the spatial representation of what the circuit will look like once produced.

The two main steps of the implementation are the synthesis step, which produces a Netlist from an RTL design; and the place&route step which produces the layout from a Netlist. This is a gross simplification of the whole process, but it is quite enough for our discussion about the place of scan in the implementation flow.

The usual way to insert scan is to do it on a Netlist, as shown on the left part of Fig. 6.1. The context of this work is the flow shown on the right, where scan is inserted already in the RTL design. We do not have to argue about this methodology here since it is part of the context of the study. But we do summarise the benefits of such a method for the reader who is not familiar with it.

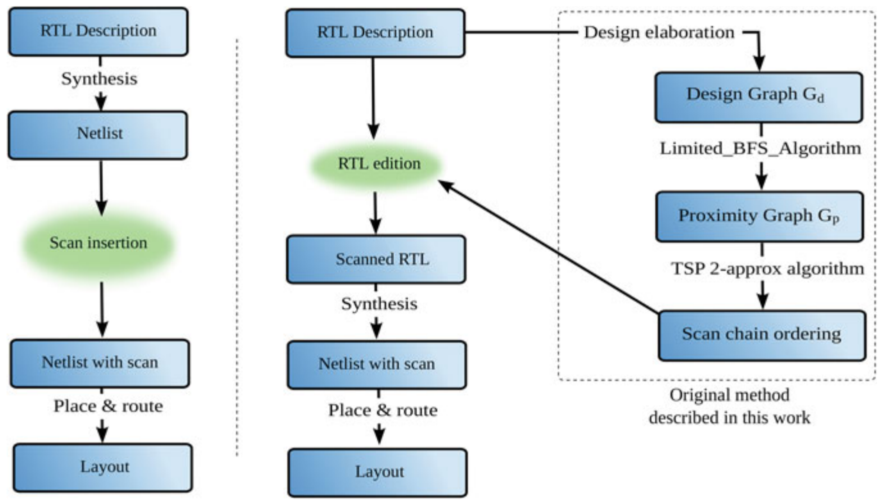


Fig. 6.1 Classical scan insertion versus our method

6.3.1.3 RTL Scan Chain Insertion

We now discuss when in the design flow of a circuit the scan chains insertions step should take place.

The common practice among designers, and indeed the only one broadly supported in VLSI CAD tools, is to insert scan chains between the synthesis and place-and-route steps. Synthesis is the transformation from design "lines of code" (so-called RT-level descriptions) to circuit elements (gates, ports and nets for connecting them); while place-and-route is the geometric step of choosing where electronic elements should be placed, which means passive and active gates, and also connecting wires. Placement is a 2D process, and routing is happening in three dimensions: The circuit is laid out on a 2D floorplan, with several layers of metal above. Gates are assigned to the lowest level, and connections can be realized either in this level or in the higher metallic levels.

Placement and routing are the most important and most visible discrete optimization problems in the whole VLSI design flow. This chapter is an illustration of other issues in VLSI design for which the techniques of Operations Research can be applied.

Research has been carried out to investigate the possibility to take DFT steps, especially scan chain insertion, before synthesis is started [1, 2, 4, 6, 9]. The two possibilities are presented in Fig. 6.1. Part of the rationale for this is that DFT is actually a design step, and thus should not be mixed with implementation steps such as synthesis of place-and-route. Another strong point is the fact that synthesis design tools try hard to optimize the design transformation to actual gates and wires; while scan insertion can be a costly step in terms of additional logic. Hence it is better to have it done before synthesis, so that synthesis can optimize away unneeded circuitry.

Such a move is usually referred as “RT-level DFT” of “RT-level scan chain insertion”. The main challenge with RT-level scan chain insertion is that the scan chain technique is initially a gates-and-wires technique, in that it is meant to test actual (combinatorial) gates, by using the state gates also known as flip-flops (FFs). It has been observed that the FFs generated by the synthesis tool can be computed beforehand; and that the scan chain function itself can be described as an additional piece of RTL code [1, 9].

But inserting scan chains at RTL induces another challenge: deciding what is a good order for chaining the memory elements together. This is also an issue in traditional scan chain insertion, but it is less of a problem in the traditional way of doing it since more information is available regarding the actual cost of chaining.

We will present in this study our work on how using Operations Research on the RT-level scan insertion problem led to an elegant and efficient solution, showing that this apparent hurdle of choosing the chaining order can be overcome.

6.3.2 RTL Scan Optimization: Delineating the Optimization Problem

Following our methodology as presented in Sect. 6.2.4, we set out to identify the optimization problem inputs and output, and the quality measure of possible solutions.

The input in this case is the actual design of the electronic block that is to be instrumented by scan. In the case of RTL scan, this block is presented in RTL code—either Verilog or VHDL.

The output of the optimization problem in itself is the way in which memory elements in the design will be chained. A scan chain is a succession of memory elements. The only rule is that each memory element should appear exactly once in one of the scan chains of the design. There is no restriction on the admissible order of the memory elements inside a chain, although it is pretty clear that the order will have an impact on the quality of measurement of the set of chains.

The electronic parameters related to the chaining, and that are candidates for evaluating the quality of a set of chains are many. Let us list them.

1. Number of scan chains: having multiple scan chains enables to load and unload chains in parallel, thus saving on testing time.
2. Longest scan chain: testing time is proportional to the longest scan chain, since most of the time during scan testing is the time needed to load and unload chains.
3. Testability.
4. Power during test.
5. Expansion of area of chip due to scan insertion.

As outlined in Sect. 6.2.4, it is always best to try to optimize only one criterion, for two criteria might already be in competition one with the other. We have listed five parameters; we now clarify which parameters really change with the choices of chaining, and then try to reduce the set of parameters to the minimum possible—ideally only one.

The number of scan chains is actually irrelevant as an optimization parameter, since it is only the length of the longest scan chain that dictates the testing time. But the number of chains is also the number of testing ports of the chip, and this may very well be upper bounded by design constraints. Hence the number of scan chains is to be considered as limited by a fixed constant C known at design time.

The longest scan chain is to be minimized. Since all memory elements need to be chained, minimizing the longest scan chain means having all scan chains of about the same length, which is the total number of memory elements divided by the number of chains, and an extra chain to accommodate for rounding up numbers. This is then fixed at design time, and does not need to be optimized.

Increasing testability is the reason why we insert scan chains in the first place. But testability is not affected by the chaining scheme, and thus is not to be considered for optimization.

Power during test has two components: power dissipated by the activity of switching gates; and power dissipated in nets. The switching power is not affected by the chaining scheme. The power dissipated in nets increases with the length of the wires. Hence power during test can be approximated by the total added wirelength due to scan chaining. Minimizing power during test can be achieved by minimizing total added wirelength.

Finally, scan insertion may lead to a dramatic increase of the chip size at the place&route step, and this is certainly to be considered as the main parameter for optimization. Unfortunatly, it is difficult to estimate precisely this quantity starting just from a scan chain scheme and the design. Analyzing the cause for area increase leads to two sources of increase. Chaining replaces gates by others, or adds gates in certain cases, but this impact on area cannot be too extreme. Chaining also add wires, actually one connection per memory element of the design. This is a much more forceful operation, and can dramatically alter the routability of the block. It is this component that should be kept in check, or optimized if at all possible.

Again, routability is difficult to estimate just from a design and a scheme of chaining. Placement software gives a congestion estimate, but it is the result of a complex algorithm, and cannot be easily incorporated in a mathematical model.

We somehow need to measure something that relates to the increase of the risk of congestion, but may be more easily evaluated. We found out that the added wirelength due to chaining is such a parameter. Indeed, the longer the wires that are added, the more stress is applied to the routing step. So minimizing the wirelength should help minimize congestion during place&route.

Keeping the added wirelength as the only optimization parameter is both satisfactory and unsatisfactory. The upside is that it represents the two remaining optimization criteria after analysis, namely power during test and area increase.

The downside is that it cannot be directly estimated based on the input and output of the optimization step, namely the RTL design and the chaining scheme. We make for this shortcoming as part of our resolution methodology for this problem.

6.3.3 RTL Scan Optimization Problem Resolution

In order to present our solution, we first give problem simplifications that allow it to fit better known optimization problems; then we solve the issue of not being able to evaluate solutions by replacing the added wirelength metric with a more suitable one; we then present the algorithmic solution to the resulting problem, together with numerical results.

6.3.3.1 Problem Simplifications

We recap from the previous section the optimization problem that we have to solve:

Optimization problem 1 Given an RTL design with N memory elements, and a maximum number of chains C, define C scan chains (i.e., sequencing of memory elements of the circuit) each of length at most $\frac{N}{C} + 1$ such that the added wirelength after place&route is minimized.

Now this problem has as a subproblem the case when $C = 1$, i.e., when there is only one scan chain:

Optimization problem 2 Given an RTL design, define a total ordering of the memory elements for stitching a scan chain such that the added wirelength after place&route is minimized.

Experimentation shows that optimization problem 1 can be reduced to optimization problem 2 simply by cutting the chain returned by a solution to optimization problem 2 into segments of the right size: the global added wirelength is not much different in one case from the other.

Hence it is optimization problem 2 that we will tackle in the following.

6.3.3.2 Another Measure for Solutions Quality

Our aim is to find a reflection in the world of the RTL design and its possible scan chains of the quality measure we have chosen for our problem—namely added wirelength due to chaining. However, between the final version of the design given by the designer, which is the input to the scan stitching tool, and the point in the flow where the wirelength can be measured—which is after place & route, there are two major transformations occurring, namely the synthesis step and the place & route step.

Synthesis replaces high-level descriptions with lower level descriptions, and this is not a big issue for our present concern.

Place & Route are the tools that actually decide where each element will be in the design, together with the routing of the connections. We need to gather some insight about how the placing and routing tools work to find out a correct reflection of our objective in the design world.

We focused on this observation: the geometrical distance defined by the placement tool is likely to reflect the logical distance in the design. By logical distance or logical proximity, we mean the number of components to go across to actually go from one memory element to the other in the design. This makes perfect sense in a Netlist, since a Netlist is a kind of labeled graph where some nodes represent gates and some nodes represent nets, with adjacencies between gates and nets that are actually connected in the circuit. The Netlist is the output of the synthesis tool and the input of the place&route tools.

According to [1], it is also possible to infer memory elements from an RTL description of the design; and with the help of a light synthesis step, which is to be included into the scan stitching tool, a generic kind of Netlist can be generated, which allows one to do the same work as described in the previous paragraph.

More precisely, given a design presented as a Netlist, this can be modelled into a so-called Design Graph (D-graph in the following) with gates and nodes linked as described above.

From a D-graph, we compute the so-called Proximity Graph (P-graph in the following) in the following way. P-graph vertices are the memory elements of the design. P-graph edges represent paths in the D-graph that do not go through another memory element. The edges are valued with the number of elements that are gone through by the path.

The whole process is illustrated in Fig. 6.2, which shows (a) an example design, (b) the associated D-graph, (c) the associated P-graph.

Computation of the P-graph, although easy to describe, can be a computation bottleneck. Our implementation uses breadth-first search on the P-graph, together with a threshold value t above which no edge is inserted in the P-graph when paths are longer than t.

6.3.3.3 Optimization Algorithm for Scan Chain Stitching

Now the problem to be solved is:

Optimization problem 3 Given an undirected graph G with values on the edges, find a total ordering on the set of vertices that minimizes the sum of the values along the sequence.

This is actually not a correct formulation, for what is to be counted as the cost of going from vertex u to vertex v when there is no edge uv in the graph?

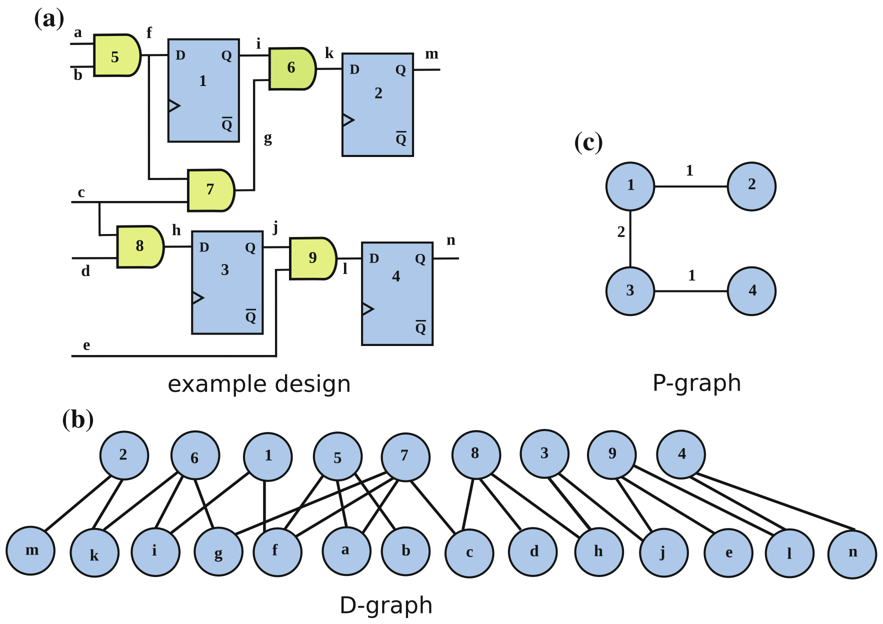


Fig. 6.2 Example design and associated graphs G_P and G_D. **a** Example design, **b** D-graph, **c** P-graph

A natural answer is to assign to such a pair the length of the shortest path between u and v in this graph. Since values on edges represent proximity, this proximity function can naturally be extended by computing the shortest paths for pairs of unconnected vertices.

After this operation, we are left with this problem:

Optimization problem 4 Given a complete graph G with values on the edges, find a minimum cost path traversing each vertex exactly once.

This problem is a classical problem: it is the hamiltonian path version of the traveling salesman problem (TSP). In TSP, one looks for a cycle traversing each vertex exactly once; in our problem, we look for a chain, meaning we do not have to come back to our starting vertex. One problem can be reduced to the other by addition of a vertex joined to all other vertices, with equal weights on all the new edges.

Now we have a good range of well-known algorithms to solve this problem. Since TSP is known to be a difficult problem (NP-complete is the technical term), there is little hope to find an efficient algorithm that finds the best solution every time. Actually, a good solution is enough, since our model is an approximation anyway, so investing a lot of time to find the absolute best solution makes little sense.

We have to keep in mind that all these problem transformations translate to actual computations to prepare the input data for the optimization algorithm that will be used. Since one wants to handle big enough data sets, some of these transformations may be too costly in practice.

For example, computing all the shortest paths for all pairs of vertices can become prohibitive if the input graph is too big. This is the reason why we solve problem 4 by working on the data of problem 3. One algorithm in particular allows this change of input easily. It is a nameless algorithm, which is classic in the literature on TSP. Let us call it the 2-approximation for the TSP. A 2-approximation optimization algorithm means that each solution given by the algorithm is at most twice the value of the optimal solution.

This algorithm starts by computing a minimum cost spanning tree on its input edge-valuated graph G. It so happens that the actual cost of a solution of this algorithm when run on a graph that is not complete is the same as that obtained by running the algorithm on the graph completed as in problem 4.

Hence the solution given by this algorithm is actually a solution for the TSP problem on a complete graph that we never need to compute.

Extensive testing of this method has been carried out on both academic and industrial designs. An industrial synthesis, place-and-route flow has been used for the tests. Our method has been tested against the integrated scan insertion tool from the same design flow.

The results are very good, with an overall reduction of wirelength for the whole design (not just added wirelength due to scan insertion) between -5 and 20 % (-5 % being an actual increase in wirelength).

For more details on the method used to solve this particular problem, together with numerical results, the reader is referred to [12, 13].

6.3.4 Conclusion

Trying to insert wisely scan chains at the RT level seems at first to be a daunting problem, for minimizing the area expansion is difficult to model. Furthermore, each DFT insertion problem appears to be multi-objective. We have shown how in the case of a scan, the optimization problem can be reduced in several steps to a mono-objective optimization problem, and then solved with well-known algorithms.

6.4 Case Study B: Integrated Memory Testing in SOCs

We now illustrate our methodology on another testing-related optimization problem, namely the optimization of the architecture for testing memories with a particular shared testing technology.

6.4.1 General Context and Objectives

After presenting the context and stakes of memory testing for SOCs, we describe the testing technology which our optimization problem arises from. We identify the general nature of the optimization challenge, and give a first high-level description of the optimization problem we are facing.

6.4.1.1 Introducing the Memory Testing Challenge for SOCs

The previous case study addressed logic testing, with the scan chain technique.

This case study focuses on memory testing [7]. Memory testing, from the point of view of DFT, is becoming an increasingly difficult problem. The main point is that Systems On Chips (SOCs) have more and more embedded memories; and where yesterday's chips had just a couple of memories, today's chips may have thousands of them. Where as manual techniques were operative just a few years ago, automatizing is now mandatory.

We recall the main fact concerning testing: although chips get efficiently produced in series, testing always has to be carried out chip by chip, be it on wafer or in package. The mass production scheme is what allows costs per unit to decrease, but it entails an increase of complexity for the testing phase.

Memory testing went from tester-based, to autonomous testing, called BIST for Built-In Self-Test. In BIST testing, the tester only sends the signal that the memory should test itself, and embedded logic drives the whole testing procedure, delivering a result of PASS or FAIL back to the tester. Since memories are regular structures, testing them with the BIST methodology is both feasible and desirable, where it is seldom used for logic testing.

Implementing BIST with several memories calls for a BIST design to architecture the test system. Several such architectures have been proposed in the literature [3, 16].

The adoption of BIST allows parallel testing, where different blocks are tested at the same time. This brings a new challenge, which is to ensure that the power budget for the testing operations does not exceed the initial functional power budget for the SOC.

In this case study, we address the optimization problem that emerges with the choice of a particular sharing architecture for the memory testing components. This architecture is broadly presented in the next section.

The optimization objective is the joint design of a shared testing architecture for embedded memories, and a scheduling for the actual testing of those memories. Schedule and architecture are linked, since the components in the architecture define the temporal behavior of the test—typically, sequential testing and parallel testing for subparts of the block define when each memory test actually begins.

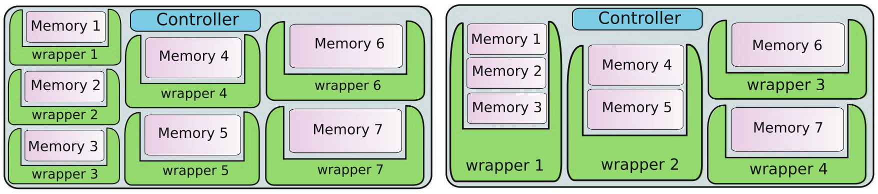


Fig. 6.3 Full-dedicated testing architecture, versus shared architecture

Although this study illustrates the architectural features of a shared BIST memory testing solution, the focus is mainly on the optimization challenge associated to its use for block testing.

6.4.1.2 Testing Architecture

After production of the chip, each memory in a SOC has to be tested. In the case of BIST, most of the work for the test of each memory is actually carried out by a dedicated component, called a *t*esting wrapper (or simple "wrapper" in the following).

This component has as its only function the test of memories. But under certain conditions, one such component can orchestrate the test of several memories; such wrappers are called *s*hared wrappers. In the case when a wrapper tests only one memory, we call it a *d*edicated wrapper.

There are two kinds of shared wrappers. *P*arallel wrappers test all their memories simultaneously. *S*equential wrappers test their memories in sequence. Shared wrappers may have cardinality constraints—meaning that there may be an limit to the number of memories that can be attached to a single wrapper. In this work, both sequential and parallel wrappers could be connected to at most four memories.

The possibility of sharing wrappers is restricted according to the kinds of memories. Some pairs of memories can be shared in sequence, others cannot. The same goes for parallel sharing.

Each memory must be attached to exactly one wrapper. When a wrapper is attached to a single memory, it is then a dedicated wrapper, which is areawise the smallest kind of wrapper. The bigger area of shared wrappers (due to its more complex function) is compounded by the fact that it is shared: thus the amortized area is smaller than that of a dedicated wrapper, and thus sharing helps decrease the additional area dedicated to memory testing.

From a technological point of view a central source, called the controller supplies stimulious signal to all wrappers. The rule in this architecture is that the work done by each wrapper is started at t_0, the moment the whole block is supposed to start its test. The time when a memory gets tested depends on the kind of wrapper it is attached to. For dedicated wrappers and parallel wrappers, all memories attached to them get started at t_0. For sequential wrappers, the first memory in the sequence is tested at t_0, the second one after the first is finished, and so on.

Figure 6.3 shows two memory testing architectures. The one presented on the left uses one wrapper for each memory. The one on the right gives an example of a sharing architecture. The number of wrappers is less; and the memory test might also be taking less time, depending on the architecture choices that were made.

6.4.1.3 Optimization Challenge

It is assumed that the designer has designed his block, and chosen the exact types of embedded memories. Some of these can share testing wrappers. Sharing a parallel wrapper allows for quicker testing, and decrease area. Sharing a sequential wrapper allows for less power-hungry testing, and decrease area also. Some memories need to have their own wrapper and cannot be shared. Some memories may also have dedicated wrappers, in case the optimization package cannot find a suitable sharing for this particular memory. Dedicated wrappers use less area than sharing wrappers; but since sharing wrappers are shared, the total area for wrappers is less when sharing than when using dedicated wrappers.

So it appears that a shared memory testing methodology brings the ability of reducing both area and testing time, as compared to testing all components one by one in sequence—which was the usual memory testing practice for less complex blocks. But decreasing testing time means increasing power consumption, and power consumption usually cannot be increased indefinitely, unless the chip is burnt during testing.

The optimization objectives for this problem are thus threefold: chip area, chip testing time, and chip testing power.

One could hope that chip area and chip testing power could be determined before chip design, leaving only chip testing time as the key parameter. Indeed, the cost of testing is directly proportional to testing time, thus each decrease of testing time is valuable. Unfortunalty, the practice of design today is such that area and power budget cannot be predicted at block-level, and are known only at the level of the whole SOC. Unrolling the whole design flow and tools is necessary to assess precisely the block area and the block functional power budget.

So the three objectives must be kept and optimized together. It is up to the designer using this optimization software package to choose the best compromise solution between these three factors.

Finally, not all pairs of memories can share wrappers, since sharing is restricted based on the actual characteristics of the different memories. This sharing ability is also distinct for sequential and parallel sharing.

So the optimization problem to tackle is the following:

Optimization problem 5 Given a set of memories, and compatibility groups for parallel and for sequential sharing, devise a sharing architecture optimizing the three parameters: chip area, testing time, and testing peak power.

6.4.2 Optimizing the Memory Testing Architecture

6.4.2.1 Detailed Methodology

We first describe the input data of the problem and the expected output. Then we present the formulas allowing us to assess the three performances criteria that are: area overhead due to wrapper insertion in the chip, consumption during the test phase and test time.
Input:

- A list of memories $\mathscr{M}$, with for each memory m four parameters:
  - Test time T_m, power consumption P_m, memory wrapper area A_m, and number of bits B_m;
- Compatibility groups for sequential sharing;
- Compatibility groups for parallel sharing;
- A maximal number of memories $N^p_{\max}$ (resp. $N^s_{\max}$) that a parallel (resp. sequential wrapper) can contain;
- A maximal number of bits (bitwidths) $B^p_{\max}$ (resp. $B^s_{\max}$) that a parallel (resp. sequential wrapper) can contain;
- A reduction factor $\alpha_p(k)$ (resp. $\alpha_s(k)$) on each memory wrapper area if it is sharing a parallel (resp. sequential) wrapper composed with k memories.

Output: We need to partition $\mathscr{M}$ in three sets of wrappers ($\mathscr{D}$, $\mathscr{S}$, $\mathscr{P}$) such that:

- For all sequential wrappers $\mathscr{W} \in \mathscr{S}$, all memories $m \in \mathscr{W}$ are compatible with each others in sequential, $\sum_{m \in \mathscr{C}} B_m \leq B^S_{\max}$ and $|\mathscr{W}| \leq N^S_{\max}$;
- For all parallel wrappers $\mathscr{W} \in \mathscr{S}$, all memories $m \in \mathscr{W}$ are compatible with each others in parallel, $\sum_{m \in \mathscr{W}} B_m \leq B^P_{\max}$ and $|\mathscr{W}| \leq N^P_{\max}$.

Evaluation criteria:

- **Test time**: All wrappers are launched simultaneously during test. Hence the test time T for the whole BIST architecture B is the maximum among the test times of the BIST wrappers in the design:

$$T = \max_{w \in B} T_w \tag{6.1}$$

  With T_w the test time needed by wrappers, w: $T_w = T_m$ for dedicated wrappers, $T_w = \sum_{m \in w} T_m$ for sequential wrappers, and $T_w = \max_{m \in w} T_m$ for parallel wrappers.
- **Peak power**: All wrappers are launched simultaneously during test. Hence the peak power consumption P of the whole BIST architecture B is given by the sum of all individual wrappers contributions:

$$P = \sum_{w \in B} P_w \tag{6.2}$$

With P_w the power consumption needed by wrapper, w: $P_w = P_m$ for dedicated wrappers, $P_w = \max_{m \in w} P_m$ for sequential wrappers, and $P_w = \sum_{m \in w} P_m$ for parallel wrappers.

- **Area overhead**: The total area A of the BIST architecture B is the sum of the area of different wrappers.

$$A = \sum_{w \in B} A_w \tag{6.3}$$

With A_w the area needed by wrapper w: $A_w = A_m$ for dedicated wrappers, $A_w = \sum_{m \in w} A_m \times \alpha_s(|w|)$ for sequential wrappers, and $A_w = \sum_{m \in w} A_m \times \alpha_p(|w|)$ for parallel wrappers.

Dealing with a multi-objective optimization, with contradictory objectives such as peak power and test time, there is no unique optimum. Therefore, we want to give to the user a set of good solutions and let him choose the best compromise (Fig. 6.4).

6.4.2.2 Problem Resolution : Designing an Adhoc Genetic Algorithm

Due to the complexity of our problem, we chose evolutionary multi-objective techniques to solve our problem. In fact, Evolutionary Algorithms (EAs) are particularly suited for multi-objective optimization problems since they work with sets of feasible solutions, and our expected outcome is a set of good compromise solutions [5]. We developed an optimization algorithm inspired by multi-objective genetic algorithms that we call Simple Multi-objective Genetic Algorithm For Memory Test Optimization (SMGA-MTO) as represented in Fig. 6.5. Our SMGA-MTO is a genetic algorithm that has been designed to produce sets of nondominated solutions.

6.4.2.3 The Chromosome

Genetic algorithms work on sets of feasible solutions to the optimization problem. To each solution is associated a representation known as the chromosome. In our case, each gene of the chromosome is associated to a particular memory of the design, and contains two pieces of information: the wrapper to which it is attached, and the way this wrapper is shared.

An example is depicted in Fig. 6.6. The chromosome length denoted by m corresponds to the number of memories present in the design: in the example, $m = 6$.

6.4.2.4 Initial Population

The initial population is created randomly. For this, we pick randomly a wrapper for each memory of the design with one of the three possible configurations (dedicated, parallel or sequential) with respect to sharing constraints. The initial population

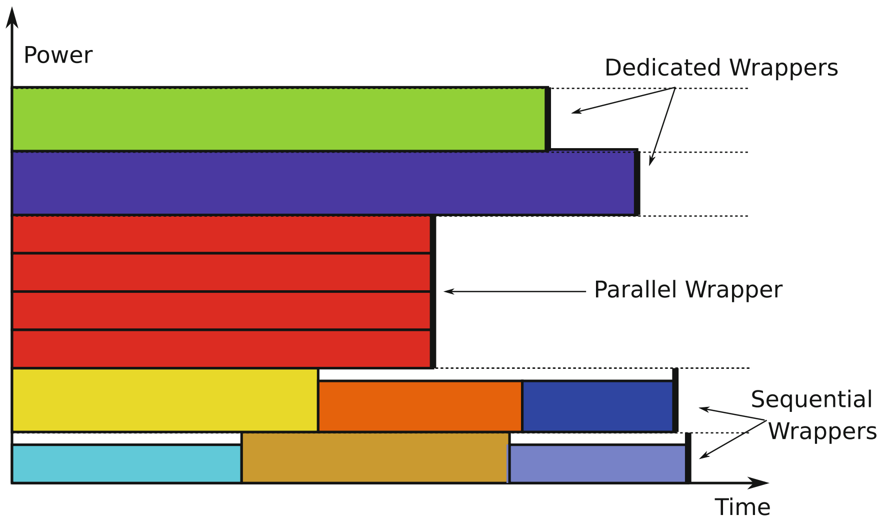


Fig. 6.4 Full-dedicated testing architecture versus shared architecture

size is a design parameter denoted by N, and depends on the requirements of the application. In our case, we fixed it through experimentation.

A random chromosome may not represent a feasible solution. Wrapper sharing types may be incompatible among different genes, and some sharing might not respect the sharing constraints. We apply a repairing routine that solves these issues while trying to retain the random character of the population.

6.4.2.5 Selection Operator

The operator used in this is the tournament selection. The three criteria, area, test power, and test time, are used for the evaluation. The tournament selection is applied as follows: individuals are picked at random in pairs; only the best survives. To decide which one is the best, Pareto dominance is used. In this context, a solution is Pareto-dominated if it is worse than another solution on each objective function: area, power consumption, and test time. The formulas for evaluating each solution for the three criteria are described in Sect. 6.4.2.1.

6.4.2.6 Genetic Operators

The traditional genetic operators are crossover and mutation. They help increase the diversity of traits among the solutions population.

Several crossover operators have been proposed in the literature of genetic algorithms. A single crossover operator is applied followed by a single mutation in our

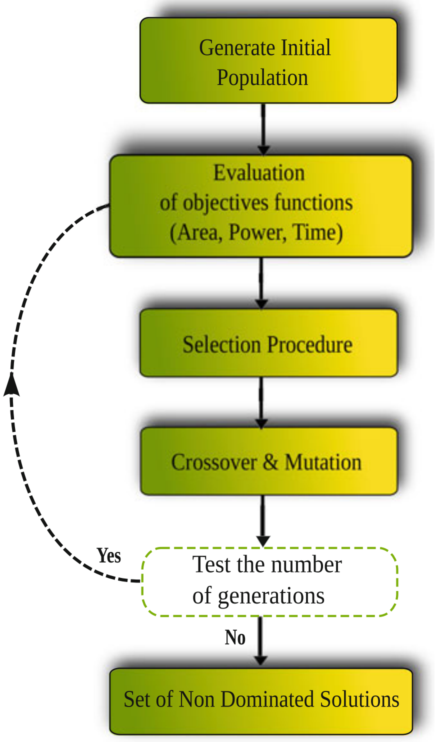

Generate Initial Population
Evaluation of objectives functions (Area, Power, Time)
Selection Procedure
Crossover & Mutation
Test the number of generations
Yes
No
Set of Non Dominated Solutions

Fig. 6.5 Genetic algorithm scheme

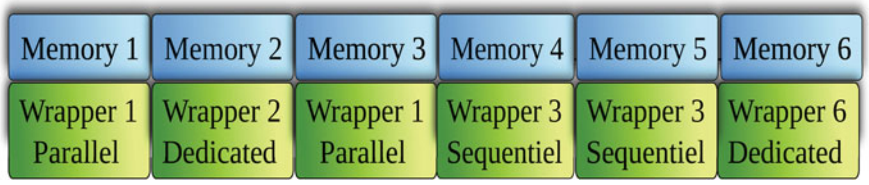


Fig. 6.6 Chromosome representation of a chip with 6 memories

approach. Objective values and dominance are not considered at this stage. They are applied later, at the replacement stage. The details about these two operators used in this study are described here.

Crossover

As reported previously, the single point crossover operator is used. This operator starts by choosing randomly a pair of survival solutions, with a probability P_c, and selects randomly a crossover point c, $0 < c < m - 1$. At this crossover position, the genetic behaviors of the parents are exchanged to build new children solutions. For this, we copy the first c memories of the parent's solution $p1$ and $p2$ to their respective children $s1$ and $s2$. Then copy the last $(m - 1 - c)$ memories of parent's $p1$ and $p2$ to the children $s2$ and $s1$, respectively. After this step, $s1$ and $s2$ become the new individuals.

Then the repair operation mentioned earlier is applied, to ensure that the resulting solution is feasible.

Mutation

The mutation operator consists here on switching randomly a wrapper configuration assigned to a memory, with a probability P_m. The mutation operator is applied with probability generally low compared to the crossover. The choice of a wrapper is done by generating a random number, if this number is less than the probability of mutation P_m then mutation operator is applied to this wrapper as follow: if its configuration is initially dedicated then a configuration is chosen parallel or sequential randomly and if the corresponding memory is compatible in parallel or sequentially.

The new individual is checked for feasibility. If it is feasible, it replaces its parent. If not, it is discarded.

6.4.2.7 Parameters Determination

These are mainly determined through experiments. One iteration consists of the selection process, crossover, and mutation. The new population obtained at the end of iteration replaces the previous generation. We stop after a certain number of generations denoted by K. This number K is fixed experimentally based on the required quality of the solutions and computation time required. We conducted extensive experimental tests over large chips. Our findings are reported in the next section.

6.4.2.8 Numerical Results and Discussion

In order to estimate the practical relevance of our approach we conducted several numerical evaluations using real-world designs provided by an industrial partner.

For all the cases, the genetic algorithm was used with an initial population of 200 individuals, evolving for 5 generations. The crossover probability was set to 0.995 and the mutation probability to 0.05.

For all designs, the optimization time is less than 10 s.

Table 6.1 Numerical results for the multi-objective genetic algorithms method ([11])

		Dedicated solution D	Sampled genetic algorithm solutions		
			S_0	S_1	S_2
Design 1	Time	38	40	38	38
	Area	2173	1538	1692	1625
	Power	973	774	734	786
Design 2	Time	122	125	122	130
	Area	2658	2133	2120	2132
	Power	1192	975	1052	972
Design 3	Time	40	40	50	43
	Area	3538	2895	2862	2890
	Power	1543	1317	1219	1252

Table 6.1 shows for each design the values of area (A), test power (P), and test time (T) for the dedicated solution (D), and for the three solutions provided by the algorithm that can best match this solution (S_0, S_1, S_2). The real values are hidden by the application of a multiplication factor. Hence, no units are given. This does not alter the value of the following analysis and discussion.

From results in Table 6.1 the following observations can be made. For Design 1, the user can choose a solution matching the test time of a dedicated solution, having a 22 % reduction in area and a 24 % reduction in power S_1, or a 25 % reduction in area and a 19 % reduction in power S_2. Or the user can go for area reduction with S_0 (29 %), with a test time only 6 % longer.

In Design 2, all three solutions reduce the area by 20 %; S_1, with a test time matching that of the dedicated solution, reduces power by only 12 %; S_0, taking 2 % longer to test, achieves 18 % power reduction.

In Design 3, solution S_0 matches the minimum test time, while allowing for 18 % less area and 15 % less power than using dedicated wrappers only. For a 9 % increase in test time, S_2 has slightly less area, and achieves a 19 % power reduction. If the user allows for more time, S_1 goes to 21 % power reduction and 19 % area reduction, costing 25 % more test time.

Other tests have shown that the running times stay within 20 s even for designs containing as much as 500 memories.

The strengths of this approach are its short running time, the fact that it takes all three optimization criteria into account, and its actual capability to release the circuit designer from any optimization work, enabling him to stay focussed on his task, which is electronic design, with its many constraints and challenges.

The only alternative work on the same topic known to the authors is that of [8]. The method presented in this article works by finding cliques in graphs, accounting for a fast-growing complexity. It is benchmarked on designs with up to 50 memories. It is not clear whether this method can be adapted for use on designs with more than 100 memories.

6.4.2.9 Memory BIST Architecture Optimization Summary

Using Operations Research on the memory testing architecture problem, we were able to design a multi-objective optimization algorithm for a problem which, before that, was left in the hands of test engineers for manual solution, despite them not being specialists either in multi-objective optimization, nor in BIST architecture design.

More details on the context of the problem, and the method used to solve the problem, can be found in [14, 15].

6.5 Conclusion and Perspectives

We conclude our journey in the land of DFT and its optimization challenges. The technologies of DFT are quite complex, and approaching their optimization challenges may be daunting at first. Operations Research invites one to neatly separate what is related to the optimization issue from what is not. Once the real stakes for the optimization are identified, and the optimization problem is formalized, the classical techniques and tools from Operations Research apply to actually solve the problem.

This is actually the main stanza of Operations Research: defining the optimization problem before trying to solve it! The focus on the notion of Optimization Problem is a main ingredient of Operations Research, allowing it to classify optimization problems, before classifying solution methods for those problems.

Both case studies presented make a point for the application of this discipline in the case of DFT insertion. Actually, it may be surmised that the same attitude would help in other contexts of electronic design where optimization issues arise.

A perspective of this work would be to explore more fully the possibilities of parallelizing the tests that are applied to the chips. Tests are thought of as tasks that are to be effected in sequence, and not much has been proposed for testing several different parts of a chip at the same time. The interest of such an undertaking lies once more in the reduction of the testing time of an individual chip, and thus of a whole series of chips.

The hurdles to overcome are again of two orders. Technology has to be devised to enable this possibility. But once the technology has appeared, planning software should help the designer in deciding the right architecture and right planning to actually reap some gain from this new possibility. Since one is not usable without the other, some kind of codesign is needed to actually make those two kinds of devices appear in usable products.

Finally, those two tools have to be integrated into the design and production flows in order to be of any use to the semiconductor company; this is another major hurdle that allows observers to safely bet that such rational methods for test planning will not be ready for use in the industry in the near future!

References

1. Aktouf, C., Fleury, H., Robach, C.: Inserting scan at the behavorial level. IEEE Des. Test Comput. **17**, 34–42 (2000)
2. Asaka, T., Bhattacharya, S., Dey, S., Yoshida, M.: H-SCAN+: a practical low-overhead RTL design-for-testability technique for industrial designs. In: Proceedings of ITC (1997)
3. Benso, A., Di Carlo, S., Di Natale, G., Prinetto, P., Lobetti Bodoni, M.: A programmable BIST architecture for clusters of multiple-port srams. In: Proceedings of International Test Conference, pp. 557–566 (2000). doi:10.1109/TEST.2000.894249
4. Bhattacharya, S., Dey, S.: H-SCAN: a high level alternative to full-scan testing with reduced area and test application overheads. In: Proceedings of VLSI Test Symposium (1996)
5. Fonseca, C., Fleming, P.: An overview of evolutionary algorithms in multiobjective optimization. Evol. Comput. **3**(1), 1–16 (1995). doi:10.1162/evco.1995.3.1.1
6. Huang, Y., Tsai, C.C., Mukherjee, N., Samman, O., Cheng, W.T., Reddy, S.M.: Synthesis of scan chains for netlist descriptions at RT-level. J. Electron. Test. **18**(2), 189–201 (2002)
7. Mazumder, P., Chakraborty, K.: Testing and Testable Design of High-Density Random-Access Memories, vol. 6. Springer, Boston (1996)
8. Miyazaki, M., Yoneda, T., Fujiwara, H.: A memory grouping method for sharing memory BIST logic. In: Asia and South Pacific Conference on Design Automation, 6 p. (2006). doi:10.1109/ASPDAC.2006.1594763
9. Roy, S.: RTL based scan BIST. In: Proceedings of VHDL International User's Forum (VIUF) (1997)
10. Wang, L.T., Wu, C.W., Wen, X.: VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon). Morgan Kaufmann Publishers Inc., San Francisco (2006)
11. Zaourar, L., Chentoufi, J., Kieffer, Y., Wenzel, A., Grandvaux, F.: A shared BIST optimization methodology for memory test. In: 15th IEEE European Test Symposium (ETS), p. 255 (2010). doi:10.1109/ETSYM.2010.5512736
12. Zaourar, L., Kieffer, Y., Aktouf, C.: An innovative methodology for scan chain insertion and analysis at RTL. In: 20th Asian Test Symposium (ATS), pp. 66–71 (2011). doi:10.1109/ATS.2011.20
13. Zaourar, L., Kieffer, Y., Aktouf, C.: A graph-based approach to optimal scan chain stitching using RTL design descriptions. In: VLSI Design (2012). doi:10.1155/2012/312808
14. Zaourar, L., Kieffer, Y., Wenzel, A.: A complete methodology for determining memory BIST optimization under wrappers sharing constraints. In: 3rd Asia Symposium on Quality Electronic Design (ASQED), pp. 46–53 (2011). doi:10.1109/ASQED.2011.6111701
15. Zaourar, L., Kieffer, Y., Wenzel, A.: A multi-objective optimization for memory BIST sharing using a genetic algorithm. In: IEEE 17th International On-Line Testing Symposium (IOLTS), pp. 73–78 (2011). doi:10.1109/IOLTS.2011.5993814
16. Zorian, Y.: A distributed BIST control scheme for complex VLSI devices. In: Eleventh Annual IEEE VLSI Test Symposium, Digest of Papers, pp. 4–9 (1993). doi:10.1109/VTEST.1993.313316

Part II
Network Design

Chapter 7
Low-Power NoC Using Optimum Adaptation

Sayed T. Muhammad, Rabab Ezz-Eldin, Magdy A. El-Moursy and Amr M. Refaat

Abstract Two power-reduction techniques are exploited to design a low leakage power NoC switch. First, the adaptive virtual channel (AVC) technique is presented as an efficient way to reduce the active area using a hierarchical multiplexing tree of VC groups. Second, power gating reduces the average leakage power consumption of the switch by controlling the supply power of the VC groups. The traffic-based virtual channel activation (TVA) algorithm is presented to determine traffic load status at the NoC switch ports. The TVA algorithm optimally utilizes virtual channels by deactivating idle VC groups to guarantee high leakage power saving without affecting the NoC throughput.

Keywords SoC · NoC · AVC · Power gating · Leakage power reduction · High throughput

7.1 Introduction

Systems-on-Chip (SoC) is introduced to offer high performance solution to satisfy the increasing communication demands of complex VLSI circuits [1]. SoC provides high productivity by reusing predefined and preverified Intellectual Property cores (IPs) [2]. SoC used to interconnect IPs through busses but shared medium busses could harm the throughput. As the complexity of SoC increases, limitations on system scalability, operating frequency, and power dissipation are becoming major problems. Large SoC causes significant increase in interconnection requirements which leads to large power consumption and delay. Network-on-Chip (NoC) is a new generation of SoC which is proposed as a solution for the interconnection problem of large-scale

S.T. Muhammad · R. Ezz-Eldin
Department of Electrical Engineering, Beni-suef University, Beni-suef, Egypt

M.A. El-Moursy (✉)
Mentor Graphics Corporation, Cairo, Egypt
e-mail: magdyaelmoursy@gmail.com

A.M. Refaat
Department of Electrical Engineering, Fayoum University, Fayoum, Egypt

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_7

SoCs [3–6]. NoC is composed of processing elements interconnected by Network Interface (NI), switches, and communication channels [7–9]. NoCs achieve high scalability, reliability, and high performance of the on-chip interconnection wires [10]. In NoC, the traditional interconnections such as point-to-point wires and busses (which suffer from low resource utilization) are replaced with switches which route the data flits (packet segments act as flow control units [11]) from source to destination IPs [12]. NoC uses routing algorithms similar to computer networks (i.e., XY and Odd–Even algorithms [13, 14]). Splitting NoC switch input port into set of Virtual Channels (VC) enhances the network throughput [15, 16], which provides better utilization of the net bandwidth [17] and improves the latency in high traffic loads [18]. Higher number of VCs improves the throughput while trading-off area and power dissipation.

With the increased complexity of interconnection networks, power consumption becomes a major design constraint for NoC [19]. Power consumption is taking significant attention due to the rapid growing market of portable battery-powered devices. Power dissipation is inversely proportional to battery life [20]. Reducing power consumption extends battery lifetime. Longer battery life is an increasing technical challenge, where reducing the power dissipation with low impact on performance is a key design issue. Dynamic power and leakage power are the main components of power dissipation in NoCs. Reducing leakage power is taking a lot of attention since it is dominating the power dissipation in today's and tomorrow's technologies. Devices which stay idle for long time leak considerable amount of power. Leakage power represents a significant proportion of the total power dissipation of NoCs [21].

Many research articles discussed reducing NoC power [22–24]. Few of such researches achieved simultaneous low power dissipation and high throughput NoC. The stoppable clock technique is used to reduce the power dissipation of NoC by reducing the network interface power [25]. The stoppable clock module can transfer or stop the local input clock of each submodule to shut down the switching power when the module is not running. This technique is concerned with only the dynamic power, while leakage power is not considered. *Boomerang* is another power-saving methodology for NoC [26]. *Boomerang* wakes up the response buffers ahead of use. When the buffer gets empty, *Boomerang* deactivates a response buffer to save power dissipation. This technique considers only the internal buffers without enough details about the circuit implementation.

The main focus of this chapter is to present a low leakage power switch using two techniques. The proposed switch reduces the leakage power dissipation and the dynamic power of a network switch. The proposed switch allows efficient power gating to be employed to reduce power dissipation. Traffic-based Virtual Channel Activation (TVA) algorithm that adopts *AVC* is also proposed as shown in Fig. 7.1. *TVA* is an efficient algorithm to highly utilize the VC. *TVA* algorithm offers reducing the leakage power dissipation with negligible impact on the network throughput.

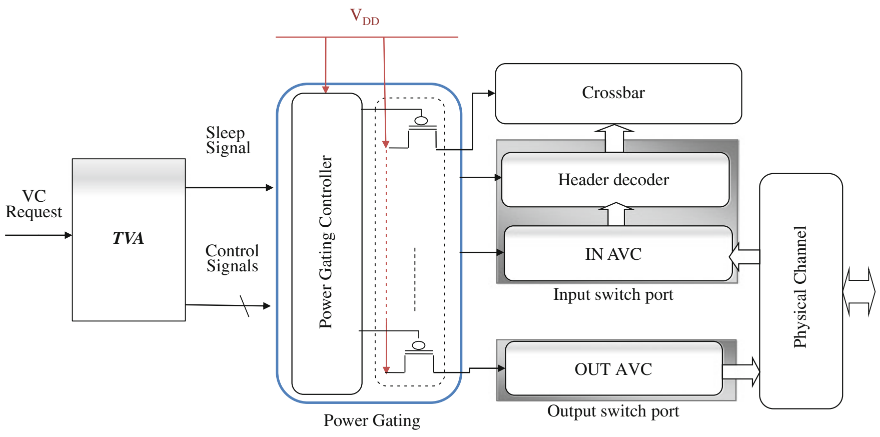


Fig. 7.1 Block diagram of NoC switch with *AVC* port

Contributions: This chapter has the following contributions:

- Introduce efficient techniques to reduce power dissipation of NoC switch.
- Introduce new algorithm for VCs activation/deactivation based on traffic load.
- Illustrate overhead of TVA algorithm.

The chapter is organized as follows: The proposed low leakage power switch is demonstrated in Sect. 7.2. *TVA* algorithm is presented in Sect. 7.3. Two types of simulators are employed on two levels of abstraction. Low-level hardware circuit simulator is used to analyze power dissipation as described in Sect. 7.4. Sensitivity analysis for *TVA* parameters along with simulation results are demonstrated in Sect. 7.5. Section 7.6 contains the architecture-level performance analysis. Finally, some conclusions are provided in Sect. 7.7.

7.2 Low Leakage Power Switch

The proposed switch is designed to use two leakage power reduction techniques which allow controlling each block individually to save power. Each port of the switch is designed using the proposed adaptive virtual channels technique as shown in Sect. 7.2.1. Switch port blocks are power gated to save leakage power as discussed in Sect. 7.2.2.

7.2.1 Adaptive Virtual Channel Technique

Adaptive Virtual Channel technique (*AVC*) is proposed to enable/disable the appropriate number of idle VCs of the switch port [27–29]. The optimum number of active

VCs is determined using NoC traffic status (as discussed in Sect. 7.4). *AVC* is used to configure IN/OUT ports. The number of available VCs is divided into power-of-two cells of configurable VCs. Each cell could be switched ON or OFF. *AVC* multiplexing tree where the VCs are located at the leaves of the tree and the physical port is located at the root ($level_n$) is illustrated in Fig. 7.2. Cells at $level_1$ are named Leaves Virtual Channel cells (*LVC*s) (i.e., $LVC_1 \ldots LVC_{2^n}$ Root Stage Cell (*RVC*) is the multiplexing cell which passes traffic to the physical channel. Stage Virtual Channel cells (*SVC*s) are the cells which connect *LVCs* to *RVC*. The tree is developed as a binary tree to optimize circuit implementation. The number of virtual channels equals 2^g where $g = m + n$. The number of VCs per *LVC* is $2^m \cdot n$ is the number of multiplexing levels. The number of *LVCs* in the tree is. $2^n \cdot m$ and n are positive integer numbers where $m \geq 1$ and $n \geq 0$. Every two cells in a low level of the tree are connected to one cell in the upper level. The total number of cells in the tree is given by

$$k = 2(2^n - 1), \tag{7.1}$$

The *RVC* consists of one multiplexer 2×1 and one grant circuit 2×2 as shown in Fig. 7.2. Every *SVC* of the tree consists of one arbiter 2×2 [30] and one multiplexer 2×1. *LVC*s consists of one multiplexer $2^m \times 1$ and one arbiter $2^m \times 2^m$. At the root, only one VC is granted the physical port. m and n introduce a single degree of freedom in designing the switch. The tree structure can be created with g different implementation options. m and n define a trade-off between circuit delay and configurability. For n equals zero, the tree contains only the *RVC* and all VCs operate simultaneously. Eliminating the multiplexing hierarchy reduces the circuit delay but the flexibility of configuring the VCs is minimum and no saving in power is possible as described in the following subsections. On the other hand, increasing the multiplexing levels n has negative effect on delay as described in the following

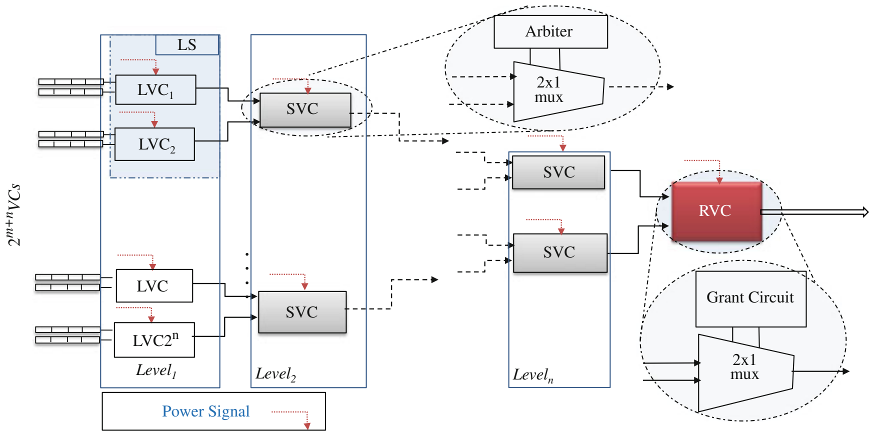


Fig. 7.2 Switch port using AVC technique

section. Tree structure hardware implementation is optimized to enhance the area of switching circuitry with increasing *n* as described in Sect. 7.2.2.

*LVC*s are grouped into *LVC* sets (*LS*). Each *LS* includes *LVC*s connected to certain *SVC*. Multiplexing tree activation/deactivation is *LS*-based process. Inactive VCs are power gated to reduce the leakage power dissipation as described in Sect. 7.2.2. *n* should be maximized to maximize the circuit configurability, minimize circuit area, and maximize the power saving as describe in Sect. 7.4.

7.2.2 Power Gated NoC Switch

Power dissipation of the switch could be defined as the summation of power dissipation of the crossbar, input port, and output port. The total power dissipation of each switch component is the summation of leakage ($P_{leakage}$), dynamic ($P_{dynamic}$) and short-circuit power (P_{SC}), and is defined as follows: [31–33]

$$P_{total} = P_{leakage} + P_{dynamic} + P_{SC}, \tag{7.2}$$

$$P_{dynamic} = \alpha.C_l.V_{dd}^2.f_{CLK}, \tag{7.3}$$

$$P_{leakage} = I_{leakage}.V_{DD}, \tag{7.4}$$

where α is the switching activity factor. $P_{dynamic}$ is proportional to the product of the load capacitance C_l, the square of voltage supply V_{DD} and the operating frequency f_{CLK}. $P_{dynamic}$ can be reduced by decreasing the total load capacitance. $P_{leakage}$ is directly proportional to the leakage current $I_{leakage}$. Power gating is an effective technique to reduce $P_{leakage}$.

Power Gating (*PG*) is used to enable/disable block(s) of the operating port according to the operating mode (input or output). Assuming that the switch has ℓ ports, p of them are set to operate as input ports and q are set to operate as output ports. Each port consists of input adaptive virtual channels (INAVC), header decoder, output adaptive virtual channels (OUTAVC), and *PG* as shown in Fig. 7.1. Each block in the switch port is power gated using one or more power gating circuitry. Header decoder and crossbar require only one power gating switch for each to be activate/deactivate. However, *IN/OUT AVC* requires *k* power gating switches for each one as shown in Fig. 7.3.

Power gating is exploited to configure the switch ports to active/inactive modes. PG activates the *INAVC* and header decoder at the input mode as highlighted in Fig. 7.3. Therefore, the *OUTAVC*s are deactivated. The crossbar switch is activated during input mode. For the output mode, only the *OUTAVC* is activated. *PG* is employed to perform two tasks, deactivate the virtual channels connected to cells for IN/OUT *AVC* and deactivate header decoder, crossbar, INAVC, and OUTAVC during the operation mode. Header decoder, crossbar, and each cell in IN/OUT AVC has one power gating switch. Power management is performed using a PG controller in addition to the sleep transistors. The flexibility of enabling/disabling the switch ports and activating only the operating blocks during the required mode increases the power

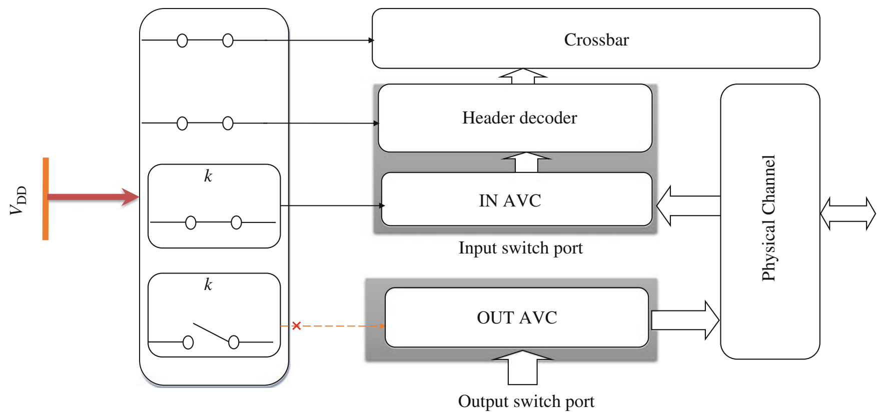


Fig. 7.3 Switch port using power gated NoC switch

saving. In Sect. 7.2.2.1, the sleep controller is described. Circuit implementation of power switching is presented in Sect. 7.2.2.2.

7.2.2.1 The Sleep Controller

Sleep controller is used to control activating /deactivating the switch port, configure the switch port as input/output mode and activate the virtual channels during the operating mode. The controller manages the sleep transistor of each cell, as well as two sleep transistors for header decoder and crossbar of each switch port. The sleep controller unit has three inputs and two output signals as shown in Fig. 7.4. A *ctrl_sleep* signal is used to enable/disable the switch port. This signal turns OFF the whole switch. The *IO_req* signal configures the switch port in input or output operating modes. The *nt* signal (*i* bits) indicates the status of NoC traffic.

The output signals *inp_ctrl/out_ctrl* are used to control the sleep transistors of IN/OUT AVC. Header decoder and crossbar are turned ON when the switch is in active mode and the switch port is operating as input port. It is required to find a general canonical representation of the truth table for the first level of the *inp_ctrl/out_ctrl* of the sleep controller. Taking into consideration that the number of bits for inputs and outputs changes depending on the traffic heaviness, the number of controlled *LVC*s is indexed as shown in Fig. 7.2. Therefore, the upper cell in of the tree has the lowest index of zero. The index increases going from top to bottom. The index of the bottom *LVC* in $level_1$ is $(2^i - 1)$. The equation is algebraically expressed in a sum of minterms form and it is considered as a general form for two signals *inp_ctrl/out_cntrl*

$$inp\_ctrl_x(nt_o, nt_1, \ldots nt_{i-1}) = \Sigma(0, 1 \ldots, x-1) \quad , 1 \leq x \leq 2^i - 1, \qquad (7.5)$$

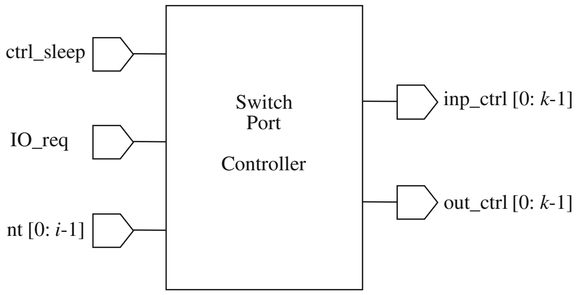


Fig. 7.4 The switch port controller block diagram

where x is the index of the *LVC*. This equation produces $(2^i - 1)$ columns of the output truth table, each column controls one *LVC* in $level_1$. At $x = 0$, the column of inp_cntrl_0 equals zero which means always activate LVC_1 regardless of the traffic load. Turning on a child *LVC* in $level_1$ requires activating all parents *SVC* of this child. Therefore, the truth table of every *LVC* which has even index is the same truth table of all parent *SVCs* in the same path from $level_1$ to *RVC*.

The output signals *inp_ctrl/out_ctrl* has k bits depending on the number of cells in the switch port as shown in Fig. 7.4. The *inp_cntrl* and *out_cntrl* signals are used to manage sleep transistors of IN and OUT *AVC*, respectively, according to the required number of VCs to be activated. Depending on the value of the *inp_ctrl/out_ctrl* signals, some sleep transistors are switched ON to activate its connected cells. The other sleep transistors are switched OFF to ensure that the connected cells are deactivated.

When *ctrl_sleep* signal is 1, all the bits of the *inp_ctrl/out_ctrl* are 1 and hence all the sleep transistors are switched OFF. Thereby, the switch port is forced to turn OFF regardless of the value of the *IO_req* and *nt*. When *ctrl_sleep* signal is 0 and *IO_req* signal is 1, the sleep controller activates input *AVC*, header decoder and crossbar. When *ctrl_sleep* signal is 0 and *IO_req* signal is 0, the sleep controller activates the output *AVC* and switches OFF the other blocks.

During input/output modes, the sleep controller calculates the value of the *inp_ctrl* and *out_ctrl* signals according to input signal *nt* which activates certain number of virtual channels. Activating VCs depends on the traffic heaviness which can take different levels. The number of traffic heaviness levels depends on granularity of activating the VCs which equals 2^i. For example, for number of VCs of eight and for $m = 1$, there are four levels of traffic heaviness ($i = 2$), “Very Heavy,” “Heavy,” “Light,” and “Very Light.” For $m = 2$, there are only two levels of traffic heaviness, “Heavy” and ”Light.” With very heavy traffic profile, all VCs are activated by switching ON all cells.

The granularity of activating the VCs is 2^m. For $m = 1$, binary multiplexing tree is used and two VCs are activated at a time. For $m = 2$, four VCs are activated simultaneously. For small m, large power saving is achieved since power gating could be applied with higher granularity. On the other hand, as m increases, the depth of the multiplexing tree decreases reducing the area overhead and the critical path delay. The proposed hierarchical switching increases the flexibility of activating the VCs making the switch more adaptive to the changes in the traffic characteristics. Accordingly, reducing *m* increases the power saving. Activating the VCs is achieved using the power switching presented in Sect. 7.2.2.2.

7.2.2.2 Circuit Implementation of Power Switching

The power switching block consists of $2(k+1)$ sleep transistors. In the proposed architecture, the sleep transistors are implemented by PMOS transistors to gate the power supply. Sleep transistor acts as a switch to turn OFF the supply voltage during the sleep mode. On the other hand, sleep transistors in the active mode are ON and hence the value of the virtual supply node is V_{DD}. Sizing the sleep transistor affects both circuit performance as well as the efficiency of power saving. There is a trade-off between power saving and performance in sizing the sleep transistor. During the active mode, the sleep transistor impedes the flow of the supply current. The transistor is required to be up-sized to keep circuit performance. On the other hand, sizing up the sleep transistor reduces its ability to mitigate the leakage current and power. In addition, the *PG* control circuit dissipates more dynamic power with larger sleep transistor.

The traffic heaviness signal *nt* is assumed to arrive to the target switch one clock cycle before the actual cycle at which the signal is needed to activate the cells. This assumption allows only one clock cycle to switch the cell from sleep-to-active mode. The switching time from sleep-to-active (*TSA*) must be less than or equal to the critical path delay t_d of the cell circuitry. The cell is considered active when its virtual supply voltage node reaches 90 % of. Equation (7.6) is used as the most tight design constrain to size the sleep transistors

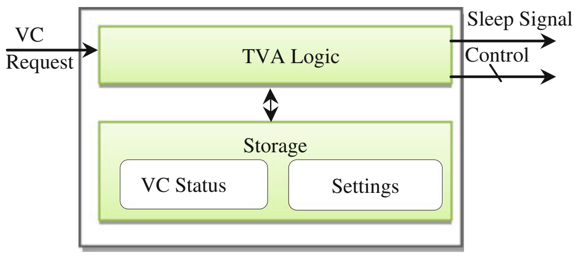
$$TSA \leq t_d, \tag{7.6}$$

Sleep signal and control signals are sent by algorithm that evaluates traffic status and consequently change signals' values. The algorithm is introduced in Sect. 7.3.

7.3 Traffic-Based Virtual Channel Activation Algorithm

The traffic load of any NoC varies up and down according to applications running on IP cores [34, 35]. Traffic variation is reflected on the number of VCs requests that arrive to Virtual Channel Allocator (VCA). In order to efficiently utilize VCs, idle VCs at any time should be powered OFF. The challenge is to adaptively control the PG unit according to the traffic variation. *TVA* is designed to efficiently achieve this goal [36]. *PG* block controls the delegation of power signal from the supply to each and every VC via *PG* controller. *TVA* algorithm lies at the heart of *VCA*. *TVA* employs *VCA* to control the *PG* according to incoming traffic load via passing control signals

Fig. 7.5 Components of TVA algorithm



and sleep signal to *PG* controller. As illustrated in Fig. 7.5, *TVA* is mainly composed of storage and logic.

Storage stores NoC settings, *LVC*s and *LS*s status. *TVA* logic determines the traffic status of the switch according to the amount of requests for *VCA*. It enables the required number of *LS*s to serve the incoming requests and deactivates idle sets. *TVA* parameters and variables as defined and discussed in Sect. 7.3.1. *TVA* procedures are presented in Sect. 7.3.2.

7.3.1 TVA Parameters and Variables

TVA defines a set of parameters and variables to measure traffic heaviness. Parameters hold design-time values and declare some architecture settings. Parameters are fully accessible at design-time and remain the same at run-time. Variables are used locally to hold run-time values (i.e., counters) which give feedback about the traffic status at any point of time. Variables are not accessible at design time.

7.3.1.1 TVA Parameters

The network is parameterized to optimize the algorithm for minimum power while maintaining the throughput. The defined network parameters are:

startupType: denotes how NoC starts up. Two startup options are proposed for initializing NoC. Assuming "*v*" *LVCs*
"*HP*": Highest Performance at startup by enabling all the *v LVCs* at startup.
"*LP*": Lowest Power dissipation at startup by deactivating whole the binary tree.
HPTime: denotes High Performance Time, this parameter is used only when $startupType = HP$, during this time all VCs are ON, so if $HPTime = \infty$, switch port operates as if *TVA* does not exist which represents the base line design point.
vcsPerCell: denotes the number of virtual channels per *LVC*.
numVCsPerPort: denotes number of virtual channels per input port.
lvcsPerSet: denotes number of *LVCs* per *LS*.

Also, two maximum waiting periods are defined

maxWP: which denotes the traffic maximum waiting period for VC to be released. It should be less than the time to switch-ON a cell.
maxIdlP: denotes maximum time for the cell to be idle.

VCA should wait before activating *LS* to serve a traffic for a time less than or equal *maxWP*. Also, before switching-OFF idle *LSs*, *VCA* should wait for time bounded by *maxIdlP*.

7.3.1.2 TVA Internal Variables

The variables in *TVA* could be divided into two types; architecture variables and performance variables. Architecture variables are related to the network architecture. Performance variables control the network efficiency and behavior.

(a) Architecture variables:

Two tri-state numerical flags indicate *LVC* and *LS* status
vcCellStatus: denotes *LVC* status. It has the value of 0 if *LVC* is *idle* (all cell VCs are *idle*), 1 if *LVC* is *busy* (all cell VCs are in traffic service), and 2 if *LVC* is partially busy *pBusy* (some VCs are in service and some are out).
lsStatus: indicates *LS* status. It has the value of 0 if *LS* is *idle* (all cell *LVCs* are *idle*), 1 if *LS* is busy (all cell *LVCs* are *busy*), and 2 if *LS* is partially busy *pBusy* (some *LVCs* are *pBusy*).

Six numerical variables are used to keep track of the number of *LVCs* and *LSs* which have different status (*idle*, *pBusy* or *busy*)

numIdleL : *number of idle LVCs.*	*numBusyL* : *number of busy LVCs.*
numPBusyL : *number of pBusy LVCs.*	*numIdleS* : *number of idle LSs.*
numBusyS : *number of busy LSs.*	*numPBusyS* : *number of pBusy LSs.*

(b) Performance variables:

Two numerical variables are used as counters for *VCA* waiting periods.
idleP: idle period, denotes how long the *LS* remains idle.
tWP: traffic waiting period, denotes time for the incoming traffic waiting for free channel, *TVA* instantiates a *tWP* variable for each incoming traffic waiting for service.

Statistical analysis for different *TVA* variables is introduced in Sect. 7.6. Traffic heaviness of any switch is to be determined by evaluating *idleP* and *tWP*. If no flits arrive to switch ports for *maxIdlP*, idle *LSs* is switched-OFF to save leakage power. If traffic arrives while there is no active free channel, traffic waits for busy channel to be released then no further *LS* are activated. If the waiting time reaches *maxWP*, an inactive *LS* is activated. Switching-OFF idle virtual channel cells has minimum effect on throughput while reducing leakage power. Flowcharts are provided in the following section to demonstrate the algorithm.

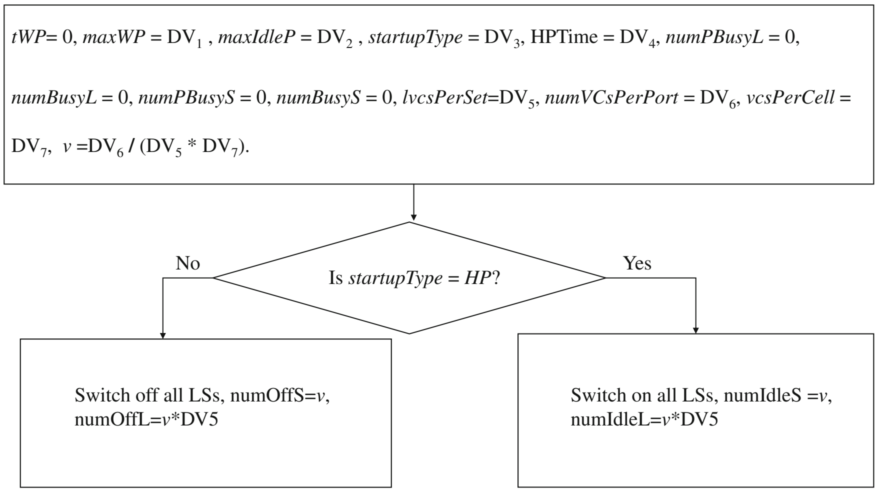

tWP= 0, maxWP = DV1 , maxIdleP = DV2 , startupType = DV3, HPTime = DV4, numPBusyL = 0, numBusyL = 0, numPBusyS = 0, numBusyS = 0, lvcsPerSet=DV5, numVCsPerPort = DV6, vcsPerCell = DV7, v =DV6 / (DV5 * DV7).
Is startupType = HP?
No
Yes
Switch off all LSs, numOffS=v, numOffL=v*DV5
Switch on all LSs, numIdleS =v, numIdleL=v*DV5

Fig. 7.6 Startup procedure flowchart

7.3.2 TVA Procedures

The proposed algorithm can be divided into four procedures; Startup (algorithm initialization), VC Request Handling, VC Release Handling, and Post-Allocation Procedure (PAP). The details of each procedure are described below:

7.3.2.1 Startup

The algorithm initiates both architecture and performance variables according to *startupType* value as shown in Fig. 7.6. It either switches ON or switches OFF all cells. Data Values DV1 and DV2 are design-dependent waiting periods which are assigned to *maxWP* and *maxIdleP*, respectively.

Both values are in the range from 0 to the time required to activate a cell and allocate one of its VCs to traffic DV_3 is the type of startup option.*HPTime*, *lvcsPerSet*, *numVCsPerPort* and *vcsPerCell* are initialized by DV_4, DV_5, DV_6, and DV_7. Total number of cell sets v is initiated by $\frac{\text{numVCsPerPort}}{\text{vcsPerCell}*\text{lvcPerSet}}$.

7.3.2.2 VC Request Handling

Whenever traffic reaches the switch input port requesting VC, request handling takes one of the following paths, as shown in Fig. 7.7. If there is a free VC in an active *pBusy LS*, the VC is allocated to the traffic. If there is no free VC (only busy *LS*s), the traffic waits for time less than or equal *maxWP*. If a VC is released during waiting

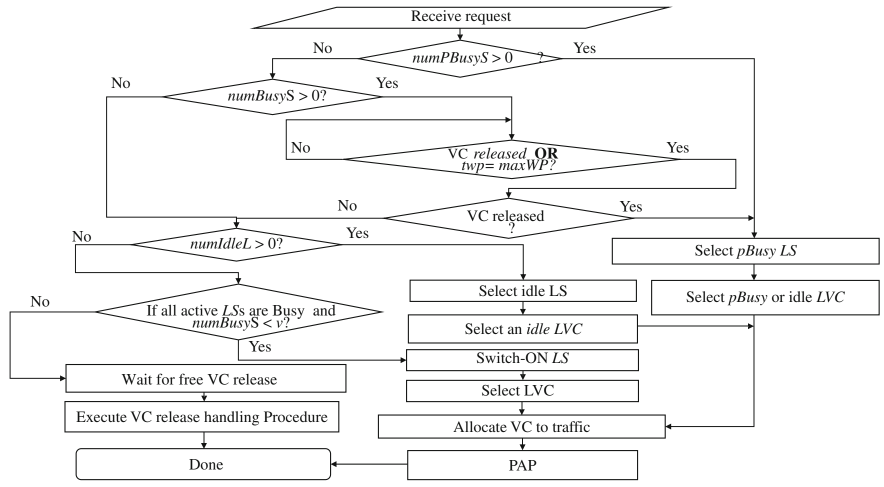


Fig. 7.7 VC request handling procedure flow chart

time, traffic is interrupted. The VC is allocated to that traffic. If there is no VC to be released, the traffic waits till *maxWP*. In this case, the *VCA* searches for *idle LS*. If *idle LS* is found, the traffic is allocated to one of its VCs. If the *VCA* does not find an *idle* cell, it searches for OFF *LS*. If there is inactive *LS*, *VCA* activates set and allocates one of the VCs to the traffic. If no OFF-*LS*s are found, the traffic waits for VC to be released. If any VC request arrives while previous VC request handling is in progress it waits for VC to be released and queued (first come first served). After every VC allocation, *PAP* is executed.

7.3.2.3 VC Release Handling

As illustrated in Fig. 7.8, whenever a virtual channel becomes traffic-free, it is allocated to a waiting traffic and the architecture variables are updated. If there is no waiting traffic, the *PAP* is executed to update the status. After that there are two possibilities. If the *LS* status is idle and *startupType* equals *HP*, *LS* waits for further traffic requests within *HPTime*.

If *HPTime* passes and *LS* is still idle or startupType is *LP*, *LS* waits for traffic. When *maxIdlP* time passes without allocating the VC to incoming flits, the cell set is switched-OFF. When the *SVC* has no active *LVC*s connected to its input, it is switched-OFF as well. Hence, the full path to *RVC* becomes OFF to achieve maximum power saving. If the *LS* status is not idle after VC is released, nothing happens and the cell set is ready for further traffic allocation requests. When *LS* becomes idle after starting up port with *HP* case (all *LS*s are ON), *VCA* waits for *HPTime* then counts *maxIdleP* to switch-OFF the *LS*. ***X-Idle*** node in the flow chart

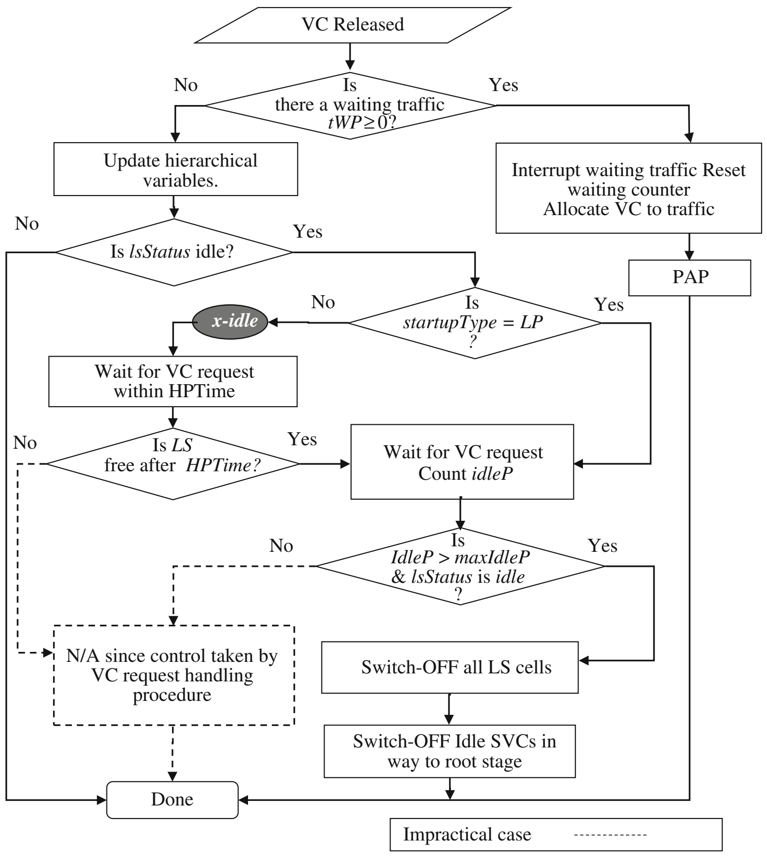

VC Released
Is there a waiting traffic tWP≥0?
No
Yes
Update hierarchical variables.
Interrupt waiting traffic Reset waiting counter Allocate VC to traffic
PAP
Is lsStatus idle?
No
Yes
Is startupType = LP ?
No
Yes
x-idle
Wait for VC request within HPTime
Is LS free after HPTime?
No
Yes
Wait for VC request Count idleP
Is IdleP > maxIdleP & lsStatus is idle ?
No
Yes
N/A since control taken by VC request handling procedure
Switch-OFF all LS cells
Switch-OFF Idle SVCs in way to root stage
Done
Impractical case

Fig. 7.8 Virtual channel release handling procedure flow chart

is the start point of a sub-procedure which runs only once after startup to cover the case when *LS* becomes idle after starting up port with *HP* (all VCs are ON).

7.3.2.4 Post-Allocation Procedure

PAP runs after VC allocation to update both architecture and performance variables and terminate VC requests. *PAP* transition diagram is shown in Fig. 7.9. *PAP* is composed of two sub-procedures which run sequentially for updating *LVCs* variables and *LSs* variables. At the cell level, if the number of *idle* VCs equals *vcsPerCell*, the *LVC* becomes idle (*vcCellStatus* = 0). If the number of in-service VCs equals *vcsPerCell*, *LVC* becomes *busy* (*vcCellStatus* = 1). Otherwise, the *LVC* becomes pBusy (*vcCellStatus* = 2). *numIdleL*, *numBusyL* and *numPBusyL* are updated according to *LVC*s status (e.g. if *LVC* status is *idle* and become *busy*, *numBusyL* is incre-

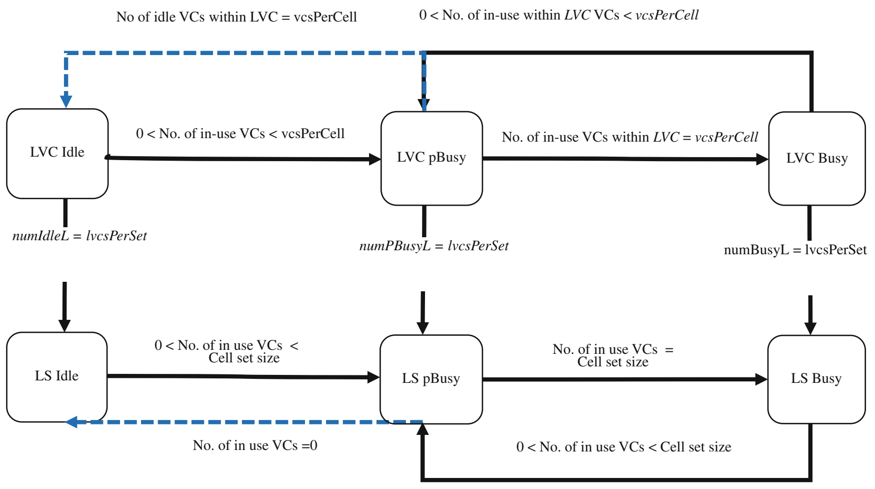

No of idle VCs within LVC = vcsPerCell
0 < No. of in-use within *LVC* VCs < *vcsPerCell*
LVC Idle
0 < No. of in-use VCs < vcsPerCell
LVC pBusy
No. of in-use VCs within *LVC* = *vcsPerCell*
LVC Busy
numIdleL = *lvcsPerSet*
numPBusyL = *lvcsPerSet*
numBusyL = lvcsPerSet
LS Idle
0 < No. of in use VCs < Cell set size
LS pBusy
No. of in use VCs = Cell set size
LS Busy
No. of in use VCs =0
0 < No. of in use VCs < Cell set size

Fig. 7.9 *PAP* state transition diagram

mented and *numIdleL* is decremented). At the set level, if the number of *idle LVCs* equals *lvcsPerSet*, the *LS* becomes *idle* (*lsStatus* = 0). If the number of *busyLVC*s equals *lvcsPerSet*, *LS* becomes *busy* (*lsStatus* = 1). Otherwise, *LS* becomes *pBusy* (*lsStatus* = 2). *numIdleS*, *numBusyS* and *numPBusyS* are updated according to *LS*s status. Different circuit-level implementation options of *AVC* and *PG* are introduced in the upcoming section as well as the circuit-level simulation results (i.e., area and leakage power dissipation).

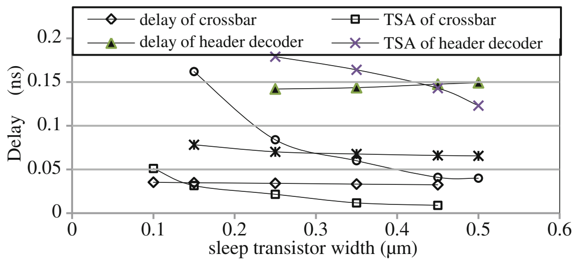
7.4 Circuit-Level Implementation

The proposed technique is implemented using the ADS tools. 45 nm technology is used with supply voltage of 1 V. In Sect. 7.4.1, sizing the sleep transistors for each block of switch port is presented. Different implementation of *IN*/*OUT AVC* and the reduction in the leakage power with each implementation are discussed in Sect. 7.4.2. In Sect. 7.4.3, the leakage power dissipation of the main components of switch port and total power dissipation for conventional and proposed NoC switch are presented. The saving in leakage power dissipation of the proposed switch with different number VCs is provided in Sect. 7.4.4.

7.4.1 Sleep Transistors Sizing

A switch port with VCs is considered. The trade-off between *TSA* switching time and the critical path delay of the cell is presented in Fig. 7.10. To increase power

Fig. 7.10 Critical path delay and *TSA* for different sleep transistor widths of different components of a switch port



saving, the sleep transistor needs to be sized down. On the other hand, *TSA* could not be larger than t_d since only one clock cycle is allowed to switch the cell from sleep-to-active mode. More relaxed timing constraints could achieve larger reduction in leakage power. However, the minimum degradation of performance is the target of our proposal.

The intersection point on t_d and *TSA* curves is used as the optimum size for high-performance and low-power switch design. Based on that, the width of sleep transistors of different components of switch port is determined to be, 0.15, 0.45 and 0.35 μm for crossbar, header decoder and each cell of *IN/OUT AVC*, respectively.

7.4.2 Depth of the Multiplexing Tree IN/OUT AVC

For eight VCs, there are $g = 3$ implementation options

$$\begin{cases} option\ a: m = 3,\ n = 0,\ k = 0 \\ option\ b: m = 2,\ n = 1,\ k = 2 \\ option\ c: m = 1,\ n = 2,\ k = 6 \end{cases} \tag{7.7}$$

For option *a*, the tree structure consists of only the root. In option *b*, the available virtual channels are divided into two sets using four virtual channels per set. The available virtual channels are divided into four sets using binary multiplexing tree and two multiplexing levels in option *c*. There are a total of six cells in the tree where at least two can be simultaneously activated.

The required area to implement the three options, including the area of the sleep controller and sleep transistors, is shown in Fig. 7.11.

The area of the multiplexing tree of option *c* is less than the area of the multiplexing tree of option *a* and *b*. As compared to option *a*, the area decreases in option *b* by 48.37 and by 53.03 % for option *c*. The overhead in the input gate capacitance (sleep controller and sleep transistors) in option *c* is 22.49 % of the total port capacitance. In option *b*, the overhead is only 8.7 %. The area overhead increases the dynamic power dissipation. However, the overhead circuitry has less activity factor than the switch since the switching in the overhead circuitry occurs only during reconfiguration when

Fig. 7.11 The area of switch port for different number of virtual channel per one set

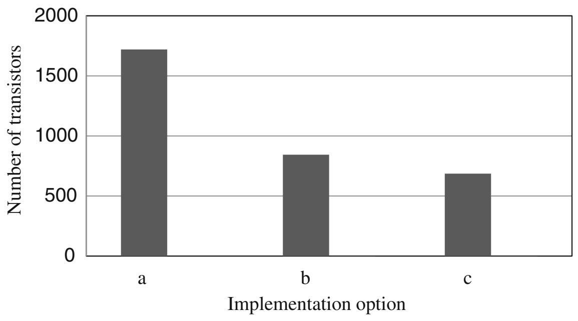


Table 7.1 Maximum operating frequency and leakage power for three implementation options

m	Maximum operating frequency (GHz)	Leakage power		
		Reduction (%)	P_{wt-pG} (nW)	$P_{max-red}$ (%)
3	18.99	–	2821.11	–
2	12.18	35.86	616.85	78.13
1	8.44	55.76	363.32	87.12

the traffic characteristics change. Accordingly, the overhead in dynamic power is less than 23 %.

The hierarchical tree implementation has twofold effect in reducing power dissipation of the switch. The leakage power is decreased with light traffic since power gating is more efficient. In addition, the dynamic power is reduced for the reduction in the input gate capacitance of the switch. With hierarchical multiplexing, dynamic power of the switch could decrease by up to 54 %. On the other hand, the hierarchical tree structure increases the critical path delay of the circuit reducing the maximum operating frequency. The maximum operating frequency and leakage power for the three implementation options are listed in Table 7.1.

Maximum power reduction $P_{max-red}$ is determined using (7.8), where $P_{max-leak}$ equals the leakage power without using *PG* when all VCs are OFF.

$$P_{max-red} = \frac{P_{wo-PG} - P_{wt-PG}}{P_{max-leak}}, \tag{7.8}$$

where, P_{wo-PG} is the leakage power without using *PG* and P_{wt-PG} is leakage power with using *PG*. The maximum operation frequency and leakage power decrease with increasing the number of levels. The leakage power for option *c* decreases by 87.12 % as compare to the leakage power of option *a*. A pipeline stage could be used to maintain the operating frequency but latency of switching would increase.

7.4.3 Power Dissipation of On-Chip Switch

Using the ADS tools, the leakage power of main component of switch port for active mode and sleep mode are determined as reported in Table 7.2. Using hierarchical multiplexing tree ($m = 1$ and $n = 2$), the power dissipation of one switch port while being active is 1.66 μW during input mode and 1.2 μW during output mode.

The leakage power for each switch port of the conventional NoC switch for $\ell = 6$, $g = 3$ without using hierarchical multiplexing tree ($m = 3$) is determined to be 36.26 μW. When the proposed switch is configured to operate with three input ports ($p = 3$) and three output ports ($q = 3$). The average leakage power of the proposed switch during active mode is 8.78 μW. The leakage power of the proposed switch is reduced by 75.7 % as compare to the power dissipation of the conventional NoC switch when the proposed techniques are employed.

7.4.4 Power Saving with Different Number of Virtual Channels

For total number of VCs of eight ($p = 3$), $m = 1$ and $n = 2$, as shown in Fig. 7.12, there are four levels of traffic heaviness. The network traffic is used to control the number of active VCs. Since $n = 2$, two, four, six, or eight VCs could be simultaneously activated depending on the traffic of the network. The reduction in leakage power dissipation is illustrated in Fig. 7.12 for different network traffic characteristics.

The power saving increases as the number of active VCs decreases. Power saving could reach up to 81 % when only two VCs are simultaneously activated. When no VCs are activated, power saving increases by up to 97 %. *AVC* with hierarchical multiplexing tree significantly decreases the power consumption of the switch.

To evaluate *AVC* with different port sizes, NoC with 16 VCs and 8 *LVC* (each has 2 VCs) is assumed. $P_{max-red}$ is calculated using Eq. (7.9), where $P_{max-leak}$ is determined to be 1963.2 nW. The power saving with inactive VC cells is determined

Table 7.2 The leakage power of the switch port components using the proposed technique

Switch port components	Leakage power		
	Active mode (nW)	Sleep mode (nW)	Reduction (%)
IN/OUT AVC	923.25	29.5	96.80
Header decoder	341.16	6.913	97.97
Crossbar	55.467	1.107	98.00
Total during input mode	1660	381.03	77.05
Total during output mode	1200	373.01	68.92

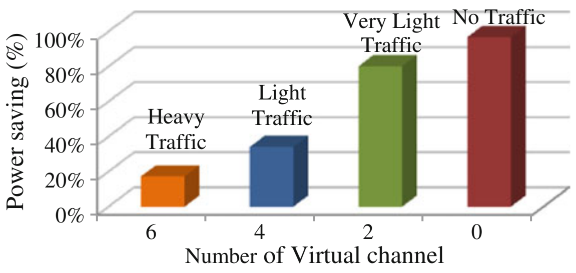


Fig. 7.12 The leakage power saving with different number of virtual channels

for even numbers VCs = 2, 4, 6, 8 . . . 16. Power numbers are reported in Table 7.3. P_{wo-PG} and P_{wt-PG} are listed in the second and third columns, respectively. Increasing number of inactive virtual channels reduces the leakage power dissipation.

According to *AVC*, at the circuit-level, the minimum cell set size is 2 cells. In the rest of the chapter "$LS = 1$" is used to refer to set size of 1 cell and "$LS = 2$" refers to $lvcsPerSet = 2$. The $LS = 1$ case effectively utilizes the VCs since it avoids activating useless *LVC*s with negative impact on throughput as discussed in next section. The hardware overhead to achieve $LS = 1$ case is high. Features of a simulator developed for *TVA* are introduced in next section in addition to algorithm operation states.

7.5 Implementation of TVA

Network Interconnect Routing and Application Modeling (NIRGAM) is a SystemC-based discrete-event, cycle accurate simulator which is used to simulate the NoC and determine the throughput and latency [37]. *NIRGAM* is extended to incorporate *AVC* using *TVA*. In order to apply the proposed algorithm, the number of virtual channels

Table 7.3 Leakage power with and without *PG* for different number of VCs

Number of active VCs	Leakage power		
	P_{wo-PG} (nW)	P_{wt-PG} (nW)	$P_{max-red}$ (%)
14	165.2	4.8	8.2
12	331	14.7	16.1
10	768.3	19.5	38.1
8	936.2	29.5	46.2
6	1266.9	34.4	62. 8
4	1399.5	43.9	69.1
2	1732.6	47.2	85.9
0	1963.2	59	97

per switch needs to be changed according to traffic heaviness. The simulator code is re-structured and re-developed to meet our design target. The new simulator for *AVC* is named (NIRGAM_AVC). *NIRGAM_AVC* has the ability to configure the number of VCs per switch port, *maxWP*, *maxIdleP*, *vcsPerCell*, *lvcsPerSet*, and *startupType*. Different leakage power levels are made reconfigurable (i.e., leakage power of 4, 8, and 12 VCs) which is another feature added to *NIRGAM* in *NIRGAM_AVC*. The developed simulator calculates the overall power of the NoC and the power reduction $P_{max-red}$ using (7.9) which means that all calculations are provided using real circuit-level simulation.

The major feature in *NIRGAM_AVC* is the number of active VCs that changes according to VC requests. *TVA* operates in two operating states; *transient state* and *steady state*. *Transient state* begins when *TVA* launches (time $= 0$) with all *LS*s are ON or OFF according to *startupType*. *startupType* takes two values; 0 for *HP* and 1 for *LP*. Startup type of *TVA* has perceptible effect on performance and leakage power saving. It adds flexibility to *TVA* to meet high performance design needs. *LP* startup makes the ports start with zero active *LVC*s and starts to switch-ON *LS*s upon VCs requests. As NoC runs, more *LS*s are switched-ON/OFF to serve incoming traffic. Hence, the variation in the number of active VCs is large. *Transient state* is identified by the high number of switching times of *LS*s which is considered an overhead. On the other hand, reducing the number of idle *LS*s saves more leakage power. Steady state starts when the change in the number of active VCs becomes relatively infrequent (i.e., around 10 % change in the considered NoC). *TVA* is auditing idle *LS*s and waiting traffic to decide to turn-ON or turn-OFF *LS*s. The variation of number of active VCs depends mainly on cell size and startup type. As the cell size increases, more VCs are activated. The number of active VCs versus simulation time for different cell sizes of the configured NoC for $LS = 2$ is shown in Fig. 7.13a for *LP startupType*.

HP startup type obligates the NoC to operate with full capacity of VCs during *HPTime*. Variation of the number of active VCs with simulation time for *TVA* with *HP* startup type is illustrated in Fig. 7.13b for different *HPTime* values. The number of active VCs remains constant until *HPTime* passes, then all *idle LS*s are deactivated after *maxIdleP* (which is configured to be 0 ns) plus one clock cycle (limited by circuit-level simulation). $LS = 1$ always employs less number of VCs to serve traffic.When $LS = 1$ is used, number of active VCs reaches *steady state* lower than when $LS = 2$. Since number of VCs to be activated (*startupType* $=$ *LP*) or deactivated (*startupType* $=$ *HP*) is the half of $LS = 2$ case.

Leakage power dissipation is tightly coupled with the alteration of the number of VCs. Leakage power savings for $LS = 2$ is shown in Fig. 7.14a for different cell dimensions with *LP startypType* and no idle time. High power savings are achieved at transient period when small number of VCs are in service. Steady state is characterized by the stable level of power dissipation since the change in number of active VCs is small. *HP* pledges that all VCs stay active for *HPTime* which cause zero leakage power savings during *HPtime* as illustrated in Fig. 7.14b for different values of *HPTimes*. *TVA* leakage power savings varies due to variation of number of inactive VCs. Transient state leakage power saving is either zero during *HPTime* (for *HPstartupType*) or extremely high (for *LPstartupType*).

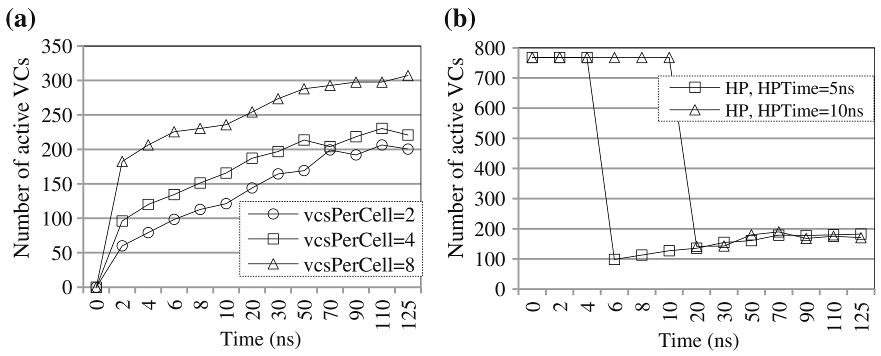


Fig. 7.13 Number of active VC for *LP* startup during simulation for 4 × 4 NoC with constant bit rate traffic. **a** *LP* with different cell sizes in *LS* = 2 **b** HP with different *HPtimes*

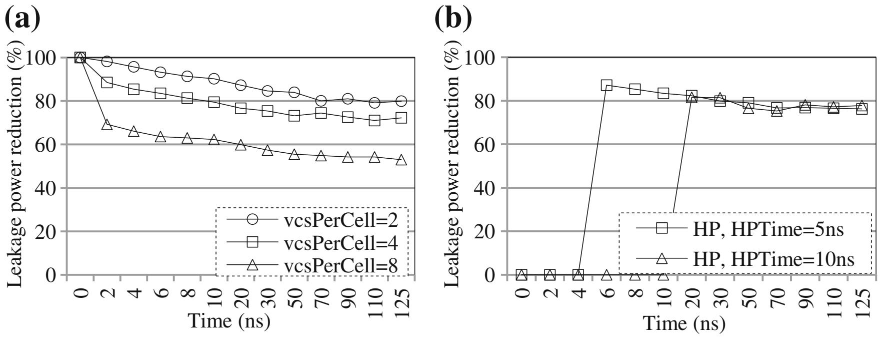


Fig. 7.14 Leakage power reduction during different states for 4 × 4 NoC with constant bit rate traffic. **a** *LP* different *vcsPerCellLS* = 2 case **b** *HP* with different *HPtimes*

NIRGAM_AVC calculates power values in steady state with step defined by a configurable parameter named *PWR_CALC_TIME_STEP* (i.e., in results section the steady state starts at 30 ns and *PWR_CALC_TIME_STEP* is configured to be 2 ns then power values are calculated for 30, 32, 34 ns …etc.). Average steady state leakage power saving is calculated as the average of the power values at different time steps. Hence, increasing *HPTime* beyond transient state causes significant degradation in steady state leakage power.

The network throughput without using *TVA* is considered as the baseline throughput to calculate throughput reduction using (7.9).

$$Th_{red} = \frac{Th_{wo-TVA} - Th_{wt-TVA}}{Th_{wo-TVA}}, \tag{7.9}$$

where, Th_{red} is throughput reduction, Th_{wo-TVA} is throughput without *TVA* and Th_{wt-TVA} is throughput with *TVA*. In next section, throughput overhead is discussed. *NIRGAM_AVC* new parameters are changed to show the efficiency of *TVA* and effect of each parameter on steady state leakage power and throughput reduction. Statistics gathered by *NIRGAM_AVC about TVA* variables are also shown and discussed.

7.6 TVA Simulation Results

Leakage power saving and throughput values of NoC which uses *TVA* are tightly coupled to different *TVA* parameters. This section illustrates the effect of those parameters on power savings and throughput. *TVA* is implemented using *NIRGAM_AVC* with the following NoC configuration: clock frequency of $8GHz$, and OE routing algorithm, constant bit rate (CBR) traffic and 16 VCs per port are assumed. Other *NIRGAM_AVC* new parameters are changed to show the efficiency of *TVA*. Without using *TVA*, the network throughput is considered as the reference throughput used to calculate throughput reduction using (7.9). In the rest of this section, results are demonstrated for 4×4(16 tiles), 8×8(64 tiles), and 16×16(256 tiles) networks. Throughput overhead of *TVA* is illustrated in Sect. 7.6.1. Startup types are discussed in Sect. 7.6.2. Variables statistics are shown in Sect. 7.6.3. Effect of port size, leaf cell size, and idle periods on throughput reduction and leakage power reduction are discussed in Sects. 7.6.4, 7.6.5 and 7.6.6 respectively.

7.6.1 Throughput Overhead

The circuit architecture guarantees a single clock cycle overhead to activate a VC set. Switching overhead degrades NoC throughput. The impact of *TVA* on throughput is determined in *NIRGEAM_*AVC by enumerating the number of times *LS*s are switched-ON then add it to the total number of clock cycles which is inversely proportional to throughput. Normalized *TVA* throughput ($Th_{TVA\text{-}Norm}$), for different *TVA* parameters for $LS = 1$ and $LS = 2$ is shown in Fig. 7.15, where, $Th_{TVA\text{-}Norm} = (Th_{wt\text{-}TVA} / Th_{wo\text{-}TVA})$. $Th_{TVA\text{-}Norm}$ of different cell sizes for *TVA* with $numVCsPerPort = 16$, $startupType$ = "*LP*" and $maxIdleP = 0$ is shown in Fig. 7.15a. $vcsPerCell = 2$ achieves $Th_{TVA\text{-}Norm}$ of 0.993 for $LS = 2$ and 0.938 for $LS = 1$ which means less than 7 % degradation in throughput due to the switching ON/OFF activity. Increasing *vcsPerCell* reduces the switching times and hence increases $Th_{TVA\text{-}Norm}$ which equals to 0.997 and 1 for $vcsPerCell = 8$ for $LS = 2$ and $LS = 1$, respectively. The effect of using different values of *LSmaxIdleP* is illustrated in Fig. 7.15b for *TVA* with $numVCsPerPort = 16$, $startupType$ = "*LP*" and $vcsPerCell = 2$. Higher cell set idle time causes low overhead since *LS*s stays ON longer. $Th_{TVA-Norm}$ for different port sizes is shown in Fig. 7.15c for *TVA* with $startupType = LP$, $maxIdleP = 10$ ns and $vcsPerCell = 2$. Increasing port size increases the probability of *idleLS*s which leads to additional overhead. $Th_{TVA\text{-}Norm}$ for $numVCsPerPort = 8$, 12 and 16 equals 0.997, 0.997 and 0.995, respectively for $LS = 2$ and 0.993, 0.996 and 0.996, re-

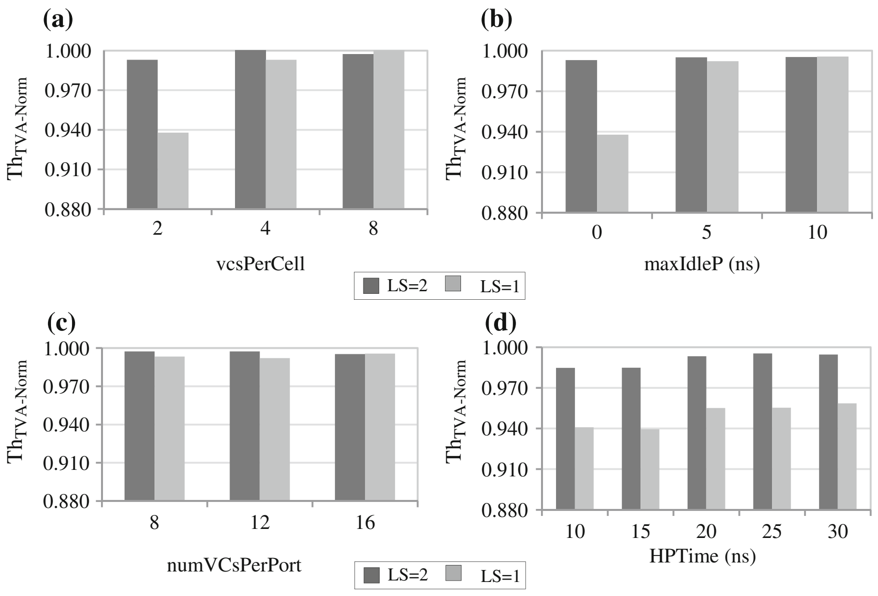


Fig. 7.15 Normalized *TVA* throughput for different *TVA* parameters of 2D-Mesh NoC **a** cell size **b** Cell set idle period **c** number of VCs per port **d** *HPTime*

spectively for $LS = 1$. The effect of changing *HPTime* when *startupType* is "*HP*" and $maxIdleP = 0$ ns for port size =16 NoC is shown in Fig. 7.15d. During *HPTime*, all *LS*s are ON which add zero overhead. Increasing *HPTime* decreases the overhead. *HPTime* of 10 ns corresponds to $Th_{TVA\text{-}Norm}$ of 0.985 for $LS = 2$. *HPTime* beyond transient state enhances the throughput. For $HPTime = 30$ ns, $Th_{TVA\text{-}Norm}$ is enhanced to reach 0.995 for $LS = 2$ and 0.959 for the $LS = 1$. Despite the existence of the overhead, *TVA* has negligible effect on the overall throughput of the NoC with maximum throughput degradation of 1.5 % for $LS = 2$.

$Th_{TVA\text{-}Norm}$ for 16 tiles, 64 tiles, and 256 tiles NoC for both $LS = 1$ and $LS = 2$ is illustrated in Fig. 7.16 for 2D-Mesh and 2D-Torus topologies. The 256 tiles NoC suffers from the highest throughput degradation of 0.88 and 2.65 % for $LS = 2$ and $LS = 1$, respectively, when 2D-Torus topology is used and 0.69, 2.39 % when 2D-Mesh topology is assumed. Other NoC dimensions have negligible throughput degradation, i.e., throughput degradation is less than 3 %.

7.6.2 *TVA Startup Type*

NoC average leakage power reduction and average throughput degradation for 16, 64, and 256 tiles 2D-Torus networks with different *startupTypes* are reported in Table 7.4 with $numVCsPerPort = 16$ and $vcsPerCell = 2$ for $LS = 2$. Increasing the

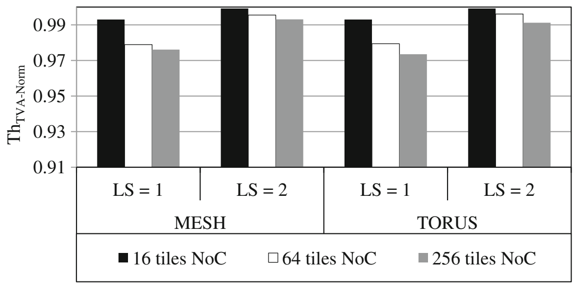


Fig. 7.16 Minimum Normalized *TVA* throughput of 16 tiles, 64 tiles, and 256 tiles NoC for $LS = 1$ and $LS = 2$ for 2D-Mesh and 2D-Torus topologies

Table 7.4 Average power savings and throughput reduction of *LP* and *HP* for 2D-Torus NoC

Startup type (ns)	16 tiles NoC		64 tiles NoC		256 tiles NoC	
	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)
LP, 0	82.98	0.08	78	−0.04*	72.02	0.43
HP, 5	82.98	0.08	78	−0.06*	72.02	0.41
HP, 15	84.06	0.03	78.27	−0.05*	72.08	0.24
HP, 25	84.06	0.03	78.18	0.31	72.19	0.55
HP, 40	73.76	0.03	68.57	0.32	62.93	0.88

*Negligible increase

HPTime enhances the power reduction. For $LS = 2$, highest leakage power savings of 84.1, 78.3, and 72.2 % are achieved for 16 tiles, 64 tiles and 256 tiles networks, respectively. Implementing $LS = 1$ improves the power reduction to 90, 83, and 78.6 % for 16, 64 and 256 tiles 2D-Torus NoC, respectively.

The results for NoC with the same configurations for 2D-Mesh topology are listed in Table 7.5. *HP startupType* achieves the highest leakage power reduction of 79.7, 75.6 and 70.6 % for 16 tiles, 64 tiles and 256 tiles NoCs, respectively. *startupType* achieves high leakage power reduction with negligible impact on NoC throughput. *HP startupType* achives higher power savings than *LP*. $LS = 1$ saves (5–9) % of leakage power above $LS = 2$.

Leakage power saving is significantly decreased for HP startup with *HPTime* greater than 30ns since steady state starts after 30ns (for the tested NoC) e.g. for $HPTime = 40$ ns no power saved at 30, 32, 34 ns . . . 40 ns. *HP* with *HPTime* less than the start of steady state achieves high power saving.

Table 7.5 Average power savings and throughput reduction of *LP* and *HP* for 2D-Mesh NoC

Startup	16 tiles NoC		64 tiles NoC		256 tiles NoC	
	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)
LP, 0	78.31	0.08	75.28	0.1	70.42	0.2
HP, 5	78.31	0.08	75.28	0.08	70.42	0.18
HP, 15	79.74	0.03	75.6	0.08	70.51	−0.02*
HP, 25	79.74	0.03	75.49	0.44	70.56	0.69
HP, 40	69.98	0.03	66.21	0.45	61.57	0.58

*Negligible increase

7.6.3 TVA Variables Statistics

TVA variables which represent the number of *LVC*s and *LS*s with difference status (i.e., *numIdleL, numBusyS*etc.) are presented in Sect. 7.3.1.2. Histogram charts for those variables are illustrated in Fig. 7.17. 2D-Mesh NoC with *LP* startup with $maxIdleP = 0$ ns, $LS = 2$ and $LVC = 2$. Number of occurrences (number of clock cycles) of *numIdleL*, *numBusyL*,*numPBusyL* and number of OFF *LVC*s in case of 16 tiles NoC is shown in Fig. 7.17a. *TVA* minimized *numIdleL* occurrence to 0.39 %, number of OFF cells occurrence to 83 % with mean equals to 96,008 cycles and standard deviation of 149,491cycles. *PAP* is executed only 34 times during 1000 cycles. 64 tiles NoC cell variables histogram is illustrated in Fig. 7.17b. *numIdleL* occurs 1.34 % while OFF *LVC*s occupy 80 % of the cycles with $\mu = 448{,}008$, $\sigma = 662{,}421$ and *PAP* execution frequency of 1026 per 1000 cycles. 256 tiles NoC histogram of *LVC* status variables is shown in Fig. 7.17c with $\mu = 1{,}920{,}010$, $\sigma = 2{,}268{,}030$ and PAP frequency equals 6393.

LS variables (*numIdleS*, *numBusyS* and *numPBusyS*) histograms are shown in Fig. 7.18a–c respectively. $LS = 1$ shows significant increase in *PAP* execution frequency, NoC has *PAP* frequency of 1930, 7086 and 26,564 in case of 16 tiles, 64 tiles and 256 tiles networks, respectively. For different NoC dimensions, *TVA* maximize the existence of OFF VCs to be (20–80)% of clock cycles which has positive effect on power saving.

7.6.4 Number of VCs per Port

NoC using *TVA* with 2D-Torus topology*, LP startuptype*, zero *maxIdleP* and $vcsPerCell = 2$ is assumed. Percentage steady state power and throughput reduction of different NoC dimensions for $numVCsPrtPort = 6, 8 \ldots 16$ is reported in

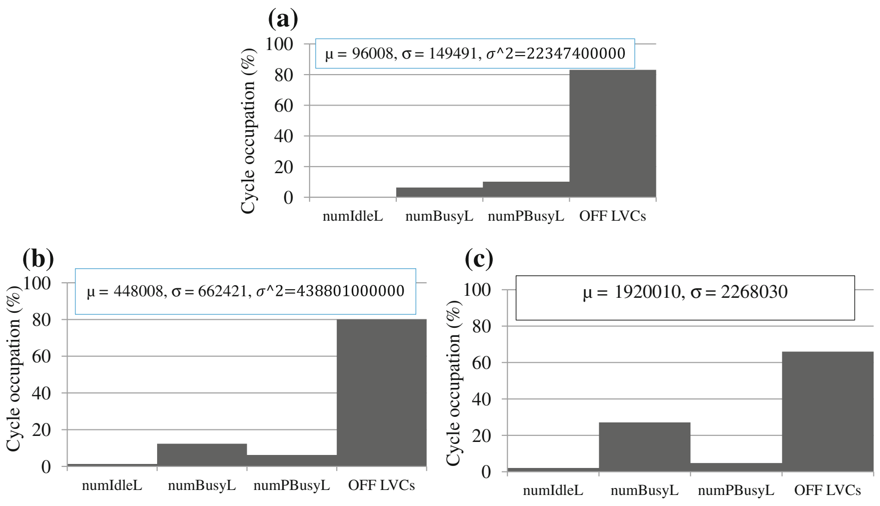


Fig. 7.17 Histograms, mean and standard deviation for different *TVA LVC* status variables of **a** 16 tiles NoC **b** 64 tiles NoC **c** 256 tiles NoC

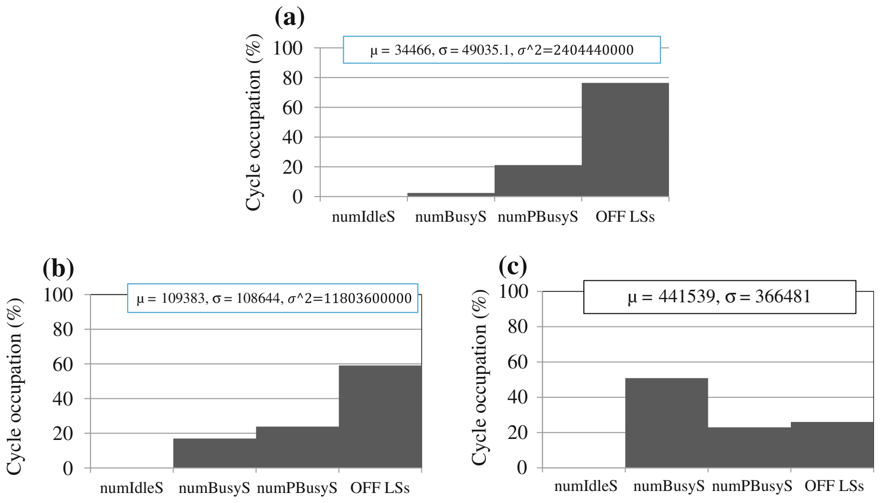


Fig. 7.18 Histograms, mean and standard deviation for different *TVA LS* status variables of **a** 16 tiles NoC **b** 64 tiles NoC **c** 256 tiles NoC

Table 7.6. *TVA* with *numVCsPerPort* = 14 achieves the highest leakage power savings of 83.7, 78.9 and 72.68 % for 16 tiles, 64 tiles and 256 tiles NoC, respectively.

With 14 VCs per port leakage power saving is more than that of 16 VCs port because in the 14 VCs port the VCs are divided into 3 *LS*s each contains 4 VCs and one *LS* has only 2 VCs where the 16 VCs port has 4 *LS*s each with 4 VCs. In case

Table 7.6 Leakage power reduction and throughput reduction with *numVCsPerPort*of 2D-Torus NoC

Port size	16 tiles NoC		64 tiles NoC		256 tiles NoC	
	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)
6	62.32	0.1	61.11	0.19	55.23	0.33
8	66.62	0.08	64.63	0.39	58.36	0.55
10	76.95	0.08	72.97	0.39	67.3	0.05
12	79.54	0.08	74.95	0.21	68.71	0.27
14	83.66	0.08	78.93	0.18	72.68	1.08
16	82.98	0.08	78	−0.04*	72.02	0.43

*Negligible increase

Table 7.7 Leakage power reduction and throughput reduction with *numVCsPerPort* for 2D-Mesh NoC

Port size	16 tiles NoC		64 tiles NoC		256 tiles NoC	
	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)
6	50.61	0.1	55.92	0.29	52.45	0.7
8	56.54	0.08	60.02	0.45	55.93	−0.05*
10	70.17	0.08	69.5	0.51	65.28	0.84
12	73.76	0.08	71.82	0.33	66.71	0.6
14	79.12	0.08	76.31	0.33	71.1	0.57
16	78.31	0.08	75.28	0.1	70.42	0.2

*Negligible increase

of light traffic (i.e., 1 or 2 VCs are needed), *TVA* turns OFF all *LS*s and allocates the *LS* that has 2 VCs of $numVCsPerPort = 14$ which means that $numVCsPerPort = 16$ has more idle VCs than $numVCsPerPort = 14$. *TVA* achieves the same results with heavy traffic when the number of VCs which are needed to serve incoming traffic is not multiple of 4, i.e., 5, 6, 9, 10, and 13.

2D-Mesh NoC is used with the same settings to get the results listed in Table 7.7. Port size of 14 VCs achieves the highest leakage power reduction of 79.1, 76.3 and 70.1 % for 16 tiles, 64 tiles and 256 tiles networks, respectively. Increasing *numVCsPerPort* enhances both throughput and leakage power saving since the available VCs to serve traffic is higher and the number of VCs to be deactivated, when traffic is light, increases.

7.6.5 Number of VC per Leaf Cell

Power and throughput reduction for 2D-Torus NoCs for *LP* startup and *maxIdle* $P = 0$ are listed in Table 7.8 for different cell sizes. As shown in the table, minor change in throughput is observed. *vcsPerCell* = *2* achieves the highest leakage power savings of 83, 78 and 72 % for 16 tiles, 64 tiles and 256 tiles NoC with $LS = 2$, respectively. $LS = 1$ improves power savings by at least 5 % with negligible impact on NoC throughput.

vcsPerCell has similar effect on 2D-Mesh NoC as reported in Table 7.9. The main difference is that 2D-Mesh NoC has less power saving than Torus NoC because the first has less number of input ports which means less number of VCs. The 16, 64 and 256 tiles NoCs achieve 78.3, 75.3 and 70.4 % of leakage power reduction, respectively, for $LS = 2$ with maximum throughput degradation of 0.1 %.

Increasing cell size improves the NoC throughput since the number of available VCs to serve traffic is increased. Lower leakage power saving is a negative impact of increasing *vcsPerCell* since the probability of idle VCs existence beside busy VCs is higher when the traffic is light which means that cell status is *PBusy* and cannot be powered OFF.

Table 7.8 Average leakage power reduction and throughput reduction for different *vcsPerCell* of 2D-Torus NoC

Cell size	16 tiles NoC		64 tiles NoC		256 tiles NoC	
	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)
2	82.98	0.08	78	−0.04*	72.02	0.43
4	74.77	0.08	69.89	0.08	64.53	0.09
8	54.56	0.08	53.8	0.08	47.21	0.09

*Negligible increase

Table 7.9 Leakage power reduction and throughput reduction for different *vcsPerCell* of 2D-Mesh NoC

Cell size	16 tiles NoC		64 tiles NoC		256 tiles NoC	
	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)	Average power reduction (%)	Throughput reduction (%)
2	78.31	0.08	75.28	0.1	70.42	0.2
4	67.36	0.08	65.93	0.21	62.42	0.12
8	40.42	0.08	47.63	0.21	43.85	0.04

7.6.6 Cell Maximum Idle Period

Increasing the *maxIdleP* means that *LVC*s would stay idle longer, which gives the opportunity for the incoming traffic to be served without waiting for VCs to be released. Hence, the throughput would increase. On the other hand when *LS* stays idle for longer period more leakage power is dissipated.

Power reduction × Throughput (*PRT*) is a new metric proposed to show the power reduction and performance of a NoC with *TVA*. *PRT* is listed in Table 7.10 for 2D-Torus NoC with *LP startupType*, $vcsPerCell = 2$ and different *maxIdleP* values. $maxIdleP = 0$ ns achieves the highest *PRT* of 0.78*GW.Bps* (83 % power savings), 2.41*GW.Bps* (78 % power savings) and 7.86*GW.Bps* (72 % power savings) for 16 tiles, 64 tiles and 256 tiles NoCs, respectively in $LS = 2$.

PRT for 2D-Mesh NoC is reported in Table 7.11 for different *maxIdleP*. Zero idle time achieves the highest *PRT* 0.55*GW.Bps* (78.3 % of leakage power reduction), 2.04*GW.Bps* (leakage power saving of 75.3 %) and 7.21*GW.Bps* (70.4 % of leakage power saved) for 16 tiles, 64 tiles and 256 tiles NoCs, respectively, for $LS = 2$. Using zero idle periods achieves the highest leakage power saving since idle *LS*s are turned-OFF immediately without leaking any power.

7.7 Conclusions

Low leakage power switch is proposed to reduce power dissipation of NoC circuit. Two techniques are applied for the proposed switch to reduce the leakage power dissipation of the switch. Circuit-level simulation shows that, *AVC* reduces the area of switch port which decreases the dynamic power by up to 54 %. The power dissipation

Table 7.10 Power reduction × throughput for different *maxIdlP* 2D-Torus topology

Idle Time (ns)	16 tiles NoC	64 tiles NoC	256 tiles NoC
	PRT (GW.Bps)	PRT (GW.Bps)	PRT (GW.Bps)
0	0.78	2.41	7.86
5	0.78	2.33	7.53
10	0.78	2.32	7.5

Table 7.11 Power reduction × throughput for different *maxIdlP* 2D-Mesh topology

Idle Time (ns)	16 tiles NoC	64 tiles NoC	256 tiles NoC
	PRT (GW.Bps)	PRT (GW.Bps)	PRT (GW.Bps)
0	0.55	2.04	7.21
5	0.55	1.95	6.84
10	0.55	1.94	6.83

of the proposed switch during the active mode is reduced by up to 81.4 % as compare to the conventional NoC switch. At inactive mode of the switch port, leakage power saving could increase to 97 %.

Using *AVC,* Traffic-based VC Activation (TVA) algorithm is proposed for low-power NoC. *TVA* minimizes power dissipation for a target throughput by activating/deactivating the VCs based on traffic conditions. *NIRGAM*_AVC simulator is developed to simulate NoC while employing *TVA*. In spite of VC switching overhead, *TVA* has negligible impact on overall NoC throughput. Increasing number of VCs per input port reduces the leakage power. *TVA* achieves high leakage power reduction for different topologies and different dimensions. *TVA* saves 84 % of 16 tiles 2D-Torus NoC leakage power with no impact on network throughput. *TVA* with low-power startup, zero idle time, and 14 VCs port saves 78.9 and 76.3 % of leakage power of 16 tiles NoC with 2D-Torus and 2D-Mesh topologies, respectively. NoC achieves power savings of 72.7 and 71.1 % when it run on 256 tiles medium size 2D-Torus and 2D-Mesh NoC, respectively, with 14 VCs input port size. *TVA* saves leakage power with negligible throughput reduction of 1.08 % in case of 2D-Torus topology and 0.84 % of 2D-Mesh network. *TVA* is evaluated for different topologies with different dimensions. Implementing VC cell set of size one enhances the leakage power savings by (5–8) % with negligible (less than) 5 % degradation in throughput. The power of *TVA* is utilized when the smallest cell and set sizes are used with smaller idle periods.

References

1. Rajsuman, R.: System-on-a-Chip Design and Test. Kluwer, Boston (2000)
2. Saleh, R., Wilton, S., Mirabbasi, S., Hu, A., Greenstreet, M., Lemieux, G., Pande, P., Grecu, C., Ivanov, A.: System-on-chip: reuse and integrate. Proc. IEEE **94**(6), 1050–1069 (2006)
3. Benini, L., Micheli, G.: Networks on chips: a new SoC paradigm. IEEE Comput. **35**(1), 70–78 (2002)
4. Bjerregaard, T., Mahadevan, S.: A survey of research and practices of network-on-chip. Proc. ACM Comput. Surv. **38**(1), 1 (2006)
5. Pande, P., Grecu, C., Ivanov, A., Saleh, R., De Micheli, G.: Design, synthesis and test of networks on chip. IEEE Des. Test Comput. **22**(5), 404–413 (2005)
6. Grecu, C.: Structured interconnect archiecture: a solution for the non-scalablility of bus-based SoCs. In: The Proceedings of the Great Lakes Symposium on VLSI, pp. 192–195, April 2004
7. Moraes, F., Calazans, N., Mello, A., Möller, L., Ost, L.: HERMES: an infrastructure for low area overhead packet-switching networks on chip. Integr. VLSI J **38**(1), 69–93 (2004)
8. Dally, W., Towles, B.: Route packets, not wires: on-chip interconnection networks: In: The IEEE Proceedings of the Design Automation Conference, pp. 684–689, June 2001
9. Ali, M., Welzl, M., Zwicknagl, M.: Networks on chips: scalable interconnects for future systems on chips. In: The Proceedings of the European Conference on Circuits and Systems for Communications, pp. 240–245, July 2008
10. Hemani, A., Jantsch, A., Kumar, S., Postula, A., Oberg, J., Millberg, M., Lindqvist, D.: Network on a chip: an architecture for billion transistor era. In: The Proceedings of IEEE NorChip Conference, pp. 166–173, November 2000

11. Huan, Y., DeHon, A/: FPGA optimized packet-switched NoC using split and merge primitives. In: The Proceedings of IEEE International Conference on Field-Programmable Technology, pp. 47–52, December 2012
12. Atienza, D., Angiolini, F., Murali, S., Pullini, A., Benini, L., Micheli, G.: Network-on-chip design and synthesis outlook. Integr. VLSI J **41**(3), 340–359 (2008)
13. Xu, Y., Jianyang, Z., Liu, S.: Research and analysis of routing algorithms for NoC. In: The Proceedings of Computer Research and Development, vol. 2, pp. 98–102, May 2011
14. Zhang, W., Hou, L., Wang, J., Geng, S., Wu, W.: Comparison research between XY and odd-even routing algorithm of a 2-dimension 3X3 mesh topology network-on-chip. In: The Proceedings of WRI Global Congress on Intelligent Systems, vol. 3, pp. 329–333 (2009)
15. Dally, W.: Virtual-channel flow control. IEEE Trans. Parallel Distrib. Syst. **3**(2), 194–205 (1992)
16. Pande, P., Grecu, C., Jones, M., Lvanov, A., Saleh, R.: Performance evaluation and design trade-offs for network-on-chip interconnect architectures. IEEE Trans. Comput. **54**(8), 1025–1040 (2005)
17. Sharma, P., Bairathi, R.: Study and analysis of the behavior of a generic mesh architecture of NoC routers. In: The Proceeding of the International Multi-Conference of Engineers and Computer Scientists, vol. II, pp. 17–19, March 2010
18. Nadi, M., Ghadiry, M., Dermany, M.: The effect of number of virtual channel on NOC EDP. J. Appl. Math. Inf. **2010**, 539–551 (2010)
19. Murali, S., Atienza, D., Meloni, P., Carta, S., Benini, L., Micheli, G., Raffo, L.: Synthesis of predictable networks-on-chip-based interconnect architectures for chip multiprocessors. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **15**(8), 869–880 (2007)
20. Burd, T., Brodersen, R.: Energy Efficient Microprocessor Design, pp. 376. Kluwer Academic Publishers, Boston (2002)
21. Kam, T., Rawat, S., Kirkpatrick, D., Roy, R., Spirakis, G., Sherwani, N., Peterson, C.: EDA challenges facing future microprocessor design. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **19**(12), 1498–1506 (2000)
22. Lee, K., Lee, S., Yoo, H.: Low-power network-on-chip for high performance SoC design. IEEE Trans. Very Large Scale Integr. Syst. (TVLSI) **14**(2), 148–160 (2006)
23. Milojevic, D., Montperrus, L., Verkest, D.: Power dissipation of the network-on-chip in a system-on-chip for MPEG-4 video encoding. In: IEEE Transactions on Solid-State Circuits, pp. 392–395, November 2007
24. Hu, Y., Zhu, Y., Chen, H., Graham, R., Cheng, C.: Communication latency aware low power NoC synthesis. In: The Proceedings of the IEEE/ACM Design Automation Conference, pp. 574–579 (2006)
25. Chouchene, W., Attia, B., Zitouni, A., Abid, N., Tourki, R.: A low power network interface for network on chip. In: The Proceeding of the International Multi-Conference on Systems, Signals and Devices, pp. 1–6, March 2011
26. Fang, Z., Hallnor, E., Li, B., Leddige, M., Dai, D.: Boomerang: reducing power consumption of response packets in NoCs with minimal performance impact. J. IEEE Comput. Archit. Lett. **9**(2), 49–52 (2010)
27. Ezz-Eldin, R., El-Moursy, M., Refaat, A.: Low power NoC switch using novel adaptive virtual channels. IJCSI Int. J. Comput. Sci. Issues **8**(5), 303–308 (2011)
28. Ezz-Eldin, R., El-Moursy, M., Refaat, A.: Novel adaptive virtual channels technique for NoC switch. In: The Proceedings of the IEEE International Design and Test Workshop, pp. 7–11, December 2011
29. Ezz-Eldin, R., El-Moursy, M., Refaat, A.: Low leakage power NoC switch using AVC. In: The Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 2549–2552, May 2012
30. Wang, H., Peh, L., Malik, S.: A power for routers: modeling alpha 21364 and Infiniband routers. In: The Proceeding of the Symposium on High Performance Interconnects, pp. 21–27, August 2002

31. Wei, L., Roy, K., Vivek, D.: Low voltage low power CMOS design techniques for deep submicron ICs. In: The Proceedings of the International Conference on VLSI Design, pp. 24–29, January 2000
32. Rabaey, J., Chandrakasan, A., Nikolic, B.: Digital Integrated Circuits. Prentice Hall, Upper Saddle River (2003)
33. Bellaouar, A., Elmasry, M.: Low-power digital VLSI design. Kluwer Academic Publishers, Boston (1995)
34. Benini, L.: Application specific NoC design. In: The Proceedings of the Design, Automation and Test in Europe, vol. 1, pp. 1–5 (2006)
35. Talwar, B., Amrutur, B.: Traffic engineered NoC for streaming applications. Microprocess. Microsyst. **37**(3), 333–344 (2013)
36. Muhammad, S.T., El-Moursy, M.A., El-Moursy, A.A., Refaat, A.M.: Traffic-based virtual channel activation for low-power NoC. In: The Proceedings of IEEE International Design and Test Workshop IDT, pp. 1–6, December 2013
37. Jain, L.: NIRGAM: a dynamic systemC simulator for NoC. http://nirgam.ecs.soton.ac.uk

Chapter 8
Decoupling Network Optimization by Swarm Intelligence

Jai Narayan Tripathi and Jayanta Mukherjee

Abstract In this chapter, the problem of decoupling network optimization is discussed in detail. Swarm intelligence is used for maintaining power integrity in high-speed systems. The optimum number of capacitors and their values are selected to meet the target impedance of the system.

Keywords Signal integrity · Power integrity · High-speed systems · Power delivery networks · Optimization · Decoupling capacitors · Stochastic optimization · Particle swarm optimization

8.1 Introduction

Power Integrity (PI) is associated with quality of power delivered in a high-speed system. In a Power Delivery Network (PDN) sufficient and stable power should be delivered to each component with proper efficiency. A PDN has off-chip and on-chip components. The components in a PDN have different impedance profiles which are dominant in different frequency ranges. In order to maintain power integrity, the cumulative impedance profile of the PDN should be below target impedance which is defined based on the maximum allowable ripple in the system. To maintain the impedance profile of the PDN, various designing and optimization techniques are used. This chapter focuses on damping cavity mode effects in power delivery networks by metaheuristic optimization techniques. The optimization techniques

Permission has been taken from IEEE for reproduction of authors' own work published earlier by IEEE.

J.N. Tripathi (✉)
STMicroelectronics Pvt. Ltd., 7/68, Shastri Nagar, Gangapur (Bhilwara) 311801, Rajasthan, India
e-mail: jainarayan.tripathi@st.com

J. Mukherjee
IIT Bombay, Mumbai, India
e-mail: jayanta@ee.iitb.ac.in

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_8

are used for selecting and placing decoupling capacitors on power planes. The S-parameters data of power plane geometry and capacitors are used for the accurate analysis including bulk capacitors and VRM, for a real-world problem. There are two case studies presented, one with the continuous optimization for a theoretical case and the other one with discrete optimization problem which is real-world problem from industry. A generic methodology is developed which can be used for design of decoupling network in any real-world power delivery network, based on its s-parameters files.

The main components of the power delivery network are die, package, the PCB planes, decoupling capacitors, bulk capacitors, and the voltage regulator module. The decoupling capacitors help the voltage regulator to supply current when there is high demand of current into the chip by supplying the charge stored in them. In a typical PDN, the electrolytic bulk capacitors are effective only at very low frequencies (< 10 MHz) and the high-frequency ceramic decoupling capacitors are effective up to a hundreds of MHz and the package decoupling capacitors can be effective up to a few hundred MHz [2]. After this frequency range, the on-die capacitance affects the PDN in the range of GHz. The frequency range of interest for decoupling network is decided by the current transient requirements of the chip, as described in the following subsection. In the earlier published literature, the range analysis optimization and analysis of decoupling network are done with bare board only, without taking the effect of bulk capacitors, package, and VRM. Though they do not significantly affect the PDN in the range where the decoupling network is dominant, all these components affect the amplitude at the upper and lower limits of the frequency range of analysis taken. The previous research work published in [3, 4] did not take into account such analysis with VRM, bulk caps, and package and that is why, is inaccurate.

8.1.1 Power Planes

Power planes are the simplest structures used as power bus. Figure 8.1 shows the simplest and most popular power plane structure which is a rectangular power plane. VRM is mounted on the power plane to supply power. With VRM, bulk capacitor is used to store the charge supplied by the VRM. The package is also mounted on the power plane.

8.1.2 Power Plane Cavity Model

At lower frequencies, rectangular power planes can be analyzed as lumped components but at higher frequencies distributed modeling is necessary. These can be considered as rectangular cavities for easier computations, compared to the full-wave numerical modeling techniques [5–8]. Various methods can be used for modeling

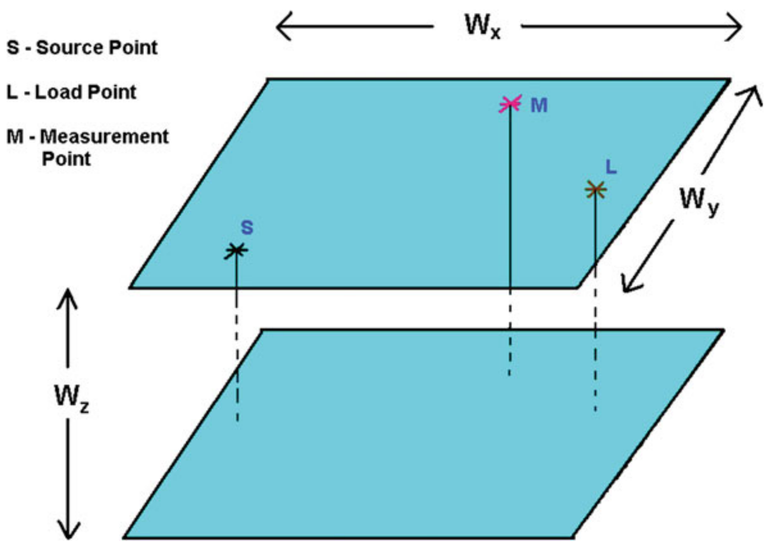


Fig. 8.1 Rectangular power plane

of power planes such as cavity model, Finite Element Method (FEM), Transmission Matrix Method (TMM), Finite Difference Time Domain (FDTD) method, etc.

If the cavity model is taken into account for computation, the impedance of the power plane can be expressed as a double infinite sum of the terms corresponding to the cavity modes. This is shown in Eq. 8.1. The impedance $Z_{Ld}(f, X_f, Y_f)$ of a power plane (of the size $(W_x \times W_y \times W_z)$ observed at (X_f, Y_f) which is point M in Fig. 8.1, fed by current I_s at source point (X_s, Y_s), loaded by N_u loads at positions (X_L, Y_L), can be given by Eq. 8.1 [4].

$$Z_{Ld}(f, X_f, Y_f) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{\gamma_{mn} \cdot c_{mn}(X_s, Y_s) \cdot c_{mn}(X_f, Y_f) \cdot \frac{W_z}{W_x W_y}}{\frac{\varepsilon\omega}{Q} + j\left(\varepsilon\omega - \frac{k_{xm}^2 + k_{yn}^2}{\omega\mu}\right) + \frac{\gamma_{mn} . W_z}{W_x W_y}\sum_{u=1}^{N_u} c_{mn}(X_{Lu}, Y_{Lu}) \cdot Y'_{Lu}} \tag{8.1}$$

where

$$c_{mn}(X_i, Y_i) = cos(K_{xm}X_i) \cdot cos(K_{yn}Y_i) \cdot sinc\frac{k_{xm}p_{xi}}{2} \cdot sinc\frac{k_{yn}p_{yi}}{2} \tag{8.2}$$

$$k_{xm} = \frac{m\pi}{W_x}, k_{yn} = \frac{n\pi}{W_y}, \omega = 2\pi f \tag{8.3}$$

$$Q = \left[tan\delta + \sqrt{2/\omega\mu k W_z^2}\right]^{-1} \tag{8.4}$$

$$Y'_L = \sum_{u=1}^{N_u} c_{mn}^2 (X_{Lu}, Y_{Lu}) \cdot Y'_{Lu} \tag{8.5}$$

$$Y'_{Lu} = \left[R_{Lu} + j(\omega L_{Lu} - \frac{1}{\omega C_{Lu}}) \right]^{-1} \tag{8.6}$$

γ_{mn} is 1 if both m and n are 0, is 4 if $m \neq 0$ as well as $n \neq 0$, and is 2 if any of them is 0. $tan\delta$, ε μ, f, and P_i are loss tangent, permittivity, permeability, frequency, and port's width, respectively.

In Eq. 8.1, j is imaginary number $\sqrt{-1}$. Y'_{Lu} corresponds to the admittance of uth load, and Y'_L is sum of Y_{Lu} for all the loads. In a bare board, since there is no load the term Y'_L vanishes. Figure 8.2 shows the impedance profile up to 1 GHz, for the board without any load, having the dimension 200 mm × 150 mm × 0.17 mm, source placed at point (0, 75 mm), and impedance observed from the point (0, 75 mm). The port width and length, both are taken 0.254 mm. The values of $tan\delta$, ε_r, and μ_r are taken 0.02, 4.1, and 1. In Eq. 8.4, k is the conductivity of the metal which is 59.6×10^6 s/m. It can be seen that there are various capacitive and inductive effects due to cavity modes. The peaks with maximum impedance are called *anti-resonance* points while the valley points with minimum impedance are called *resonance* points.

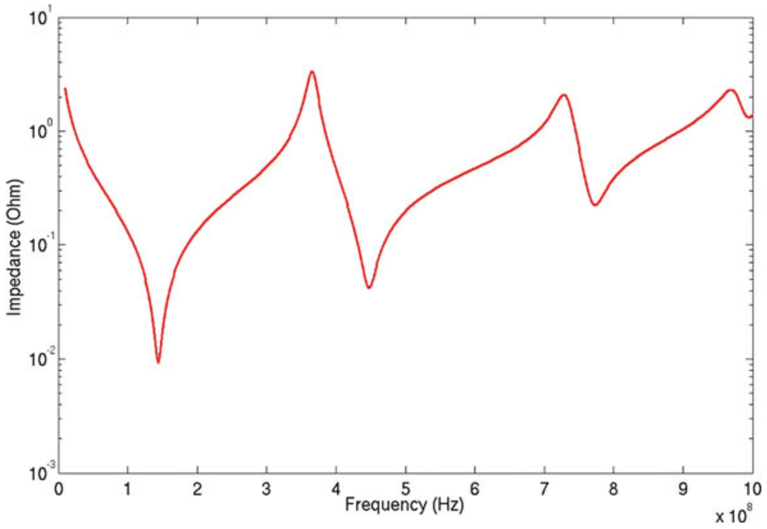


Fig. 8.2 Impedance profile of rectangular power plane by cavity model

8.1.3 Decoupling Capacitors

The capacitors which are used for the decoupling are called decoupling capacitors. Decoupling capacitors are placed on the power plane which, reduces the effective impedance of the PDN in the frequency range of MHz.

8.1.4 Capacitor Behavior at High Frequencies

A practical capacitor is not purely capacitive at wide frequency ranges. At higher frequencies, the inductive behavior comes into picture and the capacitive behavior is overshadowed by the inductive behavior. An equivalent model of a capacitor is given by resistive, capacitive, and inductive elements in series, which are called Equivalent Series Resistor (ESR) and Equivalent Series Inductor (ESL). Figure 8.3 shows the pure capacitive behavior of a 1 μF capacitor with ESR and ESL of 0.1 Ω and 10 nH, respectively. The point where the impedance is lowest is called the *resonance* point. Beyond the resonance point, capacitor behaves as an inductor.

8.1.5 Decoupling Strategies

The role of decoupling capacitors is to damp the anti-resonances' peaks on power plane impedance profile as shown in Fig. 8.2. For finding the suitable capacitors, the frequency is observed where the anti-resonance is found. The capacitors is chosen

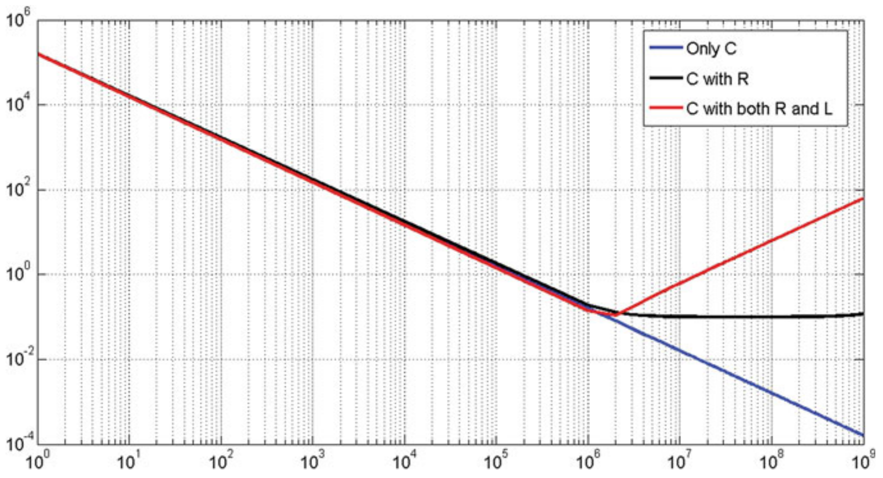


Fig. 8.3 Effects of ESR and ESL in the impedance profile of a capacitor

in such a way that its resonance point should be almost near to that frequency of board anti-resonance so that it can damp that particular anti-resonance point. But that particular capacitor will again form an anti-resonance at some higher frequency due to cumulative inductance of its own inductance and the same of the board.

8.2 Case Study 1: Optimization of Decoupling Capacitors

This case study presents an effective methodology for suppressing the cavity-mode anti-resonances peaks on a rectangular power plane. The optimum values and the optimal positions of the decoupling capacitors are found using Particle Swarm Optimization (PSO), which leads to optimum impedance of power plane loaded with decoupling capacitors. Optimum number of capacitors and their values, by which impedance of loaded board is matched below the target impedance of the system, are found.

8.2.1 Optimization

For applying PSO, the parameters or variables taken are the r, l, c values of the capacitors and their locations on power plane represented by coordinates x, y. Thus, the positions and the velocities of the particles are five-dimensional each. The problem is defined as continuous PSO problem where the ranges of values of capacitor parameter are taken from the options available in the market [9]. The ranges are given in Table 8.1.

There are two cases discussed using PSO, one with one decoupling capacitor and another one with two decoupling capacitors.

8.2.2 One Decap Optimization

From the impedance profile of unloaded power plane observed from the point $(X_s, 0)$ (as shown in Fig. 8.1), the first anti-resonance is observed at 365 MHz. At this

Table 8.1 Ranges for parameters of decoupling capacitors

Parameter	Lower bound	Upper bound
$r(\Omega)$	0.1	5
l(nH)	0.01	12
c(nF)	0.001	18
x(mm)	0	200
y(mm)	0	150

frequency the impedance is maximum in the given frequency range of interest (up to 1 GHz). According to the parameters given in Table 8.1, 75 random particles are defined for 50 iterations. The values of the PSO parameters ω_i, ω_f, p_1, and p_2 are taken as 0.9, 0.4, 1.49, and 1.49, respectively [10]. The particles converged and the solution is [0.1 Ω, 0.037 nH, 10 nF, 200 mm, 150 mm], resulting the maximum impedance (100 MHz to 1 GHz) of 0.9480 Ω. The impedance profile of the globally best particle (final solution) is shown in Fig. 8.4. It can be seen that it has damped the anti-resonance peaks.

8.2.3 Two Decap Optimization

The same optimization problem mentioned above, when solved using two capacitors, required ten-dimensional particles (5 parameters for each capacitor). For 75 particles and 50 iterations, the solution is [0.1 Ω, 0.01 nH, 11.194 nF, 200 mm, 150 mm] for one capacitor and [0.1 Ω, 9.956 nH, 3.587 nF, 188.63 mm, 66.12 mm], for another one, resulting the maximum impedance (up to 1 GHz) of 0.7352 Ω as shown in Fig. 8.5. Figure 8.5 shows the damping of peaks with both the cases, i.e., using one and two decoupling capacitors.

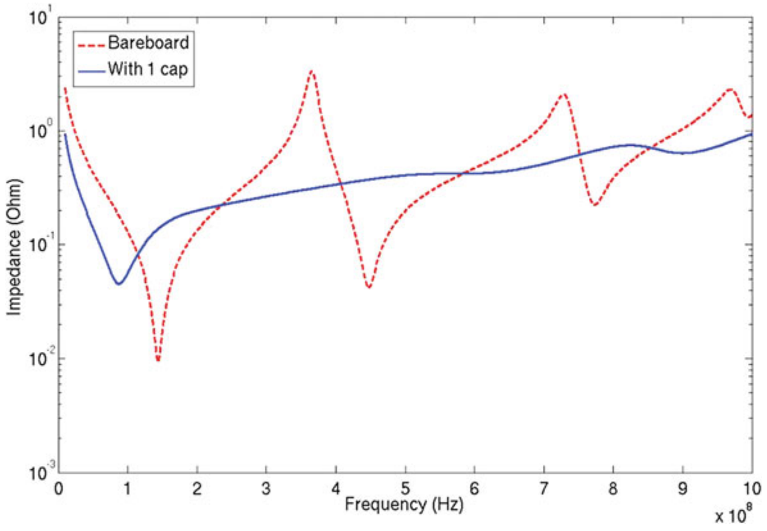


Fig. 8.4 Impedance profile of board with one decoupling capacitor of optimal value and loaded at optimal position

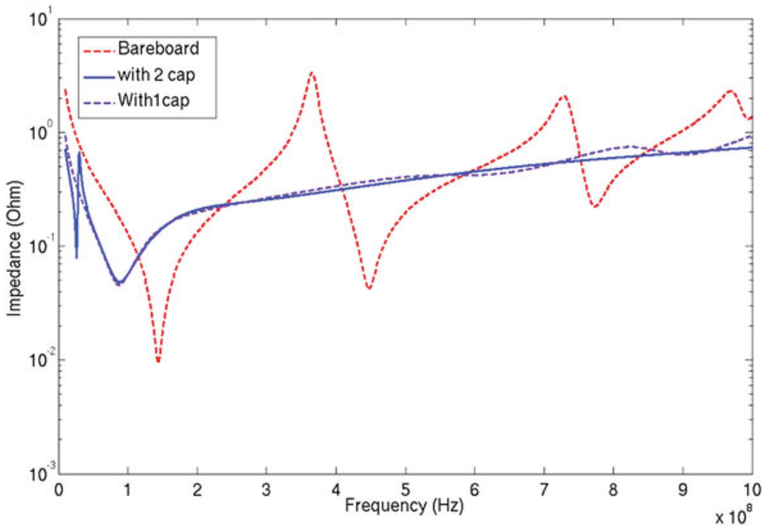


Fig. 8.5 Impedance profile of board with two decoupling capacitors of optimal values and loaded at optimal positions

8.2.4 Solving Power Integrity Problem

As shown in the previous section, the impedance of the system is loaded by placing capacitors. But even if with two capacitors, it does not match to the target impedance. In our system the voltage is 3.3 V with the tolerance of device equal to 8 %. The device draws 0.5 A current, thus target impedance needed is 0.528 Ω. So a computer program is written to find the optimal number of capacitors as well, in addition to their values and positions. The minimum number of decoupling capacitors and their corresponding values, needed to achieve this target impedance are found by PSO. The pseudocode for the search process is as follows:

Pseudocode for PSO :

```
1. Define maximum number of iterations $T_{max}$
2. For $D$ dimensions , generate $P$ particles, their respective positions $\mathbf{x}(i, j)$, and
   velocities $\mathbf{v}(i, j)$; $\forall i \in \{1, 2, \ldots, P\}$ and $j \in \{1, 2, \ldots, D\}$, within the lower
   and upper bounds.
3. Calculate the maximum impedance peak $max\_imp(i)$ corresponding to all the
   $i$ particles.
4. Define all the local particles as local best particles **lbest** and find the global best
   particle **gbest**.
5. Loop 1: while Number of iterations $t \leq T_{max}$.
6. Loop 2: for i = 1:P
7. Update inertia, velocities, and positions
   $\omega = (\omega_i - \omega_f)\frac{(T_{max})-t}{T_{max}} + \omega_f$
```

Table 8.2 Parameters and positions for optimum decoupling capacitors

Parameter	Cap 1	Cap 2	Cap 3	Cap 4	Cap 5
$r(\Omega)$	0.10	0.10	0.10	4.83	0.10
l(nH)	11.16	0.01	0.01	10.14	0.01
c(nF)	0.001	3.12	17.87	0.001	3.12
x(mm)	0	0	0	0	200
y(mm)	150	150	0	150	150

```
    $\mathbf{v}(i, j) = \omega(t)\mathbf{v}(i, j) + p_1 r_1(\mathbf{lbest}(i, j) - \mathbf{x}(i, j)) + p_2 r_2(\mathbf{gbest} - \mathbf{x}(i, j))$
    $\mathbf{x}(i, j) = \mathbf{x}(i, j) + \mathbf{v}(i, j)$
 8. Limit the positions and velocities within the lower and upper bounds.
 9. End : Loop 2
10. Update $lbest(i, j)$ and $gbest(j)$ accordingly.
11. Increment the counter: $t = t + 1$
12. END: Loop 1
13. Final Solution = gbest.
```

8.2.5 Results

The number of decoupling capacitors N needed to meet the target impedance is 5.

Their values and positions are also found by PSO. Table 8.2 shows the r, l, c values and positions of the capacitors needed to achieve the target impedance.

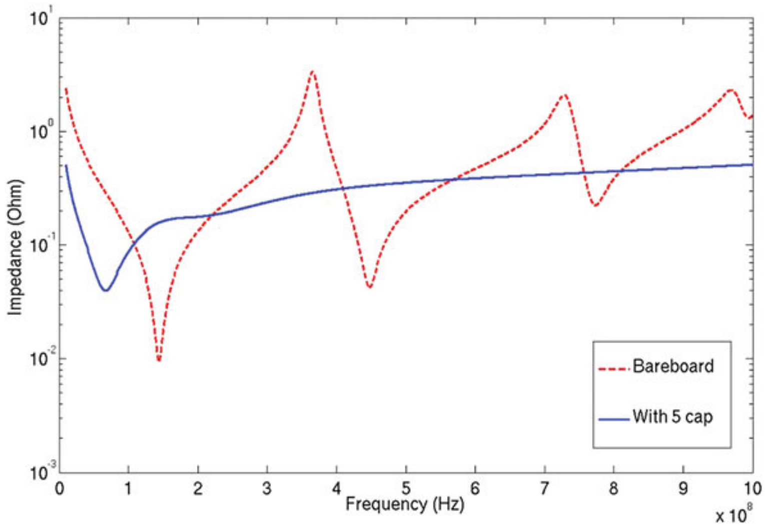


Fig. 8.6 Meeting the target impedance by optimal number of decoupling capacitors

The impedance profile of the plane after loading these capacitors at their best positions is shown in Fig. 8.6. It can be seen that the maximum impedance in the frequency range of interest, is less than the target impedance. The maximum impedance after loading by these 5 caps, is $0.5088\,\Omega$.

PSO is proved to be useful for designing Power Delivery Network, in order to maintain Power Integrity. The ease of implementation of PSO is the motivation of using it. The optimal number of decoupling capacitors, their values, and their positions are found by PSO. A pseudocode is given to solve the power plane target impedance problem, which can be used for other geometries also.

8.3 Swarm Intelligence Algorithms

For power plane decoupling capacitor placement, Genetic Algorithm (GA) and Simulated Annealing (SA) are used in the literature [11–13]. In this work, Particle Swarm Optimization (PSO), cuckoo search method, and firefly algorithm are used for finding the optimum values and positions for decoupling capacitors on a power plane of a PDN. All of these algorithms belong to the family of swarm intelligence algorithms. Swarm Intelligence (SI) is the property of a system where the collective behaviors of entities interacting locally with their environment and other entities, resulting into a global pattern formation. The inspiration often comes from nature, especially biological systems [14, 15]. The performance of these algorithms are compared, in context of the problem presented in this section, for finding decoupling capacitor locations on a power plane. For details of these algorithms, readers are requested to refer the literature [14, 16–19].

8.3.1 Optimization

For applying SI algorithms, the parameters or variables taken are the r, l, c values (termed as ESR and ESL popularly) of the capacitors, and their positions x, y. There are five capacitors chosen, so the size of the entities (particles, nests, or fireflies) are 25-dimensional each. The problem is defined as continuous optimization problem where the values lower and upper bounds of capacitor parameters are taken from the options available in the market [20]. The ranges are taken as close intervals $[0.1\,\Omega, 5\,\Omega]$, [0.01 nH, 12 nH], and [0.001 nF, 18 nF] for r, l, and c, respectively. For x and y, the lower and upper bounds are defined by the board dimensions. From the impedance profile of unloaded power plane observed from the point $(X_s, 0)$, the first anti-resonance peak is at 365 MHz as shown in Fig. 8.2. At this frequency the impedance is maximum in the given frequency range of interest (up to 1 GHz). Thus, the maximum value of the impedance over the frequency ranges, is defined as the objective function (to be minimized). The decoupling capacitors and their corresponding values, are found by PSO, CS, and FA. The pseudocodes for the

Table 8.3 Parameters taken for various algorithms

Algorithm	Parameters			
PSO	$w_i = 0.9$	$w_f = 0.4$	$p_1 = 1.49$	$p_2 = 1.49$
CS	$p_a = 0.25$	$\beta = 3/2$	$\sigma = 0.6966$	–
FA	$\alpha = 0.25$	$\beta_{min} = 0.3$	$\gamma = 1$	–

search process are given below. The number of decoupling capacitors taken for the experiments is 5. The values of all the parameters used in various algorithms are given in Table 8.3.

8.3.2 Pseudocode for PSO

```
1. Define maximum number of iterations $T_{max}$
2. For $D$ dimensions , generate $P$ particles, their respective positions $\mathbf{x}(i, j)$, and velocities $\mathbf{v}(i, j)$; $\forall i \in \{1, 2, \ldots, P\}$ and $j \in \{1, 2, \ldots, D\}$, within the lower and upper bounds.
3. Calculate the maximum impedance peak $max\_imp(i)$ corresponding to all the $i$ particles.
4. Define all the local particles as local best particles **lbest** and find the global best particle **gbest**.
5. Loop 1: while Number of iterations $t \leq T_{max}$.
6. Loop 2: for i = 1:P
7. Update inertia, velocities, and positions
   $\omega = (\omega_i - \omega_f)\frac{(T_{max})-t}{T_{max}} + \omega_f$
   $\mathbf{v}(i, j) = \omega(t)\mathbf{v}(i, j) + p_1 r_1(\mathbf{lbest}(i, j) - \mathbf{x}(i, j) + p_2 r_2(\mathbf{gbest} - \mathbf{x}(i, j)$
   $\mathbf{x}(i, j) = \mathbf{x}(i, j) + \mathbf{v}(i, j)$
8. Limit the positions and velocities within the lower and upper bounds.
9. End: Loop 2
10. Update $lbest(i, j)$ and $gbest(j)$ accordingly.
11. Increment the counter: $t = t + 1$
12. END: Loop 1
13. Final Solution = **gbest**.
```

8.3.3 Pseudocode for CS

```
1. Define maximum number of iterations $T_{max}$
2. For $D$ dimensions , generate $N$ number of nests $\mathbf{nest(i, j)}$; $\forall i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, D\}$, within the lower and upper bounds.
```

```
3. Calculate the fitness associated with all nests (as maximum impedance peak $max\_imp$).
4. Find the nest with the greatest fitness (minimum impedance) **best_nest**.
5. Loop 1 : while Number of iterations $t \leq T_{max}$
6. Loop 2 : $i = 1 : N$
   Update nests by Levy flights and evaluate their fitness $F_i$. Choose random nests
   and update them by random vector. Evaluate fitness $F_j$ of new nest.
7. if $(F_i > F_j)$, Replace old by the new solution,
   Keep the best solutions (or nests with quality solutions).
8. Limit the nests within the lower bounds and upper bounds.
9. End : Loop 2
10. Update $best\_nest$ accordingly.
11. Increment the counter : $t = t + 1$
12. END : Loop 1
13. Final Solution = **best_nest**.
```

8.3.4 Pseudocode for FA

```
1. Define maximum number of iterations $T_{max}$
2. For $D$ dimensions ,generate $P$ population of fireflies , their respective positions
   $\mathbf{pos}(i, j); \forall i \in \{1, 2, \ldots, P\}$ and $j \in \{1, 2, \ldots, D\}$, within the lower and upper
   bounds.
3. Calculate the light intensity associated with all fireflies as maximum impedance
   peak $max\_imp$.
4. Find the firefly with the highest light intensity (minimum impedance) **brightest**.
5. Loop 1 : while Number of iterations $t \leq T_{max}$
6. Loop 2 : $i = 1 : P$
7. Loop 3 : $k = 1 : P$
8. if $max\_imp(i) > max\_imp(k)$
   $r \leftarrow$ distance between the flies
   $\beta = (\beta_0 - \beta_{min})e^{-\gamma r^2} + \beta_{min}$
   $temp \leftarrow$ random vector
   $\mathbf{pos}(i, :) = \mathbf{pos}(i, :)(1 - \beta) + \mathbf{pos}(j, :)\beta + temp$
9. Limit the positions of flies within the lower bounds and upper bounds.
10. End : Loop 3
11. End : Loop 2
12. Update $brightest$ accordingly.
13. Increment the counter : $t = t + 1$
14. END : Loop 1
15. Final Solution = **brightest**.
```

Table 8.4 Performance comparison of various algorithms

Run no	PSO		CS		FA	
	Max. Imp. (Ω)	Time (s)	Max. Imp. (Ω)	Time (s)	Max. Imp. (Ω)	Time (s)
1	0.8004	19.15	0.7037	37.26	0.7789	54.68
2	0.8080	19.75	0.7094	37.42	0.7777	58.87
3	0.7867	19.15	0.7182	37.36	0.7789	54.99
4	0.8084	19.18	0.7081	42.04	0.7812	59.21
5	0.8046	19.16	0.7044	40.58	0.7844	55.23
6	0.7981	19.14	0.7271	37.35	0.7844	56.07
7	0.8094	19.21	0.7235	42.17	0.7714	59.76
8	0.7774	19.23	0.7046	42.28	0.7778	58.42
9	0.8134	19.10	0.7038	42.27	0.7775	54.87
10	0.7954	19.14	0.7191	37.52	0.7944	57.89

8.3.5 *Results*

In order to compare the algorithms the number of entities (particles/nests/flies) are limited to 5 and number of iterations to 25. The simulations are run on a machine with 32 GB RAM, 2.95 GHz CPU speed and RHEL 5.4 Operating System. Table 8.4 shows results obtained from various swam intelligence algorithms and their comparisons. In one iteration PSO calls objective function (cost function) once, CS calls it twice while FA can call it N (number of flies) to 0 times depending on the value it reaches. That is why there is the difference among the time taken by these algorithms, which can be seen clearly in Table 8.4.

Figure 8.7 shows the variation of the impedance of best entities of all three algorithms with each iteration. With current selection of variables in each algorithm, it is clear that PSO has maximum step size and so the graph is not smooth but it takes least time, on other hand fireflies have small steps and curve is smoother but it takes maximum time. Cuckoo search stands in middle of other two in respect of time taken for finding the solution. For current set of simulations, the minimum impedance is achieved by cuckoo search method. Figure 8.8 shows the impedance profile after the placement of the decoupling capacitors by these algorithms.

8.4 Case Study 2: Real-World Problem

In this case study, the s-parameters data of power plane geometry and capacitors are used for the accurate analysis including bulk capacitors and VRM. Unlike the earlier work in the literature, this study presents the practical solution of the industrial problem of discrete optimization for finding the suitable decoupling capacitors (provided

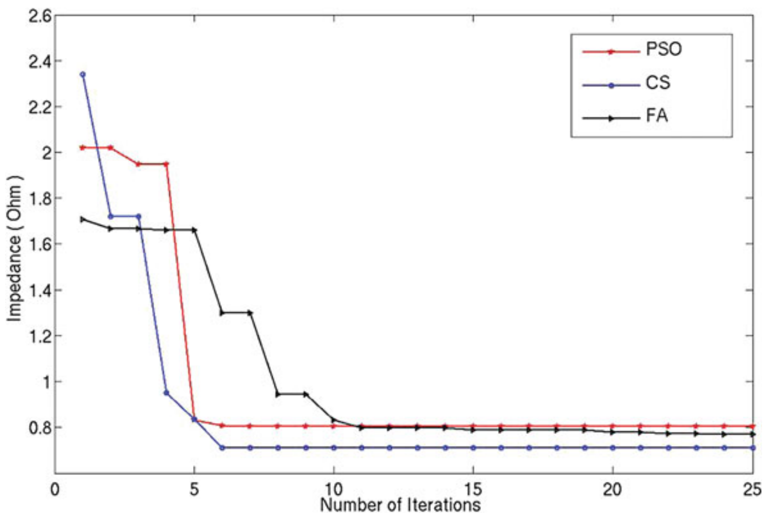


Fig. 8.7 Convergence of PSO, CS, and FA algorithms

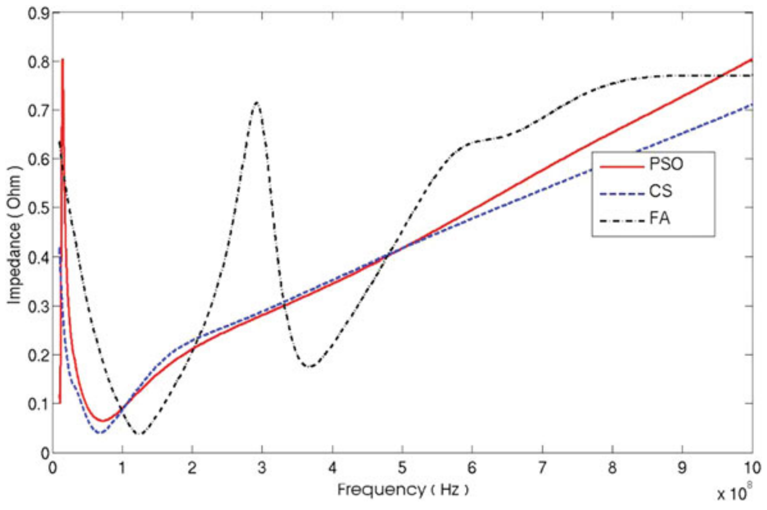


Fig. 8.8 Comparison of solutions by PSO, CS, and FA

with their s-parameter files) and their best positions from the available positions on the board. The novelties in this work, are following :

1. The methodology for decoupling optimization is based on the s-parameters models of the decoupling capacitors as practically available from the various manufacturers instead of *rlc* models adopted in tandem.

2. Using s-parameters data for board and package to take into account the effects of via, pad and anti-pads.
3. Solving the optimization problem for PDN, after taking into effect of bulk capacitors, VRM, and package in order to increase the efficiency and accuracy.
4. Finding the trajectory of the particles in PSO when the s-parameters data are used, unlike the r, l, c ranges taken in the earlier papers.
5. Interpolating the data of capacitor bank before using it for optimization, if the frequency range of the board is not the same as that of the s-parameters file from the capacitor bank.
6. Explaining the rationale behind choosing the frequency range of decoupling network analysis.

Figure 8.9 shows the impedance profile of the bare board and the board with VRM, bulk capacitors, and package. There is a huge difference in the two profiles in terms of resonance and anti-resonance peaks. If the optimized capacitors are selected taking into account bare board only, the analysis is not accurate when the board is loaded.

8.4.1 Range of Analysis

The amplitude variation of the required current for the power supply decides the target impedance while the Power Spectral Density (PSD) of the same decides the frequency range of analysis for decoupling network. Figure 8.10 shows the transient behavior of current required at the package pin, and Fig. 8.11 shows its corresponding Power

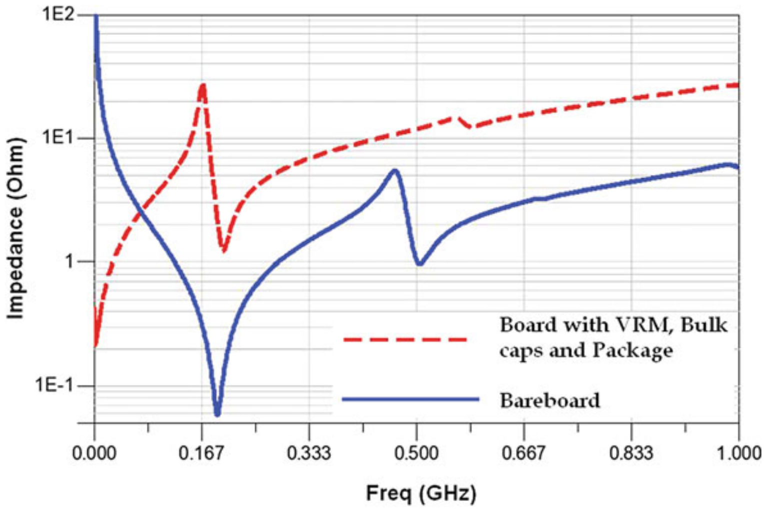


Fig. 8.9 Impedance profile of bare board and board with VRM, bulk capacitor, and package

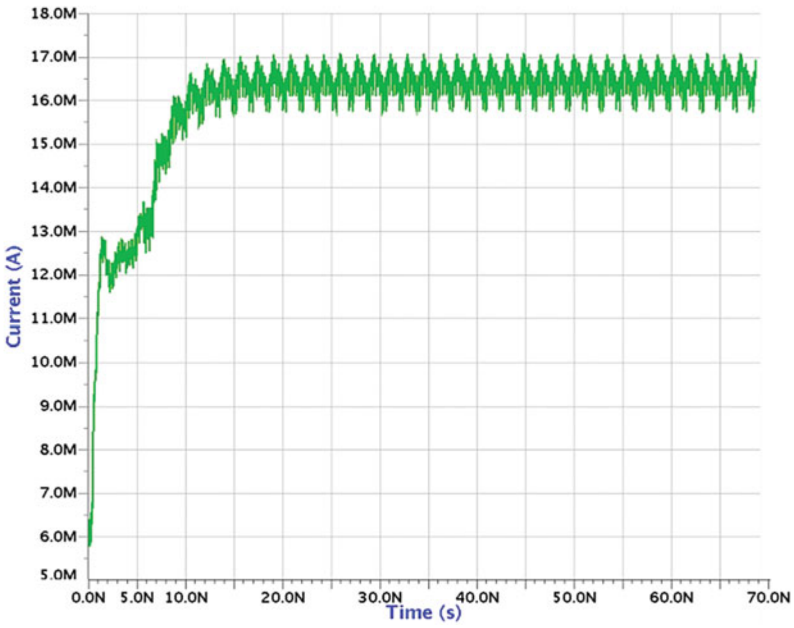


Fig. 8.10 Transient current required by the device at the pin of package

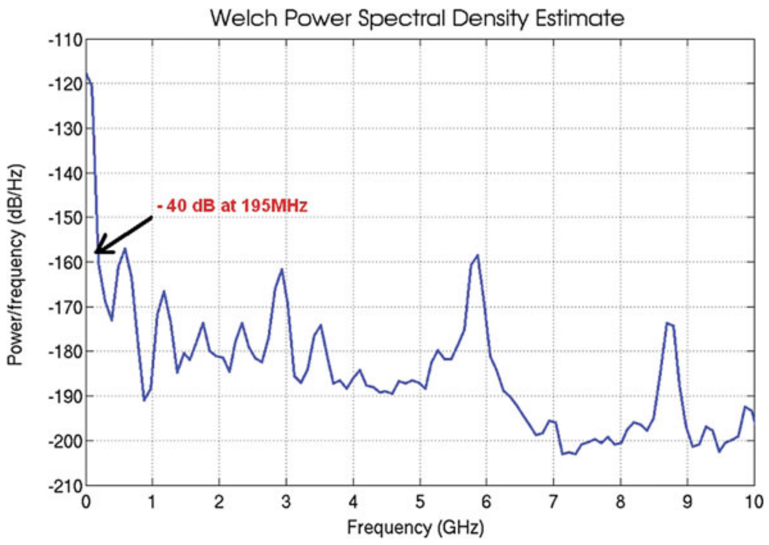


Fig. 8.11 Power spectral density of current at package pin

Spectral Density (PSD). The PSD clearly shows that the 99 % (–40 dB) of signal strength is within 195 MHz, so the upper range for decoupling network optimization is taken approximately as 200 MHz.

8.4.2 S-Parameters

The s-parameters are the 2-port parameters used to represent the characteristics of a passive or active network, as a black box. The s-parameters are used here for the accurate analysis.

8.4.2.1 Power Nets

As described in earlier section, power plane pairs can be modeled by cavity models. For the complicated geometries, the equivalent cavity models-based empirical formulae are not effective and accurate. S-parameters file extracted from the measurements or the EDA tools, becomes useful for such case. Figure 8.12 shows three-dimensional view of such a power net in a package. The similar or more complicated structures are in PDN layers of boards. Thus, for accurate PI analysis, the s-parameters data must be used.

8.4.2.2 Capacitor Bank

The approximate *rlc* models of the decoupling capacitor do not take into account the nonlinear behavior of the capacitors at higher frequencies while s-parameters models take into account higher order effects even at higher frequencies. The *rlc* models of the capacitors are defined at one spot frequency or a narrow band, instead of broader range. The capacitor models used in the earlier work are *rlc* models [4, 21]. The *rlc* models are the approximations of the actual characteristics of the capacitors. Figure 8.13 shows the comparison of the impedance profile of a capacitor obtained from s-paramteres and *rlc* values. This is the profile of capacitor LLR185C70G105ME01 from Murata Manufacturing Co. The r, l, c values (ESR of 100 mΩ and ESL of 120 pH) are taken from the data sheet and the s-parameters file is obtained from vendor's website [22]. There is clearly a difference between the two profiles. The maximum difference is in hundreds of ohms at lower frequency range, even at high frequencies, second-order effects are visible only in s-parameter models and not in linear *rlc* models as generally adopted in analysis. To avoid the inaccuracies involved in *rlc* models, s-parameters data is used in this analysis.

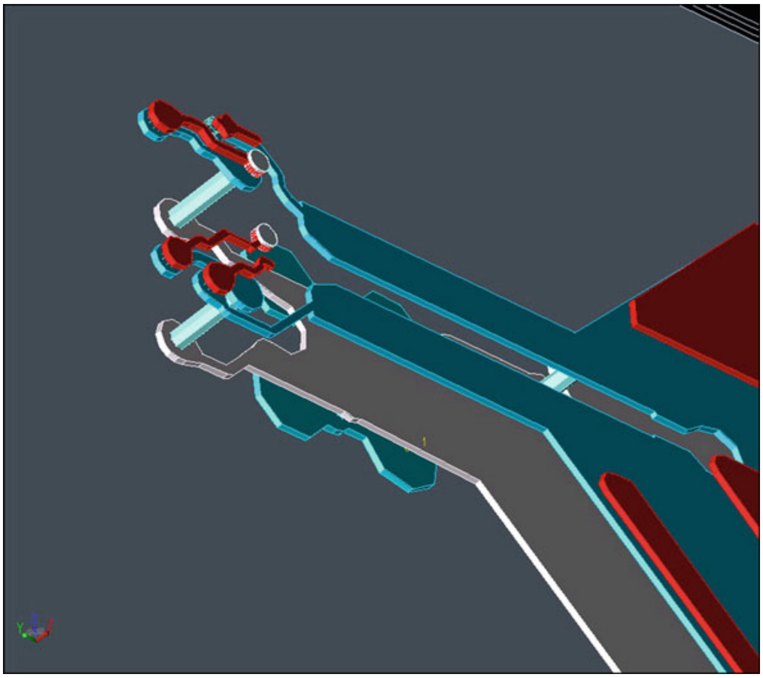


Fig. 8.12 Power delivery plane and nets in a typical package

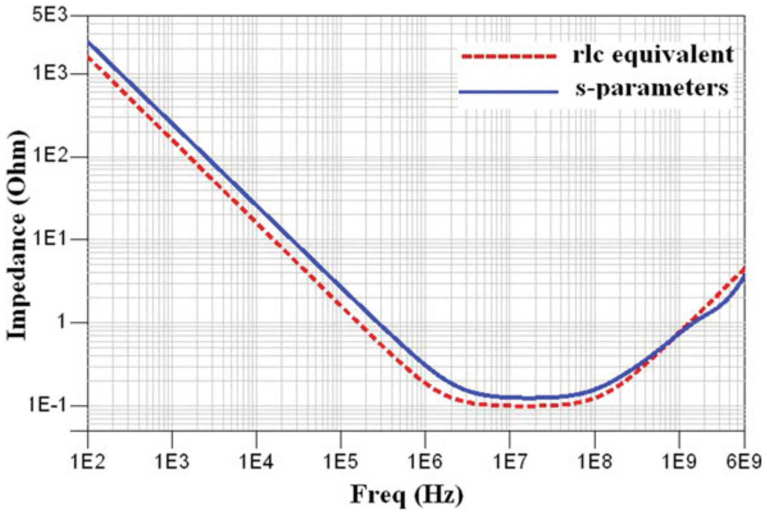


Fig. 8.13 Impedance profiles of a cap obtained from s-parameters and *rlc* model

8.4.3 Optimization Methodology

The proposed methodology aims to find the optimal number of capacitors, their values, and their positions on the board, in order to meet the target impedance of the system. The methodology is based on impedance profile generation from s-parameters data for the complex geometry of the multilayer board and the given bank of capacitors. The steps followed in this methodology are :

1. Extraction of s-parameters of multilayer board and package from the design database.
2. Including the effects of VRM, bulk capacitor, board, and package to find composite system impedance profile.
3. Generation of decoupling capacitor bank (z-profiles) from their given s-parameters profiles provided by different vendors.
4. Indexing and ordering the capacitor bank, according to the resonance points in their impedance profiles.
5. Indexing the coordinates of the available ports on the board based on their relative positioning.
6. Applying PSO in iterative loop to achieve the target impedance.

The cumulative z-parameter matrix of a board loaded with decaps can be given by the following formula [2].

$$Z_{eff} = (Z^{-1} + Z_{decap}^{-1})^{-1} \tag{8.7}$$

where the z-parameters matrix of the board is Z and Z_{decap} is the diagonal matrix in which the diagonal elements are concerned with the impedance of the decoupling capacitors on the ports corresponding to the diagonal elements and all other elements are zero. The problem with the above formula appears when the decoupling capacitors on the board are less than the available number of ports. In that case, one or more number of diagonal elements of matrix Z_{decap} are zero, which will not allow the inverse of it. To avoid this problem y-parameters were used for that step particularly. The analysis is carried out for a high-speed serial link board. The extracted s-parameters file is having 39 ports. There are 39 ports in the board from which 4 ports are reserved, one for VRM and 3 for capacitors, while one of the ports is the observation port where the package pin is connected. Thus, these five ports are not taken into account while defining or initializing the ports for decoupling capacitors. There are thousands of capacitors (3702 for this study) used for creating the bank. For defining the particles, the s-parameters data and the port numbers are used. Each particle has two dimensions corresponding to one capacitor, one for the capacitor number from the capacitor bank and one for the port number on the board. If the particle is using multiple capacitors, the dimension of it is multiple of two to the number of capacitors.

8.4.3.1 Movements of Particles in PSO

The particles in this case study are multidimensional and can have discrete values only. If one of the dimensions of a particle is some capacitor number or a port number, after the the next iteration it should have only the feasible values which should be some port number (integer values between 5 and 39 except 33) or capacitor number (integer value between 1 and 3702) depending on the dimension. In PSO, in order to move the particles for iterations, there must be a varying directional vector depending upon locally best and globally best particles. Here, in the case of capacitors, they are sorted according to their resonance points. Each decoupling capacitor available in the market, has ESR and ESL values associated with it, which cause the resonance behavior. Figure 8.14 shows the impedance profiles of three capacitors with different resonance points. The resonance points of the decoupling capacitors are used for the movement of the dimensions of the particle with which the capacitor numbers are associated. For example, if a capacitor as one of the dimensions of the globally best particle in one iteration is having resonance at 40 MHz and one of the dimensions of another particle is having its resonance at 180 MHz, the later one will move randomly toward the capacitors having value between 40 and 180 MHz.

In the case of particle dimensions concerned with the port numbers, the movement is decided by the distance between the ports which are found by the coordinates of them. A particle will move to the global best particle and it can have any value or port number (based on random variables defined by PSO) which is having less distance than that from the globally best particle.

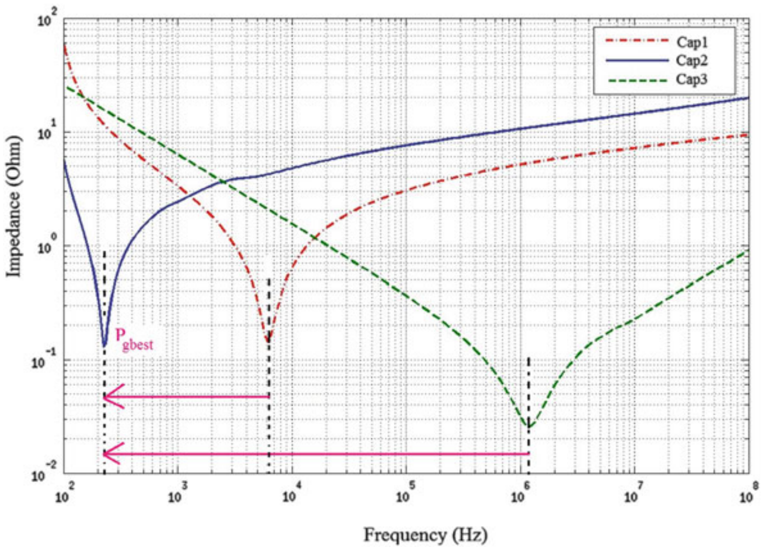


Fig. 8.14 Resonance points of various capacitors defined for guiding the movement of particles

8.4.3.2 Experiments

For applying PSO to this problem, 50 particles are initialized for 20 iterations. Each particle represents 4 capacitors s-parameters files and four corresponding port numbers. The coefficients $p1$ and $p2$ are taken as 1.49 each, and similarly ω_i and ω_f as 0.9 and 0.4. The maximum current of the system is 20 mA, supply voltage is 1.2 V, and the tolerance is 3 % so the target impedance for the system is 1.8 Ω. The frequency range of interest for the analysis is 200 MHz as discussed in the previous section. The steps used for applying PSO are as follows :

8.4.3.3 Pseudocode for PSO

```
1. Define maximum number of iterations $T_{max}$
2. For $D$ dimensions , generate $P$ particles, their respective positions $\mathbf{x}(i, j)$ and
   velocities $\mathbf{v}(i, j)$; $\forall i \in \{1, 2, \ldots, P\}$ and $j \in \{1, 2, \ldots, D\}$, within the lower
   and upper bounds.
3. Calculate the impedance profiles for all the particles as $Z_{eff_i} = (Z_i^{-1} + Z_{decap_i}^{-1})^{-1}$
4. Calculate the maximum impedance peak $max\_imp(i)$ corresponding to all the
   $Z_{eff_i}$ of all $i$ particles.
5. Define all the local particles as local best particles lbest and find the global best
   particle gbest.
6. Loop 1 : while Number of iterations $t \leq T_{max}$.
7. Loop 2 : for i = 1:P
8. Update inertia, velocities, and positions
   $\omega = (\omega_i - \omega_f)\frac{(T_{max})-t}{T_{max}} + \omega_f$
   $\mathbf{v}(i, j) = \omega(t)\mathbf{v}(i, j) + p_1 r_1(\mathbf{lbest}(i, j) - \mathbf{x}(i, j) + p_2 r_2(\mathbf{gbest} - \mathbf{x}(i, j)$
   $\mathbf{x}(i, j) = \mathbf{x}(i, j) + \mathbf{v}(i, j)$
9. Limit the positions and velocities within the lower and upper bounds.
10. End : Loop 2
11. Update $lbest(i, j)$ and $gbest(j)$ accordingly.
12. Increment the counter : $t = t + 1$
13. END : Loop 1
14. Final Solution = gbest.
```

8.4.3.4 Results

The particles converged to a solution which is shown in Table 8.5. In the table, four capacitors and their best positions which are chosen by the algorithm, and their details are given. When used these capacitors at the given ports, the profile of the PDN is below the target impedance 1.75 Ω. The maximum impedance observed from the port 33 is 1.716 Ω at 200 MHz and can be seen in Fig. 8.15.

Table 8.5 The capacitors and their locations

S.No.	Capacitor number	Capacitor name	Manufacturer	Port number
1.	1	GJ821BB31H105KA12	Murata	16
2.	3702	TWAE687M050	AVX	36
3.	2658	TPSB336K006R0450	AVX	16
4.	65	GRM033B31C332KA87	Murata	26

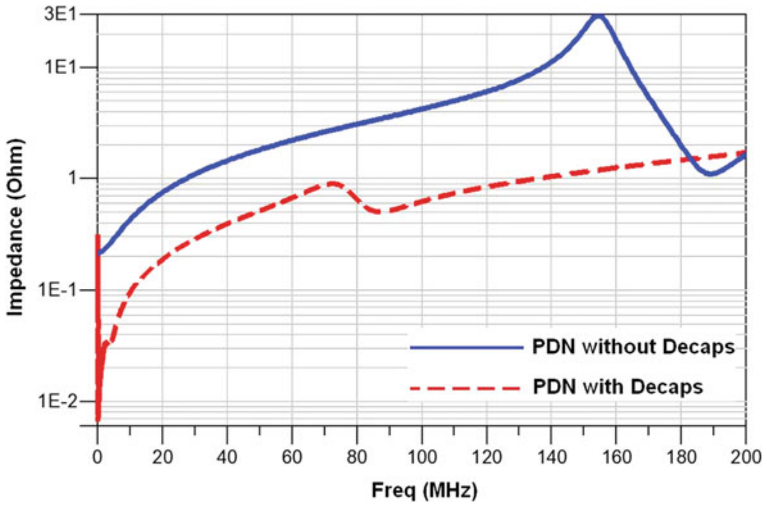


Fig. 8.15 PDN impedance with and without decaps

8.5 Conclusion

Swarm intelligence algorithms are applied successfully for power delivery network design. Decoupling capacitor values and their locations are found by various swarm intelligence algorithms. Their performances are compared for this application. A real-world discrete optimization problem for power integrity has also been discussed. The analysis is done using s-parameter files for more accuracy and to attain realistic approach. A generic methodology is presented for similar power integrity analysis and decoupling network design for any high-speed system.

References

1. Smith, J., Jones Jr, M., Houghton, L., et al.: Future of health insurance. N. Engl. J. Med. **965**, 325–329 (1999)
2. Hubing, T.: Effective strategies for choosing and locating printed circuit board decoupling capacitors. In: International Symposium on Electromagnetic Compatibility, pp. 632–637 (2005)
3. Wu, K-B., et al.: Optimization for the locations of decoupling capacitors in suppressing the ground bounce by genetic algorithm. In: Progress Electromagnetic Research Symposium 2005, pp. 411–415. Hangzhou (August 2005)
4. Kahng, S.: GA-optimized decoupling capacitors damping power bus' cavity-mode resonances. IEEE Microwav. Wirel. Compon. Lett. **16**(6) (2006)
5. Lei, G.T., et al.: High-frequency characterization of power/ground-plane structures. IEEE Trans. Microw. Theory Technol. **47**, 562–569 (1999)
6. Wada, O., et al.: Local decoupling effects of decoupling capacitors on a multilayer PCB: Fast analysis by a closed form expression. IEICE Tech. Rep. EMCJ2000-115, **100** (2000)
7. Yang, X., Li, Z., Mao, J.: Analysis of the bounces on the power/ground plane structures by planar circuit model and APA-E algorithm. IEEE Trans. Adv. Pack. **24**, 184–190 (2001)
8. Chun, S., et al.: Modeling of simultaneous switching noise in high speed systems. IEEE Trans. Adv. Pack. **24**, 132–142 (2001)
9. http://in.mouser.com/Passive-Components/_/N-5g73
10. Linderoth, J.T., Lodi, A.: MILP Software. Wiley Encyclopedia of Operations Research and Management Science. Wiley, London (2010)
11. Belotti, P., Kirches, C., Leyffer, S., et al.: Mixed-integer nonlinear optimization. Acta Numer. **22**, 1–131 (2013)
12. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Progr. **36**(2), 307–339 (1982)
13. Chen, J., He, L.: Efficient in-package decoupling capacitor optimization for I/O power integrity. IEEE Trans. CAD Integr. Circuits Syst. **26**(4) (2007)
14. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, UK (2008)
15. Griewank, A., Walther, A.: Evaluating derivatives: Principles and techniques of algorithmic differentiation, 2nd edn. SIAM, Philadelphia (2008)
16. Yang, X-S., Deb, S.: Cuckoo search via Levy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 210–214 (2009)
17. Yang, X-S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
18. Yang, X.-S.: Firefly algorithms for multimodal optimization. Stoch. Algorithms: Found. Appli. LNCS **5792**, 169–178 (2009)
19. Yang, X-S.: firefly algorithm, levy flights and global optimization. In: Research and Development in Intelligent Systems XXVI, Part 5, pp. 209–218, Springer
20. http://ds.murata.co.jp/software/simsurfing/en-us/index.html
21. Nagpal, R.K., Malik, R.K., Tripathi, J.N.: Signal and power integrity methodology for high speed serial links. In: 12th Euromicro International Conference on Digital System Design 2009, (DSD 2009), Patras (Greece), pp. 742–745, 27–29 August (2009)
22. http://www.murata.com/products/article/pdf/ta1082.pdf (accessed on Aug. 10, 2012)

Part III
Applications

Chapter 9
The Impact of Sensitive Inputs on the Reliability of Nanoscale Circuits

Usman Khalid, Jahanzeb Anwer, Nor H. Hamid and Vijanth S. Asirvadam

Abstract As CMOS technology scales to nanometer dimensions, its performance and behavior become less predictable. Reliability studies for nanocircuits and systems become important when the circuit's outputs are affected by its sensitive noisy inputs. In conventional circuits, the impact of the inputs on reliability can be observed by the deterministic input patterns. However, in nanoscale circuits, the inputs behave probabilistically. The Bayesian networks technique is used to compute the reliability of a circuit in conjunction with the Monte Carlo simulations approach which is applied to model the probabilistic inputs and ultimately to determine sensitive inputs and worst-case input combinations.

9.1 Introduction

MOSFET is the basic component of integrated circuit design. The scaling of MOSFETs has given benefits to the semiconductor industry in terms of performance and productivity [1]. Since last few decades, the scaling of MOSFET transistors has faced many obstacles which have been resolved through novel device architectures and smart engineering solutions. Currently, the fabrication process technology resides at the nanometer level, producing MOSFET transistors in deep submicron[1] dimensions [2, 3]. The International Technology Roadmap for Semiconductors (ITRS) has predicted that a single chip will integrate more than 12 billion transistors in 2020 [3].

[1] Deep submicron design technologies refers to those CMOS circuits which have physical gate length less than 100 nm but greater than 10 nm.

U. Khalid (✉) · N.H. Hamid · V.S. Asirvadam
Universiti Teknologi Petronas, Teronoh, Malaysia
e-mail: usman.khalid.dr@ieee.org

J. Anwer
University of Paderborn, Paderborn, Germany
e-mail: jahanzeb.anwer@upb.de

N.H. Hamid
e-mail: hishmid@petronas.com.my

V.S. Asirvadam
e-mail: vijanth_sagayan@petronas.com.my

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_9

MOSFET transistors are scaled down because it gives advantages in many ways to the semiconductor industry, few of which are mentioned as follows.

- The area of electronic devices decreases as more transistors can fit into similar chip area [4].
- The cost of the chip decreases with the contraction of area of the chip [5].
- The operating frequency and the switching speed of transistors get improved [6].

The reliability of MOSFETs is severely affected by scaling their physical dimensions [7]. The reliability of digital circuits is also affected due to high transient error rates [8–11]. The high transient error rates is the result of domination of noise effects in nanoscale devices like thermal and random telegraph signal noise which did not have compounding effect on earlier high-dimensional technologies [12–15].

A number of research works have already been in progress to investigate fault tolerance schemes that improve the reliability of nanocircuits and systems. In order to stay with the MOSFET technology, fault tolerance schemes are also proposed that achieve reliable MOSFET nanocircuits. Redundancy and Markov Random Field (MRF) are popular techniques that are currently used to build fault tolerant nanocircuits [16–21].

The reliability evaluation schemes cover another phase of research that presents a measure of circuit's reliability, i.e., the percentage of time in which the circuit is expected to work error-free. These reliability measurements are usually performed with reliability evaluation schemes proposed in the previous literature. The popular reliability evaluation schemes include probabilistic gate model (PGM), Boolean difference error-based calculator (BDEC), probabilistic transfer matrix (PTM) and Bayesian networks (BN) [22–28].

The above-mentioned schemes evaluate the overall cumulative reliability of a circuit. The evaluation of cumulative reliability of circuits does not provide enough information for the designers to apply fault tolerance solutions unless they are aware of the reliability at circuit's primary inputs and intermediate stages. The simplest way of modeling output reliability is via definition of inputs vectors, intermediate stages, and outputs. This research work will focus on error-sensitive primary inputs and their effects on nanoscale circuits.

9.2 Probabilistic Inputs

The deterministic input vectors are conventionally applied to a nanoscale circuit. The outputs of such circuit may not be deterministic anymore which is due to various kinds of transient errors such as single event upsets, radiation effects, power supply noise, capacitance, and inductive coupling effects. Therefore, the mechanism of propagation of transient errors can severely affect the upcoming stages of internal circuitry and ultimately degrade the desired reliability of circuit [25, 29–32]. Thus, the scenario of propagation of errors through deterministic inputs will transform into probabilistic inputs and we can formulate a conceptual diagram as shown in Fig. 9.1.

The deterministic inputs, when applied to a nanocircuit, are interpreted as probabilistic inputs. If there are numbers of subcircuits that are connected in series then the

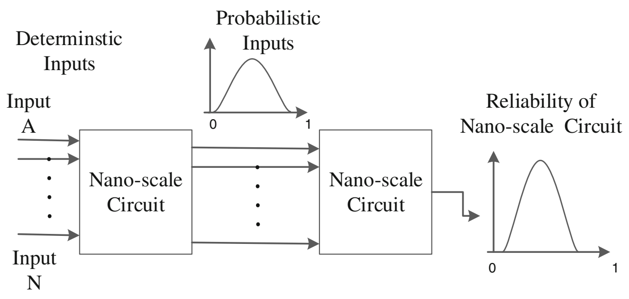


Fig. 9.1 Conceptual diagram of inputs behavior in nanoscale circuits

nature of the input will remain probabilistic. The system performance also become probabilistic in nature. The inputs of conventional digital circuits are binary, i.e., they contain only two logic values, 0 and 1. In contrast, when the circuit design enters into nanoscale level, the logic values of inputs become probabilistic, i.e., the intermediate logic values (between 0 and 1) also have the probability of occurrence greater than zero and less than 1.

The reason of modeling inputs probabilistically is the random (or probabilistic) nature of noise encountered in the nanodevices. The simplest random distribution used for modeling nodes is normal distribution. The probabilistic inputs bounded by 0 and 1 can be called as Probabilistic Inputs (PIs). Figure 9.2 represents an input in the form of normal distribution with a mean value of 0.5 for its' 10,000 Gaussian-distributed randomly selected samples.

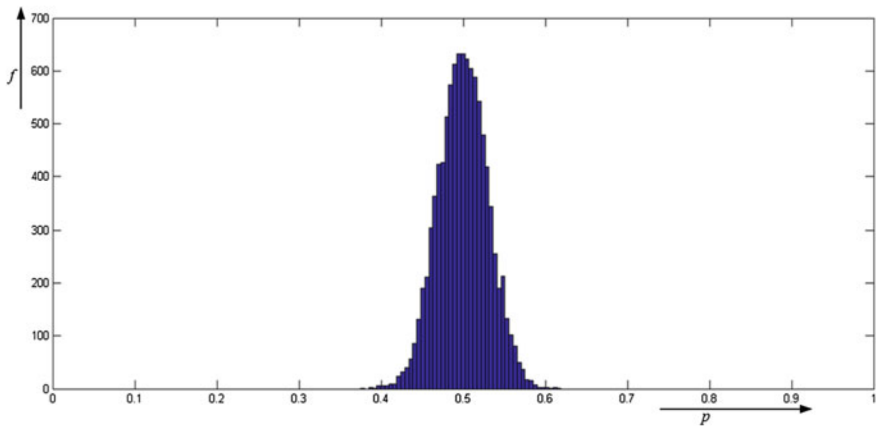


Fig. 9.2 Probabilistic input with 0.5 mean value (The total number of samples on vertical axis, when added across all the logic values near mean 0.5, add up to 10,000)

9.3 Determination of Probabilistic Input Combinations for Higher Reliabilities

The PIs can be applied to any digital circuit to evaluate its reliability in the form of normal distribution. Every PI has a mean value (expectation) which can be any probability value between 0 and 1. If the value of PI is closed to 1 then the normal distribution skews to the right as shown in Fig. 9.3, which for simplicity is called as *positively skewed* PI. On the other hand, if the value of PI is close to 0 then the normal distribution skews to the left-side of the axis as shown in Fig. 9.4, which for simplicity will be called as *negatively skewed* PI in the next sections.

The conceptual diagram of applying PIs on nanoscale circuits has been shown in Fig. 9.5 where three PIs are applied on a test digital circuit which generates the outputs probability distribution as depicted in Fig. 9.6. For each random logic value of input, its corresponding reliability value is evaluated; therefore, for each input, the number of its test samples are equal. This mechanism will be implemented in the form of Monte Carlo Simulation where the inputs are represented as probabilistic distributions and outputs as distributed functions [33, 34]. Figure 9.5 shows the example output reliability of Full Adder (XOR/NAND) circuit with its mean and standard deviation value.

Applying the Monte Carlo simulation mechanism on digital circuits helps to determine the most sensitive inputs as well as the worst-case and best-case input combinations. There will be two cases to determine the sensitive inputs for nanoscale circuits. In the first case, the sensitive inputs observed for the *positively skewed* PI values; whereas in the second case, the sensitive inputs observed for the *negatively skewed* PI values. Furthermore, there will be two cases implemented to determine the worst-case and best-case input combinations. In all cases, Bayesian networks

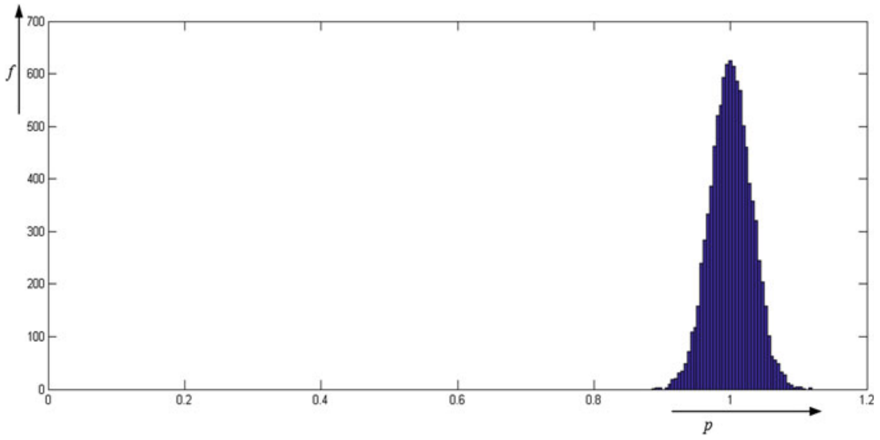


Fig. 9.3 Probabilistic input skew to the *right* with mean at 1 (Positively skewed PI)

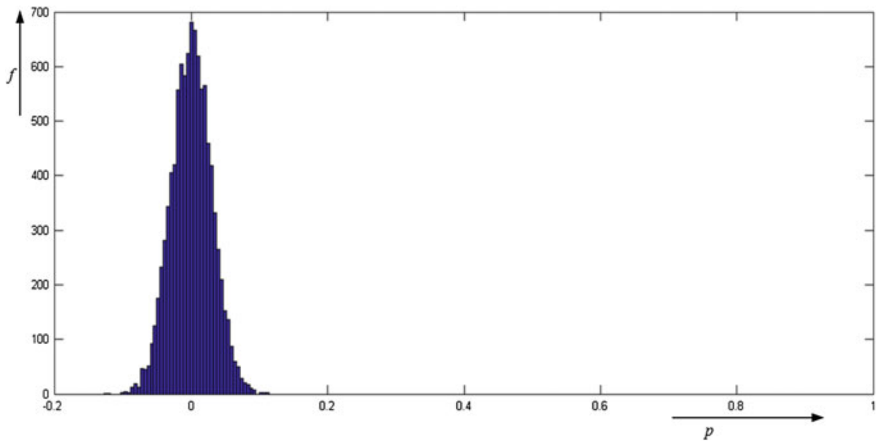


Fig. 9.4 Probabilistic input skew to the *left* with mean at 0 (Negatively skewed PI)

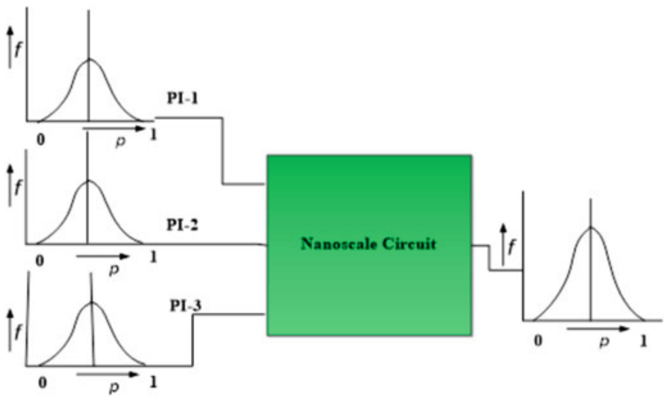


Fig. 9.5 Monte Carlo simulation for nanoscale circuits conceptual diagram

scheme will be used to evaluate the circuits' reliability. The brief methodology to estimate the nanoscale circuits' reliability using Bayesian Networks is as follows.

- First, we convert a digital circuit into Bayesian Networks graph. Consider the original circuit as an ideal circuit or errorfree circuit and make a copy of ideal circuit and represent it in an erroneous subcircuit, in parallel. Then, we will combine the outputs of both original and erroneous circuits with a comparator (XOR gate).
- Transform all the inputs and gates to nodes which will become a connected node network. In this network, every node is directed toward the next node unless we reach up to the output node and this form of graph is called as Acyclic graph.
- Modify the acquired acyclic graph by removing the directions of the all nodes in the network which results in a moral graph.

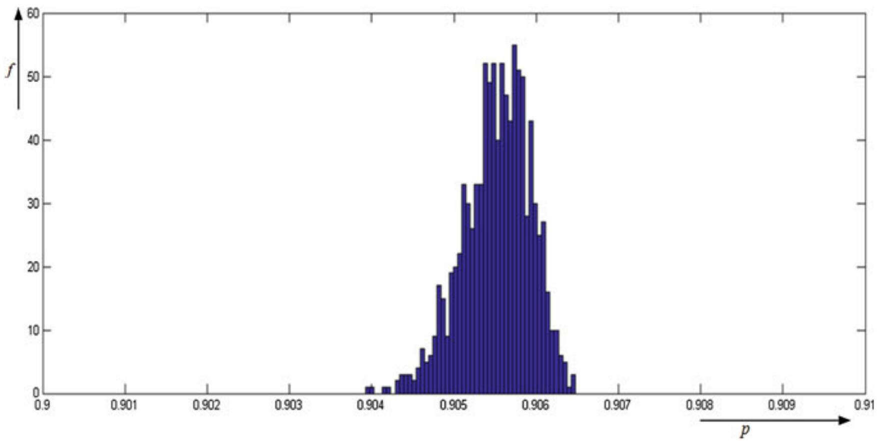


Fig. 9.6 Output reliability of full adder (XOR/NAND) with mean = 0.9055 and standard deviation = 0.0750

- Transform the resultant graph into number of node triangles connected together which will be called as triangulated graph.
- Convert the triangulated network into chunks of number of connected nodes which are called as cliques. This will ultimately form a junction tree graph where all cliques initiate the message passing mechanism.
- Finally, we compute the marginal probabilities of each node of the network to estimate the reliability of the circuit. In case, there is more than one output of a circuit, the resultant reliability value will be estimated by taking the average of all output reliability values.

For in-depth study of Bayesian Networks methodology implementation on digital circuits, interested reader may refer to [35–37]. The strategy to implement all cases has been discussed in detail in the following sections.

9.3.1 Determination of Sensitive Inputs for Positively Skewed PIs

The reliability curve for this scenario is obtained using Monte Carlo simulation method for all the positively skewed PIs. To determine the most sensitive input of a circuit, we fix all the PIs toward positive skew level (which is close to 1) except one of the varying PI for three cases; such as 0.5 (zero skew), 0.1 (negatively skewed) and 0.9 (positively skewed) mean values for calculating the reliability of the circuit.

The above-mentioned method will be repeated for all PIs of the circuit. The reliability results show that the highest reliability of the output is obtained for PIs other than the zero-skew combination of the inputs. The target is to find the lowest reliability value and the corresponding PI combination because the lowest reliability indicates the corresponding input which will be the sensitive input to the circuit. If there is more than one input combination giving the same lowest reliability then there is a need to compare their standard deviations. For comparison of two or more lowest reliability PIs, choose the PI with largest standard deviation which indicates the most sensitive input of the circuit. The low standard deviation value represents a stable reliability curve.

9.3.2 Determination of Sensitive Inputs for Negatively Skewed PIs

To determine the most sensitive input of a circuit for negatively skewed PIs, we fix all PIs close to zero except one of the varying PI. This procedure will be repeated for all PIs of the circuit and record the reliability curve. After getting all the reliability results, there is a need to look for the PI combination with the highest reliability value. So if there are two or more combinations giving the same highest reliabilities then there is a need to compare their standard deviations and choose the highest standard deviation value (the reason is described in previous subsection).

9.3.3 Worst-Case and Best-Case Input Combinations

To find the worst-case input combination, it is required to evaluate the reliability by selecting the PIs in four different scenarios. In the first scenario, all PIs will be fixed close to one and evaluate the reliability of circuit. For the second scenario, the reliability of circuit is selected by fixing all PIs close to zero. The third scenario is to fix all the PIs toward positive skew level and vary one PI with different values and evaluate the reliability. The fourth scenario is to fix all the PIs toward negative skew level (close to zero) and vary one PI with different values and evaluate the reliability. Record all the reliability values for the circuit and find out the lowest reliability value. The corresponding input combination which gives the lowest reliability value is the worst case input combination of the circuit.

On the other hand, the best-case input combination can be determined by computing the reliability values with the same scenarios as discussed in the worst-case input combination. However, the objective here will be to find out the combination which gives the highest reliability for a circuit.

9.4 Simulation Results and Analysis

The simulations are performed in MATLAB in conjunction with Bayes Net Toolbox (BNT) [38]. As mentioned in the previous section, we have implemented Bayesian networks algorithm in order to compute reliability values used with Monte Carlo

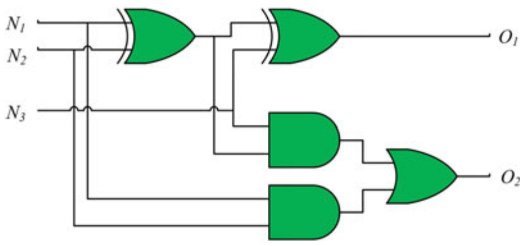


Fig. 9.7 Full adder (XOR/NAND) circuit

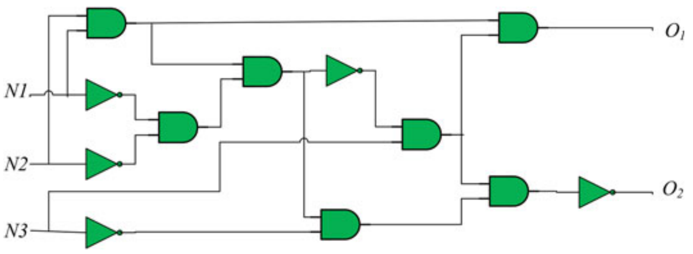
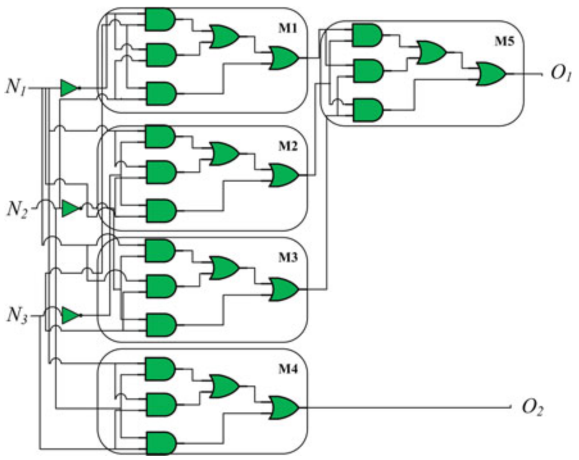


Fig. 9.8 Full adder (NAND) circuit

Simulation technique. By applying Monte Carlo simulation mechanism, the sensitive input combinations can be determined for the small-scale digital circuit. There are two types of circuits which have been chosen for simulation analysis. The first type of the circuits have similar functional output while second type has different functionality circuits. The similar functionality (logic) circuits, Full adder (XOR/NAND), Full adder (NAND), and Full adder (Majority) are shown in Figs. 9.7, 9.8 and 9.9, respectively. The different functionality circuits C17 (benchmark LGSynth'93 series) and 2–4 Decoder circuit are shown in Figs. 9.10 and 9.11, respectively.

Fig. 9.9 Full adder (Majority) circuit



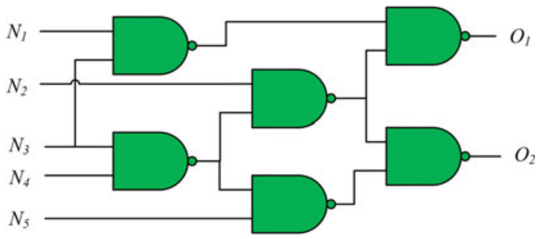


Fig. 9.10 C17 LGSynth93 benchmark circuit

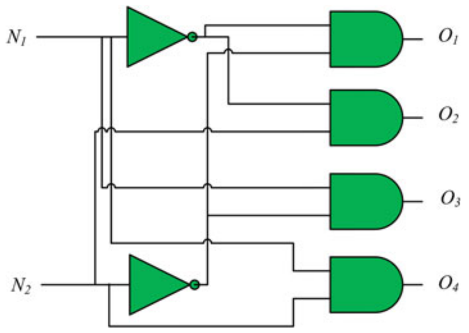


Fig. 9.11 Decoder 2–4 circuit

There are 10,000 samples taken for each PI to be applied on any digital circuit and there will be the same number of samples of the reliability at the output as shown in Fig. 9.5. The results of reliability and standard deviations are obtained with fourth decimal accuracy in order to converge with the nominal values of each simulation. The nominal (i.e., deterministic probability values without a distribution of Monte Carlo Gaussian samples) reliability simulation results for each PI case are equal to the mean values of each PI reliability value; hence, in order to keep nonredundant results they are not reported in the following tables. If the results of nominal reliability values with Monte Carlo reliability values do not converge, then one has to perform more than 10,000 iterations of Monte Carlo simulations in order to get better accuracy. The reliability evaluation has been performed as discussed in Sect. 9.3 using Bayesian networks. The Monte Carlo simulations are run on Core 2 Duo, 2.7 GHz with 4 Gb RAM which takes average of around 9 h for each PI case.

The results are evaluated based on three types PI mean values. The first type of PI value is taken 0.5 mean which is the center of the normal distribution (not skewed). The second type of PI value is taken 0.1 mean which is close to zero (negatively skewed). The third type of PI value is taken 0.9 which is (positively skewed). The three different PI mean values will be given as input to digital circuits in such a way where there is only one PI which change its value several times. But all the rest of the PIs will have a fix mean value at every time of evaluation of reliability.

Table 9.1 Full adder (XOR/NAND) circuit's sensitive inputs reliability for positively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.9	0.9	0.9	0.9055	0.0750
0.5	0.9	0.9	0.9020	0.0012
0.1	0.9	0.9	0.8980	0.0022
0.9	0.5	0.9	0.9021	0.0012
0.9	0.1	0.9	0.8980	0.0023
0.9	0.9	0.5	0.9028	0.1749
0.9	0.9	0.1	0.8997	0.0017

Table 9.2 Full adder (NAND) circuit's sensitive inputs reliability for positively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.9	0.9	0.9	0.8428	0.0066
0.5	0.9	0.9	0.7734	0.0070
0.1	0.9	0.9	0.6526	0.0179
0.9	0.5	0.9	0.7736	0.0066
0.9	0.1	0.9	0.6522	0.0180
0.9	0.9	0.5	0.8042	0.0045
0.9	0.9	0.1	0.7335	0.0111

9.4.1 Simulation Results of Sensitive Inputs for Positively Skewed PIs

Tables 9.1, 9.2, 9.3, 9.4 and 9.5 show results only for the higher values of PIs. Table 9.1 shows that Full adder (XOR/NAND) reliability and its standard deviation for the higher values of PI. To determine the sensitive input combination, the focus will be toward finding the lowest reliability value in the table. There are two PI combinations which gives the same lowest reliability values. The PI combinations are 0.1, 0.9, 0.9 and 0.9, 0.1, 0.9, and the corresponding reliability value is 0.8980.

But when two or more reliability values are same then there is need to consider their corresponding standard deviation values of the reliability. The PI combinations 0.1, 0.9, 0.9, and 0.9, 0.1, 0.9 have different standard deviation values. The standard deviation value of PI with combination 0.9, 0.1, 0.9 is large compare to other PI's standard deviation value. This analysis indicates that the most sensitive PI of the circuit combination is 0.9, 0.1, 0.9. The corresponding PI is the second PI which is the most sensitive input to the Full adder (XOR/NAND) circuit.

Table 9.3 Full adder (Majority) circuit's sensitive inputs reliability for positively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.9	0.9	0.9	0.8755	0.0072
0.5	0.9	0.9	0.8098	0.0069
0.1	0.9	0.9	0.7099	0.0100
0.9	0.5	0.9	0.8015	0.0060
0.9	0.1	0.9	0.7276	0.0056
0.9	0.9	0.5	0.8551	0.0089
0.9	0.9	0.1	0.7604	0.0249

The varying PI-1 reliability has two standard deviation values which are 0.0070 and 0.0179 and for the varying PI-2 have 0.0066 and 0.0180 standard deviation values, respectively. The PI's standard deviation value is 0.0004 larger in magnitude than the PI-2's standard deviation value. The PI-2's reliability is only 0.0001 larger than the first PIs standard deviation since the PI-1's standard deviation of reliability is larger than the PI-2's. Hence, the PI-1 is the most sensitive input to the circuit.

The Full adder (NAND) reliability and its standard deviation for the higher values of PI are shown in Table 9.2. There are two PIs which gives almost the same low reliability values so there is need to observe their standard deviation value. The first varying PI gives 0.6526 and 0.7734 reliability values and second varying PI gives 0.6522 and 0.7736, respectively. Both PIs reliability values are almost equally large from each other.

The Full Adder (Majority) reliability and its standard deviation for the higher values of PI are shown in Table 9.3. Again there are two PIs which give almost the same low reliability values. The PI-1 gives 0.8098 and 0.7099 reliability values and second varying PI gives 0.8015 and 0.7276, respectively. Both PIs reliability values are almost equally large from each other. The varying PI-1's reliability has two standard deviation values which are 0.0069 and 0.0100. The PI-2's reliability has also two standard deviation values which are 0.0060 and 0.0056. The PI-1's standard deviation is larger than the PI-2's standard deviation of reliability by observing their difference from 0.0069 to 0.0060 and 0.0100 to 0.0056, respectively. The PI-1's standard deviation is 0.0009 and 0.0044 larger than the second PI standard deviation values, respectively. Therefore, PI-1 becomes the most sensitive input to the Full Adder (Majority) circuit.

Table 9.4 shows the Decoder 2–4 reliability values and standard deviation values for higher values of PIs. This circuit has only two inputs and both give almost the same low reliability values. The varying PI-1 gives 0.9204 and 0.9069 reliability values. The varying PI-2 gives 0.9202 and 0.9068 reliability values. Both PIs reliability values are almost equally large from each other.

Table 9.4 Decoder 2–4 circuit's sensitive inputs reliability for positively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.9	0.9	0.9069	0.0052
0.5	0.9	0.9204	0.0050
0.1	0.9	0.9069	0.0051
0.9	0.5	0.9202	0.0057
0.9	0.1	0.9068	0.0053

Table 9.5 C17 circuits sensitive inputs reliability for positively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	PI-4 (Mean)	PI-5 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.9	0.9	0.9	0.9	0.9	0.8356	0.0088
0.5	0.9	0.9	0.9	0.9	0.8094	0.0114
0.1	0.9	0.9	0.9	0.9	0.7592	0.0183
0.9	0.5	0.9	0.9	0.9	0.8291	0.0081
0.9	0.1	0.9	0.9	0.9	0.8230	0.0068
0.9	0.9	0.5	0.9	0.9	0.8758	0.0027
0.9	0.9	0.1	0.9	0.9	0.8955	0.0016
0.9	0.9	0.9	0.5	0.9	0.8875	0.0026
0.9	0.9	0.9	0.1	0.9	0.9115	0.0018
0.9	0.9	0.9	0.9	0.5	0.8336	0.0087
0.9	0.9	0.9	0.9	0.1	0.8313	0.0086

The PI-1's reliability has two standard deviation values which are 0.0050 and 0.0051 and for the PI-2's reliability has 0.0057 and 0.0053 standard deviation values, respectively. The PI-2's standard deviation is larger than the PI-1's standard deviation by observing their difference of 0.0007 and 0.0002, respectively. So the PI-2 is the most sensitive input to the Decoder 2–4 circuit.

The LGSynth93 series C17 circuit reliability and standard deviation results have been shown in Table 9.5. The PI combination 0.1, 0.9, 0.9, 0.9, 0.9 gives the lowest reliability value. As discussed earlier in previous section that if a circuit gives the lowest reliability value for single PI then there is no need to observe their standard deviation value. Hence, the PI-1 is the most sensitive input to the C17 circuit.

9.4.1.1 Discussion

The results show the concept of determining the sensitive inputs for positively skewed PIs. In multiple-logic path where the inputs of one circuit is connected to the output

Table 9.6 Full adder (XOR/NAND) circuit's sensitive inputs reliability for negatively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.1	0.1	0.1	0.7067	0.0101
0.5	0.1	0.1	0.7695	0.0133
0.9	0.1	0.1	0.8076	0.0145
0.1	0.5	0.1	0.7690	0.0138
0.1	0.9	0.1	0.8078	0.0144
0.1	0.1	0.5	0.7485	0.0150
0.1	0.1	0.9	0.7716	0.0170

of the preceding circuit which is having most of the inputs on high logic except one low logic input then that particular low logic input will give the lowest circuit's reliability value. The results show the most of the reliability values have higher values which shows that the circuits are less error-prone to positively skewed PIs. Therefore, all the test circuits show that only some particular sensitive inputs gives the lowest reliability and unstable standard deviation values. Both same functionality and different functionality circuits have different sensitive inputs for high logic input levels.

9.4.2 Simulation Results of Sensitive Inputs for Negatively Skewed PIs

Tables 9.6, 9.7, 9.8, 9.9 and 9.10 show the results for the lower values of PIs. The Full Adder (XOR/NAND) reliability values and standard deviation results have been shown in Table 9.6. In order to determine the sensitive input combination, the focus will be on to finding the highest reliability value in the table. There are two PIs which have almost same higher reliability values. The varying PI-1's reliability values are 0.7695 and 0.8076. The PI-2's reliability value is 0.7690 and 0.8078. Both PI's reliability values are almost equally large from each other.

So there is a need to focus on PI's standard deviation values. The varying PI-1 has two standard deviation values which are 0.0133 and 0.0145, and the varying PI-2's has also two standard deviation values which are 0.0138 and 0.0144. The PI-2's standard deviation of value is 0.0005 larger than the first PI's standard deviation of reliability. The PI-2's reliability is 0.0001 larger than the PI-1's corresponding standard deviation of reliability. Comparatively the PI-2's standard deviation of reliability value is larger than the PI-1's standard deviation value. Hence, the PI-1 is the most sensitive input to the Full Adder (XOR/NAND) circuit.

Table 9.7 Full adder (NAND) circuit's sensitive inputs reliability for negatively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.1	0.1	0.1	0.7091	0.0114
0.5	0.1	0.1	0.7844	0.0050
0.9	0.1	0.1	0.8280	0.0054
0.1	0.5	0.1	0.7845	0.0051
0.1	0.9	0.1	0.8279	0.0051
0.1	0.1	0.5	0.7086	0.0122
0.1	0.1	0.9	0.7593	0.0047

The Full Adder (NAND) reliability values and standard deviation results are shown in Table 9.7. The PI-1 gives 0.7844 and 0.8280 reliability values and PI-2 gives 0.7845 and 0.8279, respectively. Both PIs reliability values are almost equally large from each other.

The PI-1's reliability has two standard deviation values which are 0.0050 and 0.0054 and for the PI-2's has 0.0051 and 0.0051 standard deviation values, respectively. The PI-1's standard deviation value is 0.0001 less but 0.0003 larger than the PI-2. So the PI-1 has large standard deviation value and becomes the most sensitive input to the Full Adder (NAND) circuit.

The Full Adder (Majority) reliability and its standard deviations for the negatively skewed PIs has shown in Table 9.8. The PI-3 gives the highest reliability in the table which is 0.7011. The respective PI combination is 0.1, 0.1, 0.9. There is no need to observe the standard deviation values for this circuit as PI-3 is already giving much higher reliability value as compare to the rest of PI combinations. Therefore, the PI-3 is the most sensitive PI to the Full Adder (Majority) circuit.

Table 9.8 Full adder (Majority) circuit's sensitive inputs reliability for negatively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.1	0.1	0.1	0.7089	0.0115
0.5	0.1	0.1	0.6156	0.0106
0.9	0.1	0.1	0.6783	0.0122
0.1	0.5	0.1	0.5839	0.0087
0.1	0.9	0.1	0.5882	0.0102
0.1	0.1	0.5	0.6435	0.0079
0.1	0.1	0.9	0.7011	0.0068

Table 9.9 Decoder 2–4 circuit's sensitive inputs reliability for negatively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.1	0.1	0.9069	0.0051
0.5	0.1	0.9203	0.0051
0.9	0.1	0.9067	0.0053
0.1	0.5	0.9205	0.0055
0.1	0.9	0.9069	0.0051

Table 9.9 shows the Decoder 2–4 reliability values and standard deviation values for the lower values of PIs. The PI-1's reliabilities are 0.9203 and 0.9067. The PI-2's reliabilities are 0.9205 and 0.9069. Here both input reliabilities are almost equally large from each other. The PI-1's standard deviation values are 0.0051, 0.0053, and for the PI-2's standard deviation values are 0.0055 and 0.0051. The PI-2's standard deviation is 0.0004 large but 0.0002 is less than the PI-1. So the PI-2 is the most sensitive PI to the Decoder 2–4 circuit.

The C17 circuit's reliability and standard deviation results have been shown in Table 9.10. The highest reliability is 0.8933 which is given by the second varying PI. And the respective PI combination is 0.1, 0.9, 0.1, 0.1, 0.1. As discussed in previous section, that if a circuit gives the highest reliability value for one input then there is no need to consider their standard deviation values. So finally the PI-2 is the most sensitive input to the C17 circuit.

Table 9.10 C17 circuit's sensitive inputs reliability for negatively skewed PIs

PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	PI-4 (Mean)	PI-5 (Mean)	Reliability (Mean)	Standard deviation of reliability
0.1	0.1	0.1	0.1	0.1	0.8356	0.0088
0.5	0.1	0.1	0.1	0.1	0.8094	0.0114
0.9	0.1	0.1	0.1	0.1	0.7592	0.0183
0.1	0.5	0.1	0.1	0.1	0.8291	0.0081
0.1	0.9	0.1	0.1	0.1	0.8230	0.0068
0.1	0.1	0.5	0.1	0.1	0.8758	0.0027
0.1	0.1	0.9	0.1	0.1	0.8955	0.0016
0.1	0.1	0.1	0.5	0.1	0.8875	0.0026
0.1	0.1	0.1	0.9	0.1	0.9115	0.0018
0.1	0.1	0.1	0.1	0.5	0.8336	0.0087
0.1	0.1	0.1	0.1	0.9	0.8313	0.0086

9.4.2.1 Discussion

The results show the concept of determining the sensitive inputs for negatively skewed PIs. If one circuit is connected to the output of the following circuit which is having most of the inputs on low logic except one high logic input then that particular high logic input is giving the abrupt highest reliability of the circuit. All the test circuits show the particular sensitive inputs with highest reliability and unstable standard deviation values.

9.4.3 Overall Sensitive Inputs of Circuits for Higher and Lower Values of PIs

Table 9.11 shows the summary of all sensitive inputs of all test circuits for higher and lower values PIs in second and third column of table, respectively.

9.4.3.1 Discussion

The results show that both, same and different, functionality circuits have the different sensitive PIs. If a circuit has different architecture, then the sensitive input will be dependent on its structure and as well as the input logic levels. However, there is one thing common in all results that the lower values of PI combinations are giving low reliability values. This observation shows that low logic value inputs are more sensitive and error-prone to the circuit and helps in determining the worst-case input combinations.

Table 9.11 Sensitive inputs of nanoscale circuits for both higher and lower values of probabilistic inputs

Circuit name	Sensitive (Higher values of PIs)	Sensitive (Lower values of PIs)
Full adder (XOR/NAND)	2nd input (N2)	2nd input (N2)
Full adder (NAND)	1st input (N1)	1st input (N1)
Full adder (Majority)	1st input (N1)	3rd input (N3)
2–4 decoder	2nd input (N2)	2nd input (N2)
C17	1st input (N1)	2nd input (N2)

9.4.4 Simulation Results of Worst-Case and Best-Case Combinations

Tables 9.12 and 9.13 show the worst case and the best-case input combinations, respectively. The worst case input combination can be chosen when a circuit gives the lowest reliability value. On the other side, the best-case input combination can be chosen when the circuit gives the highest reliability value. Tables 9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 9.7, 9.8, 9.9 and 9.10 also show some worst-case and the best-case PI combinations. However, we have considered all possible combinations in order to find the worst and best-case combinations, e.g., a circuit that has three inputs (like any of the presented full adder), can have up to 27 different input combinations (normal, positively skewed, and negatively skewed). The following tables are the summary of all worst and best-case PI combinations for test circuits.

Table 9.12 shows the worst-case input combinations for every test circuit. Full Adder (XOR/NAND) and Full Adder (NAND) circuits have the same worst-case input combination which also represents the low logic level combination. Full Adder (Majority) has the different worst-case input combination which shows that the low reliability value is not necessarily represented by all low logic level input values. It can be any input logic value combination which is producing the lowest reliability value. The Decoder 2–4 and C17 circuit also have different input logic combination.

Table 9.13 shows the best-case input combinations for all test circuits. Full Adder (XOR/NAND), Full Adder (NAND), and Full Adder (Majority) circuits have the same best-case input combination which represents that all input combinations are on high logic level. Both Decoder 2–4 and C17 circuits have the different best-case input combination values which shows that the high reliability value is not necessarily represented by all high logic level input values. It can be the combination of both high and low input logic values which are producing the highest reliability values.

9.4.4.1 Discussion

The worst-case and best-case input combinations are also dependent on the logic level of output of preceding connected circuit as well as the circuit's own functionality

Table 9.12 Worst-case PI combinations for selected test circuits

Circuit name	Worst-case (Lowest reliabilities)	PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	PI-4 (Mean)	PI-5 (Mean)
Full adder (XOR/NAND)	0.7067	0.1	0.1	0.1	–	–
Full adder (NAND)	0.7091	0.1	0.1	0.1	–	–
Full adder (Majority)	0.5839	0.1	0.5	0.1	–	–
2–4 decoder	0.9067	0.9	0.1	–	–	–
C17	0.7363	0.1	0.1	0.1	0.9	0.1

Table 9.13 Best-case PI combinations for selected test circuits

Circuit name	Best-case (Highest reliabilities)	PI-1 (Mean)	PI-2 (Mean)	PI-3 (Mean)	PI-4 (Mean)	PI-5 (Mean)
Full adder (XOR/NAND)	0.9055	0.9	0.9	0.9	–	–
Full adder (NAND)	0.8428	0.9	0.9	0.9	–	–
Full adder (Majority)	0.8755	0.9	0.9	0.9	–	–
2–4 decoder	0.9204	0.5	0.9	–	–	–
C17	0.9115	0.9	0.9	0.9	0.1	0.9

and architecture. It is not necessary that all low logic level combinations will give the worst case input combinations or all high logic level combinations will give the best-case input combinations. This shows that there can be any random input combination of both high and/or low logic level values. But overall, this also shows that higher values of PI combinations always give the higher reliability values and vice versa. So the nanoscale circuits are more sensitive to the lower logic input combinations.

9.4.5 Enhancement in Reliability by using Appropriate Deterministic Inputs

The reliability of nanoscale circuits can be enhanced by replacing the conventional inputs with the appropriate deterministic inputs (inputs having deterministic probability i.e., without Monte Carlo Gaussian samples) by having the same logic of the circuit. These inputs can be computed by changing the conventional input probability from 0.5 (which is default) for both logic 0 and logic 1, to any appropriate input probability value. In order to compute the appropriate input probability values, the probability values are varied from 0.1 to 0.9. The analysis haven been performed on all selected circuits in order to find the most appropriate inputs to replace where the maximum reliability values can be achieved. Figure 9.12 shows the graph of circuits for possible reliability values on vertical axis with deterministic input probability values on horizontal axis.

All selected circuits have different appropriate inputs where they achieve maximum reliability values. Full adder circuits FA-I (XOR/NAND), FA-II (NAND), and FA-III (Majority) gained the highest possible reliability when all inputs are replaced to 0.9 and above to probability 1 (observed by using polynomial trend-lines). However, C17 and Decoder 2–4 circuits achieved the replaced input probability at 0.6 and 0.5, respectively. The appropriate deterministic inputs can be used in order to achieve optimized reliability values by simply replacing the state-of-the-art deterministic inputs with the determined appropriate inputs.

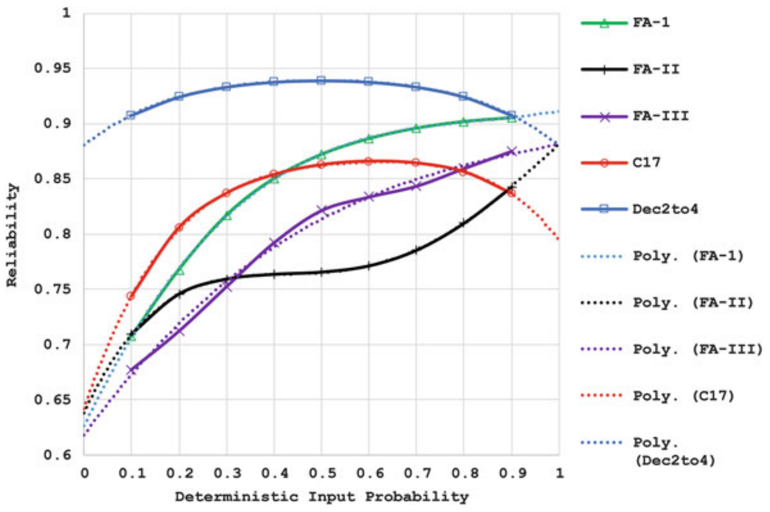


Fig. 9.12 Enhanced reliability by changing deterministic inputs

9.5 Summary

In this chapter, we described how to model the inputs in the form of PIs in nanoscale circuits. The purpose to represent inputs of nanoscale circuits as PIs is to analyze their probabilistic behavior and effects on the circuit's output reliability. The phenomenon of PIs used in this work allows the designer to consider not only the circuit's intermediate stages but also its input vector to analyze their effect on circuits reliability. A PI has a range of Gaussian random samples for every single input for Monte Carlo analysis. The PIs propagate in the circuit and provide a probabilistic output as a range of random numbers (normal distribution) with a mean and standard deviation.

The variation at the output let the circuit designer determine the sensitive inputs of his circuit. The determination of sensitive inputs including worst-case input combinations is the key information required before application of fault tolerance alternative to the circuit. This analysis also provides the best case input combinations which gives the highest possible reliability of the circuit. Moreover, we have analyzed that by using appropriate deterministic inputs instead of conventional inputs for truth table combinations, we can raise the reliability to a higher level. The fault tolerant schemes such as various redundancy techniques can be further applied to the internal architecture in order to get the further highest possible reliability value for a circuit.

References

1. Zhang, K.: Challenges and opportunities in nano-scale VLSI design, International Symposium on VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT). IEEE VLSI-TSA **6**(7), 27–29 (2005)
2. Siewiorek, D.P., Swarz, R.S.: Reliable Computer Systems: Design and Evaluation. AK Peters, Natick (1998)
3. ITRS. International Technology Roadmap for Semiconductors (ITRS). Available via DIALOG. http://www.itrs.net/ (2013)
4. Stanisavljevic, M.: Optimization of nanoelectronic systems reliability by reducing logic depth. In: Stanisavljevic, M., Schmid, A., Leblebici, Y. (eds.) Fourth International Conference on Nano-Nets, Nano-Net. Lucerne, Switzerland (2009)
5. Bhaduri, D.: Design and analysis of defect- and dault-tolerant nano-computing systems. Ph.D. dissertation, Virginia Polytechnic Institute and State University, etd-03182007-194734 (2007)
6. Khakifirooz, A., Antoniadis, D.A.: MOSFET Performance Scaling—Part II: Future Directions. IEEE Trans. Electron Dev. **55**(6), 1401–1408 (2008)
7. Rusu, S.: Trends and scaling in VLSI technology scaling towards 100nm (Invited Paper). In: Rusu, S. (eds.) European Solid-State Circuits Conference (ESSCIRC) (2001)
8. Huang, C.: Robust Computing with Nanoscale Devices: Progresses and Challenges. Springer, New York (2010)
9. Stanisavljevic, M., Schmid, A., Leblebici, Y.: Reliability of Nanoscale Circuits and Systems: Methodologies and Circuit Architectures. Springer, New York (2010)
10. Ryhnen, T., Uusitalo, M.A., Ikkala, O., Krkkinen, A.: Nanotechnologies for Future Mobile Devices. Springer, Cambridge (2010)
11. Casati, G., Matrasulov, D.: Complex Phenomena in Nanoscale Systems. Springer, New York (2009)
12. Wooley, J.C., Lin, H.: Catalyzing Inquiry at the Interface of Computing and Biology. National Academies Press, Washington (2005)
13. Tehranipoor, M.: Emerging Nanotechnologies: Test, Defect-Tolerance and Reliability, Frontiers in Electronic Testing, vol. 37. Springer, New York (2008)
14. Bahar, R.I., Mundy, J., Chan, J.: A probabilistic based design methodology for nanoscale computation. In: International Conference on Computer Aided Design (IC- CAD), pp. 480–486 (2003)
15. Anwer, J., Khalid, U., Singh, N., Hamid, N.H., Asirvadam, V.S.: A novel error-detection mechanism for digital circuits using Markov random field modelling. In: IEEE International Conference on Computational Intelligence and Communication Networks (CICN), pp. 883–886 (2012)
16. Nepal, K., Bahar, R.I., Mundy, J., Patterson, W.R., Zaslavsky, A.: MRF reinforcer: a probabilistic element for space redundancy in nanoscale circuits. IEEE Micro, IEEE Comput. Soc. Mag. **26**(5), 19–27 (2006)
17. Nepal, K.: Markov random field: in designing reliable nanoscale circuits using principles of Markov random fields. Ph.D. dissertation. Brown University
18. Anwer, J., Khalid, U., Singh, N., Hamid, N.H., Asirvadam, V.S.: Highly noise-tolerant design of digital logic gates using Markov random field modelling. In: International Conference on Electronic Computer Technology (ICECT), pp. 24–28 (2010)
19. Anwer, J., Shaukat, S.F., Khalid, U., Hamid, N.H.: Reliable area index: a novel approach to measure reliability of Markov random field based circuits. In: International Conference on Intelligent and Advanced Systems (ICIAS), pp. 24–28 (2012)
20. Anwer, J., Fayyaz, A., Masud, M.M., Shaukat, S.F., Hamid, N.H.: Fault-tolerance and noise modelling in nanoscale circuit design. In: International Symposium on Signals Systems and Electronics (ISSSE), vol. 2, pp. 1–4 (2010)
21. Anwer, J., Khalid, U., Singh, N., Hamid, N.H., Asirvadam, V.S.: Joint and marginal probability analyses of Markov random field networks for digital logic circuits. In: International Conference on Intelligent and Advanced Systems (ICIAS), pp. 1–4 (2010)

22. Han, J., Taylor, E., Gao. J., Fortes, J.: Faults, error bounds and reliability of nanoelectronic circuits. In: Proceedings of 16th International Conference on Application-Specific Systems, Architecture and Processors (ASAP), pp. 247–253 (2005)
23. Mohyuddin, N., Pakbaznia, E., Pedram, M.: Probabilistic error propagation in logic circuits using the boolean difference calculus. In: Proceedings of 26th International Conference on Computer Design (ICCD), pp. 7–13 (2008)
24. Krishnaswamy, S., Viamontes, G.F., Markov, I.L., Hayes, J.P.: Probabilistic transfer matrices in symbolic reliability analysis of logic circuits. ACM Trans. Des. Autom. Electron. Syst. (TODAES) **13**(1), 7–13 (2008)
25. Rejimon, T., Lingasubramanian, K., Bhanja, S.: Probabilistic error modeling for nano-domain logic circuits. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **17**(1), 5565 (2009)
26. Han, J., Chen, H., Liang, J., Zhu, P., Yang, Z., Lombardi, F., Lombardi, F.: A stochastic computational approach for accurate and efficient reliability evaluation. IEEE Trans. Comput. **63**(6), 1336–1350 (2014)
27. Singh, N.S.S., Hamid, N.H., Asirvadam, V.S., Khalid, U., Anwer, J.: Evaluation of circuit reliability based on distribution of different signal input patterns. In: IEEE 8th International Colloquium on Signal Processing and Its Applications (CSPA), pp. 5–9 (2012)
28. Singh, N.S.S., Hamid, N.H., Asirvadam, V.S., Khalid, U., Anwer, J.: Sensitivity analysis of probability transfer matrix (PTM) on same functionality circuit architectures. In: IEEE 8th International Colloquium on Signal Processing and Its Applications (CSPA), pp. 250–254 (2012)
29. Zhao, C., Bai, X., Dey, S.: Evaluating transient error effects in digital nanometer circuits. IEEE Trans. Reliab. **56**(3), 381–391 (2007)
30. Mavis, D.G., Eaton, P.H.: SEU and SET modeling and mitigation in deep submicron technologies. In: IEEE international Reliability Physics Symposium (IRPS), pp. 293–305, 15–19 (2007)
31. Veeravalli, V.S., Polzer, T., Steininger, A., Schmid, U.: Architecture and design analysis of a digital single-event transient/upset measurement chip. In: 15th Euromicro Conference on Digital System Design (DSD), pp. 8–17, 5–8 (2012)
32. Ferlet-Cavrois, V., Massengill, L.W., Gouker, P.: Single event transients in digital CMOS a review. IEEE Trans. Nucl. Sci. **60**(3), 1767–1790 (2013)
33. Sulieman, M., Beiu, V.: Design and analysis of SET circuits: using MATLAB modules and SIMON. In: 4th IEEE Conference on Nanotechnology, pp. 618–621, 16–19 (2004)
34. Landau, D.P., Binder, K.: A Guide to Monte Carlo Simulations in Statistical Physics, pp. 384. Cambridge University Press, Cambridge (2009)
35. Khalid, U., Anwer, J., Singh, N., Hamid, N.H., Asirvadam, V.S.: Reliability-evaluation of digital circuits using probabilistic computation schemes. In: National Postgraduate Conference (NPC), pp. 19–20 (2011)
36. Khalid, U., Anwer, J., Singh, N., Hamid, N.H., Asirvadam, V.S.: Computation and analysis of output error probability for C17 benchmark circuit using Bayesian networks error modeling. In: IEEE Student Conference on Research and Development (SCOReD), pp. 348–351 (2011)
37. Khalid, U.: Reliability-evaluation of nanoscale circuit design using Bayesian networks. Masters thesis, Universiti Teknologi Petronas. http://utpedia.utp.edu.my/id/eprint/3315 (2012)
38. BNT. Bayes Net Toolbox for Matlab (BNT). Available via DIALOG. http://code.google.com/p/bnt/ (1997–2002). Accessed April 2014

Chapter 10
Pin-Count and Wire Length Optimization for Electrowetting-on-Dielectric Chips: A Metaheuristics-Based Routing Algorithm

Mohamed Ibrahim, Cherif Salama, M. Watheq El-Kharashi and Ayman Wahba

Abstract Electrowetting-on-dielectric chips are gaining momentum as efficient alternatives to conventional biochemical laboratories due to their flexibility and low power consumption. In this chapter, we present a novel two-stage metaheuristic algorithm to optimize electrode interconnect routing for pin-constrained chips. The first stage models channel routing as a traveling salesman problem and solves it using the ant colony optimization algorithm. The second stage provides detailed wire routes over a grid model. The algorithm is benchmarked over a set of real-life chip specifications. On average, comparing our results to previous work, we obtain reductions of approximately 39 % and 35 % on pin-count and total wire length, respectively.

10.1 Introduction

Digital Microfluidic Biochip (DMFB) technology has come into sight as an efficient alternative for the conventional biochemical laboratory procedures [9]. It provides a miniaturized platform for developing a wide range of automated diagnostic applications, such as DNA sequencing, environmental monitoring, and point-of-care diagnosis of diseases [19]. Many advantages like low cost, low sample and reactant consumption, and immunity to human errors have led DMFB to gain momentum in recent years. The global market value for biochip products has exceeded $3 billion and is expected to increase to over $9 billion in 2016 [4]. However, this continuing growth of various applications have dramatically complicated the chip design complexity. Typical requirements now involve multiple and concurrent assays on the same chip,

M. Ibrahim · C. Salama (✉) · M.W. El-Kharashi · A. Wahba
Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt
e-mail: cherif.salama@eng.asu.edu.eg

M. Ibrahim
e-mail: m.ibrahim@eng.asu.edu.eg

M.W. El-Kharashi
e-mail: watheq.elkharashi@eng.asu.edu.eg

A. Wahba
e-mail: ayman.wahba@eng.asu.edu.eg

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_10

as well as more sophisticated control for resource management. For example, DNA sequencing analysis involves millions of base pairs and probe combination, with each combination corresponding to a unique experimental determination [3]. It is hard to depend on human efforts alone for efficient DNA sequencing analysis. In addition, time to market and fault tolerance are also expected to emerge as design considerations. Moreover, it is expected that DMFBs will be integrated with microelectronic components in next-generation System-on-a-Chip (SoC) designs. The International Technology Roadmap for Semiconductors (ITRS) has clearly identified the integration of electrochemical and electrobiological techniques as one of the system-level design challenges that have been faced beyond 2009 in chips with feature sizes below 50 nm [15]. Considering the above discussions, it is necessary to develop high-quality CAD tools for efficient DMFB design automation which, in turn, is also expected to facilitate the integration of fluidic components with microelectronic components in next-generation SoCs [6].

The most popular subclass of DMFB chips depends on a principle known as ElectroWetting-On-Dielectric (EWOD) and are consequently known as EWOD chips [12, 18]. As presented in Fig. 10.1, the EWOD chip typically consists of a 2D array of cells through which the discrete chemical droplets can be *digitally* manipulated. A unit cell includes a pair of electrodes forming two parallel plates. The electrodes are derived through pins using an external microcontroller device forming a time-varying voltage actuation sequence for each cell. In this regard, droplets are controlled due to the electrowetting phenomenon executing the intended bioassay operations [17]. Consequently, early *fluidic-level* design automations [7, 20, 24] have been proposed for droplets manipulation starting with the sequencing graph of the biochemical procedure and ending with the individual voltage actuation sequence for each cell, considering an independent control pin for each electrode. This early *electrode addressing* is known as the *direct-addressing* [11].

Although *direct-addressing* gives maximum droplet manipulation freedom, it requires a large pin-count (and accordingly a large number of microcontroller ports), which could be infeasible for practical biochips. On the other hand, *broadcast-addressing* [21], a pin-constrained design approach grouping electrodes that have mutually compatible actuation sequences by connecting them to a common pin, suffers from the wiring problem arising from chip complexity.

To date, large efforts are dedicated to *broadcast-addressing*. When selecting a broadcast-addressing solution, there is a trade-off between wiring complexity and the number of pins allocated. Hence, as in *chip-level design automation*, an integrated design considering broadcast-addressing requirements (pin-count) and wire routing constraints simultaneously is expected to achieve higher routability results.

10.1.1 Previous Work

Compared with fluidic-level synthesis, chip-level design is a relatively young research field in the computer-aided design (CAD) methodology of EWODs. The most crucial problem in chip-level design is the wire routing problem, in which electrical wires should be routed among electrodes to establish the signal connections and transmission. Earlier design methods are mostly based on manual efforts, i.e., experienced designers draw the layout and wiring connections by hand. However, the high design complexity potentially makes the traditional manual design manner inefficient [12, 13]. This complexity motivated the development of the first automatic wire routing algorithm [14]. Given that wire routing was proved to be NP-complete [1], it is a fairly challenging problem to solve, specially when the growing EWOD chip complexity is taken into account.

Huang et al. were the first to combine both wire routing and broadcast-addressing in chip-level routing [14]. Their work followed a sequential approach using a two-stage technique. Pin-count-aware global routing is presented in the first stage to reduce the complexity of routing to be through global tracks, instead of a complete 2D array. Progressive routing follows in the second stage to route unaddressed electrodes. Their work mainly depends on formulating the stages into maximum-flow and minimum-cost, maximum-flow networks. However, this sequential mechanism suffers from a strong dependence on the net ordering. This can cause some nets to unnecessarily block others. In this chapter, we propose a predicable approach benefiting of metaheuristics to avoid this problem. We generate a set of routing solutions such that the independent routing solution with the minimum cost is prioritized.

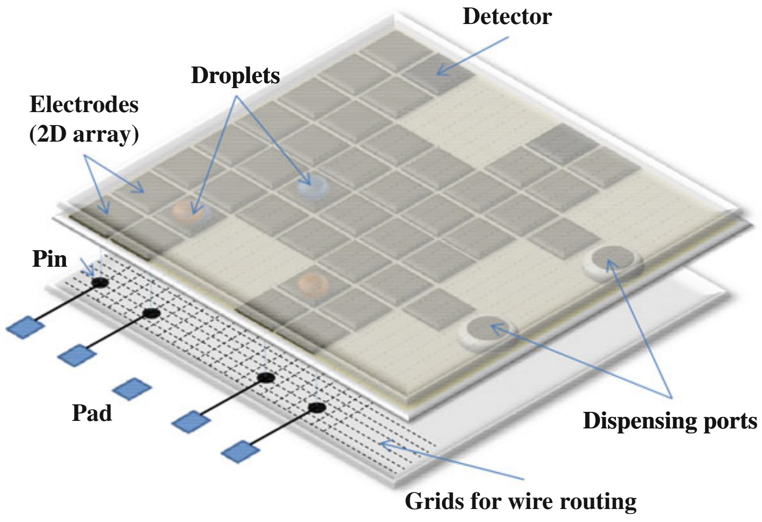


Fig. 10.1 Schematic view of an EWOD chip (adapted from [14])

10.1.2 Problem Statement

We can divide the problem statement, in the context of EWOD chip level routing, into two parts. The first one is concerned with addressing the electrodes with control pins, considering pin-count reduction, whereas the other one is concerned with efficient electrical wiring along the chip grid. Although the two parts are separately handled in the context, it is necessary to develop an integrated chip-level design automation that considers both simultaneously. So, the problem statement is:

Given the individual control information for the chip electrodes from the fluidic-level synthesis, we want to get a chip design featuring minimum pin-count and minimum wire length.

10.1.3 Contributions

Throughout this chapter, we introduce a novel approach for solving the broadcast-addressing EWOD chip-level routing. Our contributions can be listed as follows:

- *Two-stage interconnect routing:* We propose a two-stage algorithm. First, the *Channel Routing* stage is a congestion-aware metaheuristic routing scheme used to highlight the preferred channel paths for each net. The *Detailed Routing* stage comes next to provide the specific routes for nets.
- *Metaheuristic solution for channel routing:* Solutions of channel paths are mapped to a variation of the traveling salesman problem (TSP) [10], such that a complete routing of all nets can be represented as a complete journey by the salesman. This problem model can be solved using the Ant Colony Algorithm (ACO) and accordingly a near-optimal solution is obtained. The ACO algorithm of Marco Dorigo simulates the ants behavior for solving optimization problems [8]. Artificial ants are thrown in the search-space and allowed to move freely searching for minimizing a cost function. ACO is used due to its flexibility, high performance, and ability to be used for parallel programming on mutli-core frameworks.
- *Pin-count reduction:* When *broadcast-addressing* is used, a single electrode is often compatible with more than one set of mutually compatible electrodes, giving rise to multiple electrode grouping solutions. We introduce a novel algorithm, baptized *Adapted Tracks*, to resolve this ambiguity, while minimizing the number of pins required by broadcast-addressing.

The remaining of this chapter is organized as follows. Section 10.2 introduces related preliminaries including broadcast-addressing and the routing model for the pin-constrained design. Section 10.3 formulates the problem, whereas Sect. 10.4 details the proposed broadcast-addressing chip-level routing algorithm. Section 10.5

presents the benchmarking results. Finally, Sect. 10.6 discusses the conclusions and future work.

10.2 Preliminaries

This section introduces the concept of broadcast-addressing as a pin-constrained design approach. It also introduces the EWOD chip-level routing model we adopt.

10.2.1 Electrode Broadcast-Addressing

As discussed in Sect. 10.1, a primary issue in performing various fluidic-level handling functions through EWOD chips is the manipulation of droplets. An EWOD chip generates electric potential by actuating electrodes to change the wettability of droplets, such that droplets can be shaped and driven along the active electrodes. To induce enough wettability change for droplet motion, the voltage value applied to electrodes must exceed a threshold. This phenomenon enables a binary bit (i.e., ‘1’/‘0’) to represent the status of an actuation voltage. Typically, control signal of moving a droplet in a specific time step can be represented as activated bit ‘1,’ deactivated bit ‘0,’ or do not care ‘X.’ The bit ‘1’/‘0’ represents a control signal with a relative logic-high/logic-low value of the actuation voltage. The symbol ‘X’ indicates that the input signal can be either ‘1’ or ‘0,’ which has no impact on scheduled fluidic controls. Since droplets are controlled in a time-multiplexed manner, controlling information of electrodes can be obtained by concatenating these bits and symbols time-step by time-step. The concatenated outcome is called electrode *activation sequence*. Examples of an electrode layout and the corresponding activation sequences are presented in Fig. 10.2a, b.

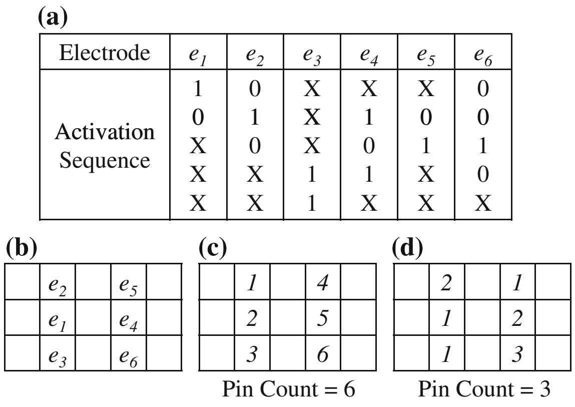


Fig. 10.2 Example on electrode layout, activation sequences, and broadcast-addressing. **a** Scheduled fluidic functions in the form of activation sequences. **b** Electrodes used for handling fluidic functions. **c** Pin-count and assignment in case of direct-addressing scheme. **d** Pin-count and assignment in case of broadcast-addressing scheme

(a)

Electrode	e_1	e_2	e_3	e_4	e_5	e_6
Activation Sequence	1	0	X	X	X	0
	0	1	X	1	0	0
	X	0	X	0	1	1
	X	X	1	1	X	0
	X	X	1	X	X	X

(b)

	e_2		e_5	
	e_1		e_4	
	e_3		e_6	

(c)

	1		*4*	
	2		*5*	
	3		*6*	

Pin Count = 6

(d)

	2		*1*	
	1		*2*	
	1		*3*	

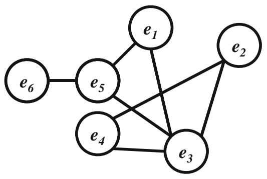
Pin Count = 3

Typically, broadcast-addressing focuses on electrode grouping and control signal merging through the compatibility of activation sequences. Specifically, each electrode activation sequence may contain several do not care terms. By carefully replacing these do not care terms with ‘1’ or ‘0,’ multiple activation sequences can be merged to an identical outcome, which is also referred to as the *common compatible* sequence of these electrodes. Therefore, these electrodes can be assigned by the same control pin to receive the same control signal. Take electrodes e_5 and e_6 in Fig. 10.2a, for example, by replacing ‘X’ in the activation sequence of e_5 with ‘0,’ we can merge the activation sequences of e_5 and e_6 to “0010X.” Therefore, e_5 and e_6 can be addressed with the same control pin due to their mutually compatible activation sequences. Figure 10.2c, d compare the direct-addressing and broadcast-addressing outcomes. The numbers shown in these figures represent pin identifiers that have been assigned to each of the corresponding electrodes in Fig. 10.2b. Compared with the direct-addressing result in Fig. 10.2c, the broadcast-addressing result in Fig. 10.2d significantly reduces the required control pins from 6 to 3. This reduction results in fewer electrical devices and connections to perform the same fluidic functions, thus improving chip reliability and saving on fabrication cost. Therefore, the derivation of a correct electrode-addressing result under the pin constraint issue is of great importance, especially in the DMFB marketplace.

To implement broadcast-addressing, electrode grouping is introduced such that for all electrodes in any group, the corresponding activation sequences are mutually compatible. Toward this goal, a compatibility graph is constructed, where each vertex represents an electrode and an edge between two electrodes indicates that their corresponding activation sequences are compatible. For example, Fig. 10.3a demonstrates a compatibility graph G_c derived from Fig. 10.2a. Based on the compatibility graph, the electrode grouping can be mapped to the clique partition problem, which is a well-known problem in graph theory. Since each clique represents an electrode group with mutually compatible control signals, we can individually assign each clique with a dedicated control pin. Accordingly, by recognizing maximal electrode cliques in the compatibility graph, the required number of control pins can be minimized.

Reporting the maximal electrode cliques of a graph is a fundamental problem arising in many areas. In our design, the Bron–Kerbosch algorithm, shortly known as the BK algorithm, is applied to get the maximal electrode cliques of mutually compatible electrodes within the undirected compatibility graph [2]. It is known to be one of the most efficient algorithms that uses recursive backtracking

(a)



(b)

Two electrode grouping results

Pin	*Result 1*	*Result 2*
1	e_2, e_3, e_4	e_1, e_3, e_5
2	e_1	e_2, e_4
3	e_5, e_6	e_6

Fig. 10.3 **a** A compatibility graph G_c derived from Fig. 10.2a. **b** Two possible electrode grouping results

Algorithm 1: The recursive BK_Enumerate algorithm

```
   Input: Compatibility Graph $G$, Vertices Sets $P$, $R$, and $X$
   Output: Maximal Electrode Cliques $R_{new}$ of graph $G$
   /* $N[u_i]$ represents the neighbors of vertex $u_i$ */
 1 begin
 2     if $P=\phi$ and $X=\phi$ then
 3         print $R$ as maximal electrode clique ;
 4     else
 5         $u_p \leftarrow$ pivot vertex in $P \cup X$;
 6         Assume $P = \{u_1; u_2; ...; u_k\}$;
 7         for $i \leftarrow 1$ to $k$ do
 8             if $u_i$ is not a neighbor of $u_p$ then
 9                 $P = P - \{u_i\}$;
10                 $R_{new} = R \cup \{u_i\}$ ;
11                 $P_{new} = P \cap N[u_i]$ ;
12                 $X_{new} = X \cap N[u_i]$ ;
13                 BK_Enumerate ($G$, $P_{new}$, $R_{new}$, $X_{new}$) ;
14                 $X = X \cup \{u_i\}$ ;
15             end
16         end
17     end
18 end
```

BK_Enumerate (G,P, R, X)	**Pivot**	**Reported clique**
BK_Enumerate (G,[1,2,3,4,5,6],ϕ,ϕ)	3	
BK_Enumerate (G,[1,2, 4,5],[3],ϕ)	1	
BK_Enumerate (G,[5],[3,1], ϕ)	5	
BK_Enumerate (G,ϕ,[3,1,5], ϕ)		**(3,1,5)**
BK_Enumerate (G,[4],[3,2],ϕ)	4	
BK_Enumerate (G,ϕ, [3,2,4],ϕ)		**(3,2,4)**
BK_Enumerate (G,ϕ,[3,4],2)	2	
BK_Enumerate (G,[5], [6], ϕ)	5	
BK_Enumerate (G,ϕ, [6,5],ϕ)		**(6,5)**

Fig. 10.4 Outcome of applying BK algorithm on the compatibility graph G_c in Fig. 10.3

to find maximal electrode cliques [5]. The core of the BK algorithm is the recursive *BK_Enumerate*(G, P, R, X) function shown in Algorithm 1. G is the compatibility graph, while P, X, and R are sets of vertices from G. P is initialized to include all the vertices of G, while X and R are initialized to the empty set ϕ. Based on the observations raised in [5], which aim at reducing the size of the recursion tree of the BK algorithm, a pivot selection criterion is applied at Line 5 of Algorithm 1. Experimentally, it consists of choosing, as pivot, the vertex with the largest degree in P.

When applying the BK algorithm on the example illustrated in Fig. 10.3, we get three maximal electrode cliques; {3,1,5}, {3,2,4}, and {6,5}, as illustrated in Fig. 10.4. Note that electrode 3 is included in two cliques, accordingly a certain criterion should be settled for selection, as discussed in subsequent subsections.

10.2.2 EWOD Chip-Level Routing

In order to address a group of mutually compatible electrodes with the same pad, wires must be appropriately routed, connecting the corresponding control pins and escaping these connections into a pad at the chip boundary.

For cost-effective EWOD chip-level design, we follow the state-of-the-art routing model in [14], where the PCB fabrication process is used. Only horizontal and vertical wires are permitted through one routing layer only. In addition, the capacity of the channel passing between two adjacent pins is a maximum of three wires.

10.2.3 Channel Network Flow Model

Since routing on the grid model directly is computationally expensive, we perform routing on a channel network flow model instead. In our proposed algorithm, we define the *channel* to be any track passing between two adjacent pins. So, a bidirectional graph $G = (V, E)$ is used to model channels in a grid array. Each vertex $v \in V$ represents an intersection point between two orthogonal channels, and the edge (v_i, v_j) represents the channel path itself, as shown in Fig. 10.5. All edges are with 3 *units* of capacity.

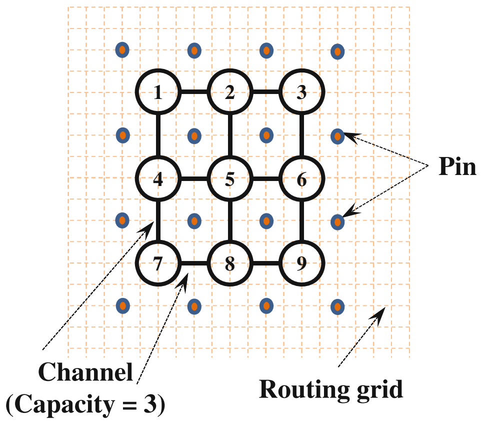
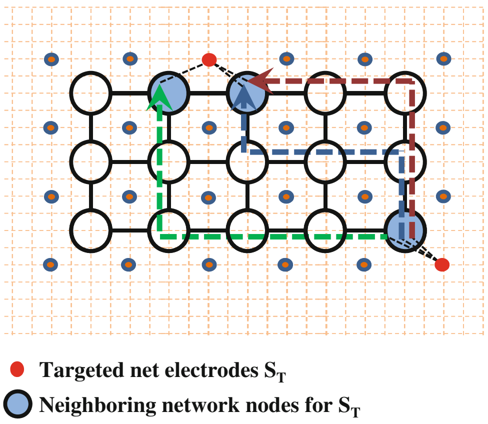


Fig. 10.5 The channel network flow model for the proposed EWOD routing



● Targeted net electrodes S_T
● Neighboring network nodes for S_T

Fig. 10.6 Generating a set of possible channel paths using Yen's Algorithm

10.2.4 *Yen's K-Shortest Channel Paths*

EWOD chip routing is an optimization problem which contains a set of parameters seeking a global optimum. So, the shortest path connecting two compatible control pins is not always the best solution due to the wire-crossing that may happen with other nets. In addition, a sequential routing of nets as in the work by Huang et al. [14] causes some routed nets to block other unrouted ones. So, a widely used approach in VLSI is concurrent routing where a number of possible paths are generated for each net. A decision for a routing solution is then to be taken based on a cost function, as discussed in Sect. 10.4.

Effectively, in the channel routing, not only the shortest path should be explored, but also the K-shortest paths for each net. In addition, increasing K leads to a wider search-space and accordingly a minimum cost function. Yen's algorithm [23] solves K-shortest channel paths, as illustrated in Fig. 10.6. Note that a multiterminal net is considered as a set of two-terminal nets. For example, in a 3-terminal net, the K-shortest paths are generated between two terminals. Then, the nodes of each generated path is combined into a super node from which other K-shortest paths toward the third terminal are generated. Consequently, we generate K^{n-1} shortest channel paths for any n-terminal net.

10.3 Problem Formulation

This chapter handles the EWOD pin-count aware routing based on the following formulation:

Inputs:

- *Chip specifications: Chip size, P_{max}, chip electrode set including the location of each electrode*

- *Electrode actuation sequences*

Constraints: *Broadcast constraints based on the routing model*
Objective: *An interconnect routing solution, minimizing both pin-count and wire length*

Method: *A metaheuristic approach*

10.4 Algorithm

In this section, we present our proposed metaheurisitc routing algorithm.

Algorithm 2 introduces the top view of our routing algorithm with its inputs and outputs matching our problem formulation in Sect. 10.3. Lines 2–4 construct the electrodes compatibility graph. The BK algorithm is then used to derive the maximal electrode *cliques* of compatible electrodes in Line 5. However, in some cases, the maximal electrode cliques solution cannot be directly used as the basis for broadcast-addressing because a single electrode can be a member of multiple cliques. In these cases, there are multiple choices regarding which clique this electrode should be connected to. To illustrate this situation, consider the *protein* biochip [14] in Fig. 10.7. The protein biochip is a real-life application chip that we used to benchmark our algorithm. In this biochip, electrode 18 is included in both cliques {18, 19, 20} and {18, 30}. Huang et al. in [14] proposes a "Global Tracks"-based solution through which broadcast-addressing is performed. The resulting solution is shown in Fig. 10.7a. We propose an alternative method that we name '*Adapted Tracks*,' relaxing the global tracks requirement. The resulting solution, shown in Fig. 10.7b, significantly reduces the number of control pins. In our routing algorithm, we invoke the adapted tracks algorithm in Line 6. The remaining part of Algorithm 2, Lines 7–13, invokes the metaheuristic channel routing algorithm followed by the detailed routing algorithm until an acceptable solution is found. The adapted tracks algorithm, the metaheuristic channel routing algorithm, and the detailed routing algorithm are presented in the following subsections.

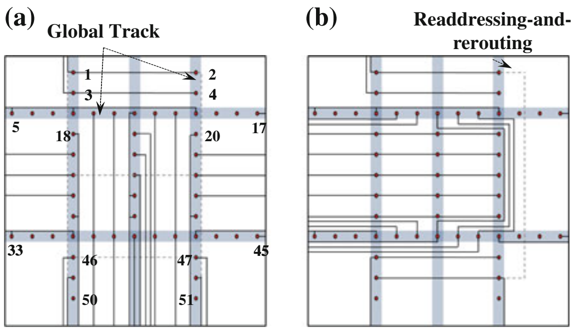


Fig. 10.7 Track selection for broadcast-addressing and the routing result in the protein chip [14]. **a** Considering the Global Tracks: the router uses 27 control pin. **b** Considering the Adapted Tracks: the router uses 21 control pin. (Unaddressed electrodes are "don't cares".)

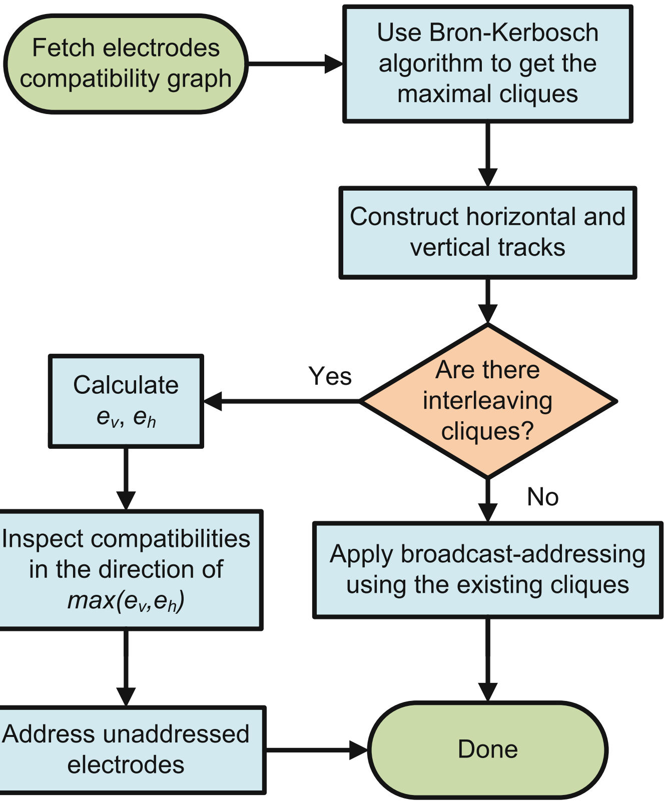

Fetch electrodes compatibility graph
Use Bron-Kerbosch algorithm to get the maximal cliques
Construct horizontal and vertical tracks
Are there interleaving cliques?
Yes
No
Calculate e_v, e_h
Inspect compatibilities in the direction of $max(e_v, e_h)$
Apply broadcast-addressing using the existing cliques
Address unaddressed electrodes
Done

Fig. 10.8 Flowchart of broadcast-addressing using adapted tracks

10.4.1 Adapted Tracks Algorithm

Figure 10.8 shows the flow of the adapted tracks as a post-processing stage after maximal electrode cliques derivation. The adapted tracks method starts with constructing all the horizontal and vertical tracks combining all electrodes. If there is at least one electrode existing in more than one clique (overlapping cliques), the algorithm should go through these tracks to inspect compatibilities in both directions. The orientation vectors $\mathbf{e_h}$ and $\mathbf{e_v}$ are then determined such that $|\mathbf{e_i}| = \sum_j E(C_{ij})$, where the symbol $E(C_{ij})$ is the number of electrodes in the clique C_j located in the direction i. The magnitudes of these vectors are used to highlight the higher priority direction through which the broadcast-addressing is performed for the interleaving cliques. By considering the adapted tracks in the *protein* chip, we find that $|\mathbf{e_h}| = 28$ whereas $|\mathbf{e_v}| = 18$. Since $|\mathbf{e_h}|$ has a larger value, broadcast-addressing in the horizontal direction is given a higher priority. Figure 10.7b shows the effectiveness of the used adapted tracks method, as it needs 21 pins instead of 27.

Algorithm 2: Top view of the proposed two-stage routing algorithm.

```
Input: Chip size, $P_{max}$, and the electrode set $S_e$ location and control information
Output: A wire routing solution $Sol$, minimizing pin-count and wirelength and the
        corresponding pin assignment $P_{used}$
/* $P_{used}$: Pin assignment for the derived solution          */
/* $CP$: Channel preferred paths set for all nets                */
1  begin
2      read chip specs and $S_e$ control information;
3      construct channel network model $N$;
4      derive compatibility graph $G_c$ for $S_e$;
5      construct maximal electrode cliques of $G_c$;
6      apply adapted tracks for broadcast-addressing;
7      $P_{used} \Leftarrow \phi$; $Sol \Leftarrow \phi$;
8      while $Sol = \phi$ or $|P_{used}| > P_{max}$ do
9          $CP \Leftarrow$ metaheuristic_channel_routing($N$);
10         $Sol$, $P_{used} \Leftarrow$ detailed_routing($CP$);
11     end
12     $Result \Leftarrow \{Sol, P_{used}\}$;
13     return $Result$;
14 end
```

Some electrodes may be left unaddressed due to the arrangement of the tracks. We first need to connect these electrodes to one of the existing pins for minimum pin-count expansion. If there is no compatible pins, we connect the electrodes to dedicated pins. However, in the case where there are many pins that can be assigned to an unaddressed electrode, we select the nearest compatible electrode for the purpose of wire length minimization.

10.4.2 Channel Routing

To overcome the routing problem, we initially transform it into congestion-aware channel routing. So, we adapt a metaheuristic approach to obtain the preferred channel paths for each net. Algorithm 3 details the channel routing. The inputs to this algorithm are the maximal electrode cliques based on the adapted tracks and the electrode locations. Lines 2–5 generate the shortest channel paths for every clique. Lines 6 and 7 model the channel routing problem as a TSP and invoke the ACO algorithm to solve it. The TSP formulation and the ACO solution are explained below.

Algorithm 3: Channel routing of maximal electrode cliques.

Input: A stack of maximal electrode cliques (EC) and the electrode locations
Output: Channel preferred paths CP

```
/* $CG_i$: Net $i$ channel generated paths */
1 begin
2     while $EC \neq \phi$ do
3         $EC_i \Leftarrow EC$.POP();
4         $CG_i \Leftarrow$ K_shortest_paths($EC_i$);
5     end
6     generate $E$-$GTSP$ model $M$ for all $CG_i$;
7     $CP \Leftarrow$ ACO($M$);
8     return $CP$;
9 end
```

10.4.2.1 Traveling Salesman Problem (TSP) Formulation

As explained in Algorithm 3, we obtain multiple channel paths for each clique. A global solution requires us to pick the best combination of paths (i.e., a channel path for each clique). The problem of finding such solution can be modeled as a search-space problem. In particular, we model each channel path as a city, each group of channel paths of a single clique as a region, and the search problem as an Equality-Generalized TSP (E-GTSP) [10]. The E-GTSP is a version of the classical TSP that requires the traveling salesman to visit each region exactly once.

In order to select a meaningful cost function, we classify edges of a channel path into *non-congested*, *congested*, and *crossing* edges, as illustrated in Fig. 10.9. The counts of these edges are given the symbols e_n, e_c, and e_x, respectively. We propose the following cost function combining these lengths with different weights:

$$Cost = \sum_{j=1}^{m} \left[\alpha . \sum e_n + \beta . \sum e_c + \gamma . \sum e_x \right] \tag{10.1}$$

In our tests, the weights α, β, and γ are given the values 10, 15, and 150, respectively.

Figure 10.9 shows two different configurations for the channel routing solutions for the *amino-acid* chip [14]. Like the protein biochip, the amino-acid biochip is a real-life application chip that we used to benchmark our algorithm. Both solutions are the result of two different journeys for the traveling salesman among the 7 regions forming the cliques. When computing the cost function on Fig. 10.9a, we found 30 non-congested, 6 congested, and 4 crossing edges. So, the cost is $10(30) + 15(6) + 150(4) = 990$. On the other hand, the solution in Fig. 10.9b contains 39 non-congested and 6 congested edges only. Therefore, its cost is

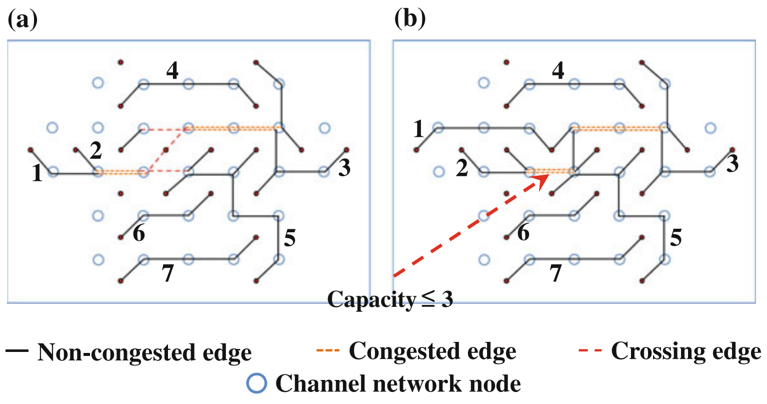


Fig. 10.9 Channel routing solutions for the amino-acid chip [14]. **a** Solution with non-congested, congested, and crossing edges having a total cost of 990. **b** A better solution without crossing edges having a total cost of 480

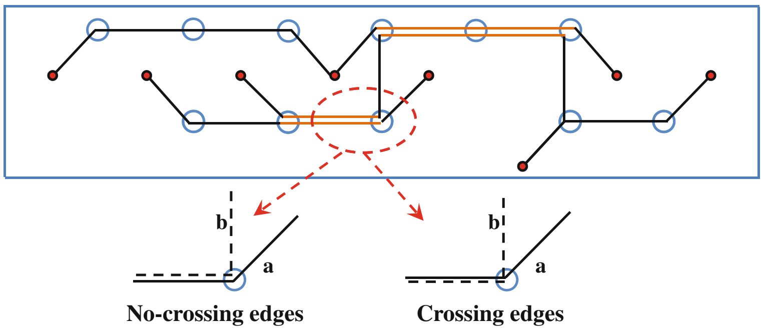


Fig. 10.10 Nets ordering in the edges according to their spanning angles

$10(39) + 15(6) = 480$. Note that counting the crossing edges during a journey must be done carefully. For instance, the highlighted edge in Fig. 10.10 might be counted as crossing unless a way of appropriately ordering the nets is provided. Net *a* is said to contain net *b* if the spanning angle of *a* is larger, as shown in Fig. 10.10. We use a *net orientation* matrix to store the ordering of the nets based on containment.

Many heuristics can be used to solve the E-GTSP problem, we choose to adapt the ACO algorithm for this purpose, as explained below.

10.4.2.2 ACO Method

In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but instead they are likely to follow the trail, returning, and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. The ACO algorithm can be adapted to solve the proposed E-GTSP model. Algorithm 4 explains the details of the adapted ACO algorithm.

The algorithm takes the E-GTSP model as its input search-space through which the ants are thrown. Each ant is allowed to orderly visit each region only once. So, an ant at region 1 will move toward a city within region 2 and so on. Line 2 of Algorithm 4 initializes the pheromone level at all edges with a constant value τ_0. In our work, we select τ_0 to be 100. Then, in Line 5, the ants are randomly thrown in the search-space such that each ant keeps its initially visited city at its *Tabu List*, which is an ant-specific vector that stores the cities already visited up to time t. Each ant is then allowed to "step" toward the neighboring region, according to Lines 9–12, such that the probability that an ant at city i will move to city j is calculated using the following equation:

$$P_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}; \; j \in k \tag{10.2}$$

where k is the list of cities located in the new region, $\tau_{i,j}^{\alpha}$ is the amount of pheromone on the path (i, j), $\eta_{i,j}^{\beta}$ is the desirability of the path (i, j), which equals the reciprocal of the evaluated cost function as defined by the E-GTSP. It should be noted that α and β are user-defined parameters. In our tests, they are given the values 1 and 5, respectively.

As in genetic algorithms, in order to prevent being stuck at a local minimum, sometimes it is useful to increase the likelihood of selecting a city with a lower probability since it may contribute later in the process of choosing a good candidate solution. This is performed using the roulette wheel selection function in Line 13.

Algorithm 4: Adapted ACO for the E-GTSP model.

```
Input: Solutions of the nets modeled as E-GTSP
Output: Minimum-cost channel routing solution $OptSol$
/* step: counter for visited cities by each ant      */
/* m: number of regions                              */
/* R: total number of cities                         */
begin
    initialize parameters & pheromone;
    while termination condition is not met do
        empty ants memory;
        place ants randomly at all cities in the $m$ regions;
        $step = 2$;
        $OptLength = \infty$;
        $OptSol = \phi$;
        while $step \leq m$ do
            foreach ant of R ants do
                calculate ant step distance;
                $p \Leftarrow$ calculate $P_{step-1,step}$;
                $nextnode \Leftarrow$ roulette($p$);
                refresh Tabu List of the ant;
            end
        end
        close tour;
        $ToursLen \Leftarrow$ calculate tour lengths for all ants;
        $MinLength \Leftarrow \min(ToursLen)$;
        refresh pheromone;
        if $MinLength < OptLength$ then
            $OptLength \Leftarrow MinLength$;
            update $OptSol$ ;
        end
    end
    return $OptSol$;
end
```

At the end of the journey, each ant declares its *TourLength* (i.e., the journey cost value) in order to select the best candidate solution for this run, as illustrated in Lines 17–19. Finally, the pheromone refresh process per ant is performed at the visited paths, as shown in Line 20. The process can be characterized by the following equations:

$$\Delta\tau_{i,j} = \begin{cases} \frac{Q}{TourLength} & (i, j) \in TabuList_{ant} \\ 0 & Otherwise \end{cases} \tag{10.3}$$

$$\tau_{i,j}(t) = \rho.\tau_{i,j}(t-1) + \Delta\tau_{i,j} \tag{10.4}$$

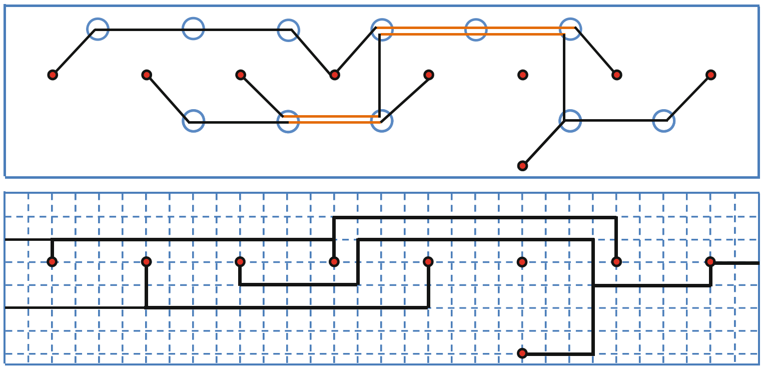


Fig. 10.11 Detailed routing for a set of nets in amino-acid

where $\Delta\tau_{i,j}$ is the amount of pheromone deposited by ants visiting path (i, j), ρ is the evaporation rate which is set to be 0.35 in our case, and Q is a constant set to 100.

Finally, as shown in Line 3, the algorithm procedure is repeated until either reaching saturation or exceeding the maximum number of cycles (750 in our case).

10.4.3 Detailed Routing

A channel routing solution from the ACO is used in detailed routing. According to the grid routing model, we are maximally able to route three wires between two adjacent pins. So, we incrementally pick up each net to be routed horizontally and vertically through the grids. Also, it is escaped to the chip boundaries. Typically, routing through a channel depends on the nets ordering depicted in the *net orientation* matrices. Figure 10.11 illustrates the idea behind detailed routing after considering the net ordering criterion from Fig. 10.10.

In some cases, as in Fig. 10.7b, routing or escaping creates crossing nets. A simple workaround for this issue is to readdress and reroute the blocking net to have a complete chip-level routing solution.

10.5 Benchmark Results

The proposed routing algorithm has been tested on a 2.4 GHz, 64-bit Intel Core i5 computer. The evaluation of the required objectives is performed on a set of real-life EWOD chip benchmark applications [14], including two amino-acid synthesis benchmarks (*AC-1* and *AC-2*), two protein synthesis benchmarks (*Pro-1* and *Pro-2*), and one multiplexed assay benchmark (*Mux*). Figure 10.12 shows the electrodes

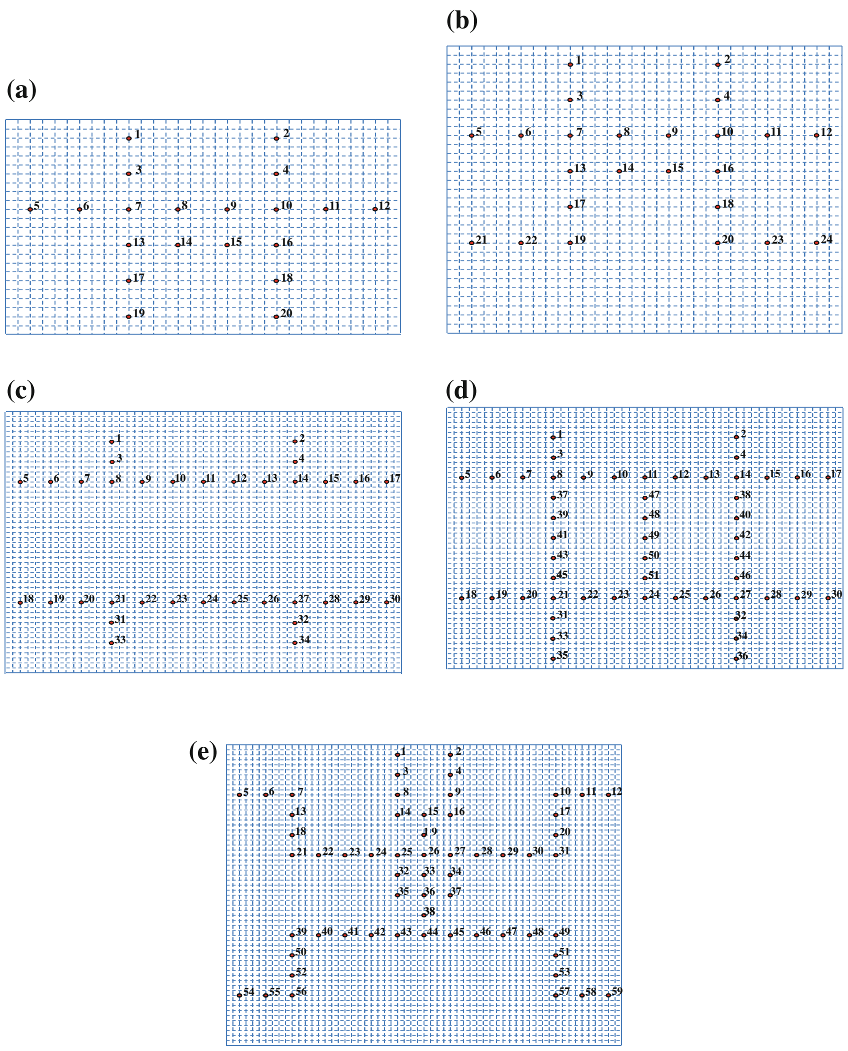


Fig. 10.12 Chip layouts for benchmark applications [14]. **a** Amino-acid-1 (AC-1). **b** Amino-acid-2 (AC-2). **c** Protein-1 (PRO-1). **d** Protein-2 (PRO-2). **e** Multiplexed assay (MUX)

distribution and chip layouts for the used benchmarks, whereas Fig. 10.13 shows their compatibility graphs. Note that the input to our algorithm is the control information of the electrodes, represented as activation sequences. However, for the reader's convenience, the compatibility graphs of the electrodes are shown instead. Also note that excluding an electrode from the compatibility graph, like electrode 1 in the amino-acid-1 chip, indicates a full do not care sequence for this electrode. Thus, any

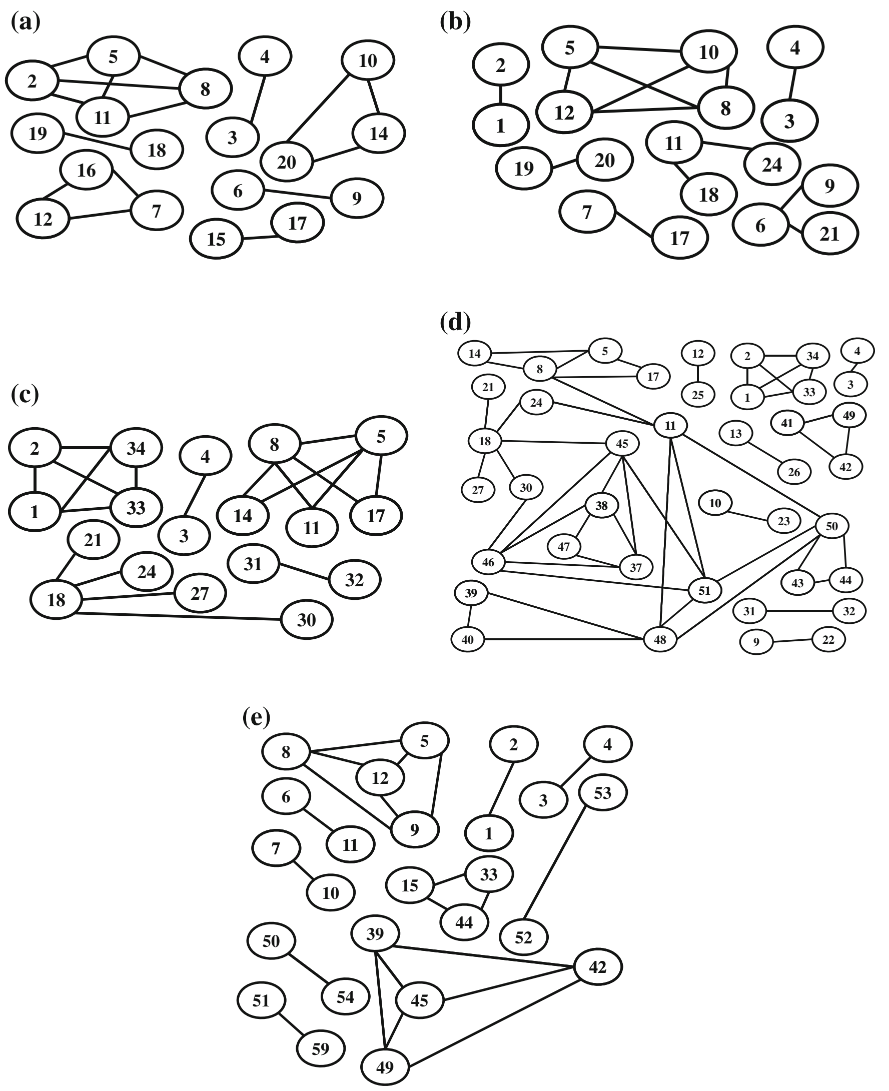


Fig. 10.13 Compatibility graphs for benchmark applications [14]. **a** Amino-acid-1 (AC-1). **b** Amino-acid-2 (AC-2). **c** Protein-1 (PRO-1). **d** Protein-2 (PRO-2). **e** Multiplexed assay (MUX)

excluded electrode is explicitly understood as an electrode that can be connected to any pin.

Table 10.1 lists the overall comparison results in terms of pin-count and wire length for all the real-life chip applications listed earlier in addition to a randomly-generated chip layout and activation sequences. The P_{max} for the amino-acid chips is 16 and 32 for the others. The table demonstrates the effectiveness of our algorithm compared to

Table 10.1 Electrode addressing and wire routing comparisons between DA [11], BA [21], EBA [14], and ours

Chip	#E	Size	Number of pins					Total wire length				
			DA	BA	EBA	Ours	PR (%)	DA	BA	EBA	Ours	WM (%)
AC-1	20	6 × 8	20	13	9	7	22.2	156	254	190	179	5.79
AC-2	24	8 × 8	24	16	11	9	18.2	168	236	207	175	15.46
Pro-1	34	13 × 13	–	19	24	11	54.2	–	814	462	267	42.21
Pro-2	51	13 × 13	–	21	25	21	16.0	–	898	662	724	–9.37
Mux	59	15 × 15	–	–	36	10	72.2	–	–	1444	502	65.24
Rand	20	10 × 10	20	14	8	11	–37.5	281	353	278	251	9.71
Total	–	–	–	–	113	69	38.9	–	–	3243	2098	35.31

#E refers to the number of electrodes of the chip. *PR%* refers to the percentage of pin-count reduction of ours with respect to EBA, whereas the *WM%* refers to the percentage of wire length minimization of ours with respect to EBA

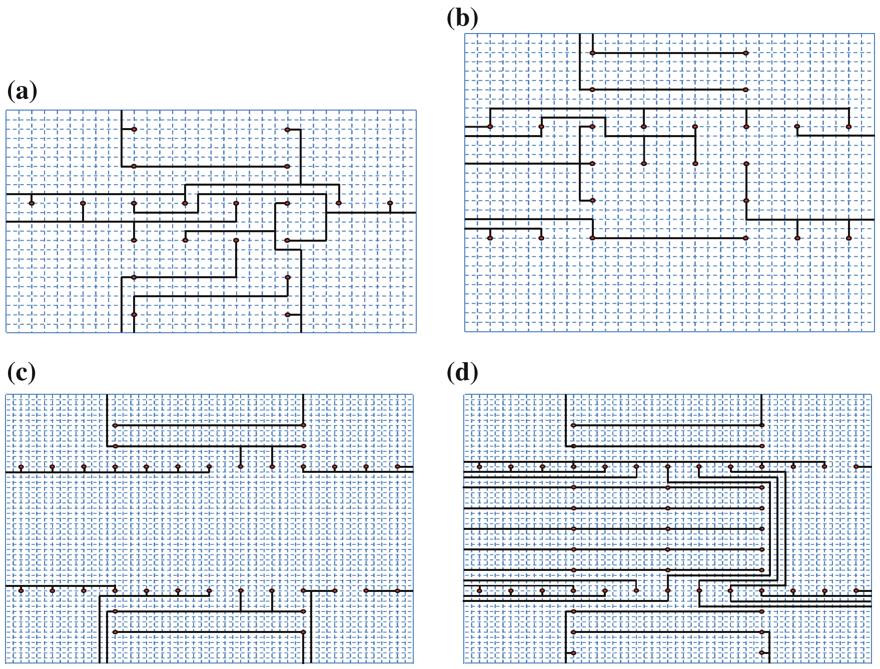


Fig. 10.14 Chip-level routing solutions for benchmark chips. **a** Amino-acid-1 (AC-1). **b** Amino-acid-2 (AC-2). **c** Protein-1 (PRO-1). **d** Protein-2 (PRO-2)

direct-addressing (DA), broadcast-addressing (BA), and the enhanced BA algorithm proposed in [14] (EBA). On the average case, we obtain reductions of 38.9 % and 35.3 % on pin-count and total wire length, respectively with respect to EBA. Due to the nature of our algorithm, it produces solutions with less control pins and shorter conduction wiring in general. In addition, we achieve large pin-count reductions as well as wire length minimizations in *Pro-1* and *Mux* since the electrodes with full *do not cares* sequences can be connected to any pin. However, in *Pro-2* chip case, ours produces longer conduction wires due to the wiring detour caused by the pin-count reduction objective. Finally, the randomly-generated (*Rand*) chip consumes more control pins in our solution due to the randomness in the distribution of electrodes compatibility, which might not be very realistic. This test case was only included to demonstrate the completeness of our solution.

Figure 10.14 shows the generated routing solutions for the amino-acid and protein synthesis EWOD chips.

In our analysis, we did not consider CPU time since heuristics are known to consume more time with a wider search-space. However, a wider search-space causes our router to converge to optimality.

10.6 Conclusion and Future Work

In this chapter, we introduced a novel approach for pin-count-aware chip-level routing of EWOD chips. Using the cliques of mutually compatible electrodes, many channel routing solutions are generated for each net. Each solution is then modeled as a TSP city that can be traversed to maintain a complete journey. This approach results in a near-optimal channel routing solution which is then used for detailed routing of nets through the grids. The effectiveness of our algorithm was demonstrated by benchmarking it over a set of real-life chip specifications. Comparing our results to the best known approaches, on average, we obtained reductions of approximately 38.9 % on pin-count and 35.3 % on total wire length. These significantly better routability results were achieved due to the integration of pin-count reduction and wire length minimization cost functions.

Successful implementation of CAD tools for DMFB technology opens many avenues of inquiry, especially in chip-level design. For example, coupling chip reliability into the CAD flow of EWOD is an interesting problem to tackle. Control pin/signal sharing might introduce additional and unnecessary electrode actuations, which has the potential to make an electrode confront excessive actuations in case of a naïve design. Studies on EWOD chips have reported that this kind of problem eventually leads to a permanent degradation of dielectric layer [22]. This scenario inevitably impedes correct fluidic controls and therefore degrades the chip reliability. Thus, it becomes desirable and crucial to balance pin sharing and reliability preservation when the chip size and assay functionality grow.

Optimization in energy domains also needs to be investigated. Optimization problems that span several energy domains (e.g., electrical, circuit, fluidic, and thermal domains) appear to be extremely difficult due to the further involvements of energy-related constraints. For example, thermal-aware signal planning is important for the prevention of fluidics from overheating in some areas that have congested electrical connections.

Current research is also conducted to provide a feedback to the control software from the underlying hardware platform for the purpose of error recovery [16]. The motivation is that error recovery based on the repetition of experiments leads to wastage of expensive reactant and hard-to-prepare samples. However, these efforts are presented as "physical-aware" system reconfiguration techniques, which should be coupled with the chip-level design process for a cost-effective design.

References

1. Alpert, C.J., Mehta, D.P., Sapatnekar, S.S. (eds.): Handbook of Algorithms for Physical Design Automation. CRC Press, Boca Raton (2008)
2. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973). doi:10.1145/362342.362367

3. Burns, M.A., Johnson, B.N., Brahmasandra, S.N., Handique, K., Webster, J.R., Krishnan, M., Sammarco, T.S., Man, P.M., Jones, D., Heldsinger, D., Mastrangelo, C.H., Burke, D.T.: An integrated nanoliter DNA analysis device. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **282**(5388), 484–487 (1998). doi:10.1126/science.282.5388.484
4. Business Communications Company Inc: Global Biochip Markets: Microarrays and Lab-on-a-Chip, chap. Biotechnology. BCC (2013). http://www.bccresearch.com/market-research/biotechnology/biochip-markets-microarrays-bio049e.html. Accessed 19 June 2015
5. Cazals, F., Karande, C.: A note on the problem of reporting maximal cliques. Theor. Comput. Sci. **407**(1–3), 564–568 (2008). doi:10.1016/j.tcs.2008.05.010
6. Chakrabarty, K.: Design automation and test solutions for digital microfluidic biochips. IEEE Trans. Circuits Syst. I: Regul. Pap. **57**(1), 4–17 (2010). doi:10.1109/TCSI.2009.2038976
7. Cho, M., Pan, D.Z.: A high-performance droplet router for digital microfluidic biochips. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(10), 1714–1724 (2008). doi:10.1109/TCAD.2008.2003282
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **26**(1), 29–41 (1996). doi:10.1109/3477.484436
9. Fair, R., Khlystov, A., Tailor, T., Ivanov, V., Evans, R., Griffin, P., Srinivasan, V., Pamula, V., Pollack, M., Zhou, J.: Chemical and biological applications of digital-microfluidic devices. IEEE Des. Test Comput. **24**(1), 10–24 (2007). doi:10.1109/MDT.2007.8
10. Fischetti, M., Gonzàlez, J.J.S., Toth, P.: The symmetric generalized traveling salesman polytope. Networks **26**(2), 113–123 (1995). doi:10.1002/net.3230260206
11. Gong, J., Kim, C.J.: Direct-referencing two-dimensional-array digital microfluidics using multilayer printed circuit board. J. Microelectromech. Syst. **17**(2), 257–264 (2008). doi:10.1109/JMEMS.2007.912698
12. Ho, T.Y., Zeng, J., Chakrabarty, K.: Digital microfluidic biochips: a vision for functional diversity and more than Moore. In: 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'10), pp. 578–585 (2010)
13. Huang, T.W., Lin, Y.Y., Chang, J.W., Ho, T.Y.: Chip-level design and optimization for digital microfluidic biochips. In: 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS'11), pp. 1–4 (2011). doi:10.1109/MWSCAS.2011.6026535
14. Huang, T.W., Yeh, S.Y., Ho, T.Y.: A network-flow based pin-count aware routing algorithm for broadcast-addressing EWOD chips. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(12), 1786–1799 (2012). doi:10.1109/TCAD.2011.2163158
15. International Roadmap Committee: International Technology Roadmap for Semiconductors, chap. Design. ITRS (2009). http://www.itrs.net/. Accessed 19 June 2015
16. Luo, Y., Chakrabarty, K., Ho, T.Y.: Error recovery in cyberphysical digital microfluidic biochips. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(1), 59–72 (2013). doi:10.1109/TCAD.2012.2211104
17. Pollack, M.G., Shenderov, A.D., Fair, R.B.: Electrowetting-based actuation of droplets for integrated microfluidics. Lab Chip **2**, 96–101 (2002). doi:10.1039/B110474H
18. Song, J., Evans, R., Lin, Y.Y., Hsu, B.N., Fair, R.: A scaling model for electrowetting-on-dielectric microfluidic actuators. Microfluid. Nanofluidics **7**, 75–89 (2009). doi:10.1007/s10404-008-0360-y
19. Srinivasan, V., Pamula, V.K., Fair, R.B.: An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids. Lab Chip **4**, 310–315 (2004). doi:10.1039/B403341H
20. Su, F., Hwang, W., Chakrabarty, K.: Droplet routing in the synthesis of digital microfluidic biochips. In: Design, Automation and Test in Europe (DATE'06), pp. 1–6 (2006). doi:10.1109/DATE.2006.244177
21. Xu, T., Chakrabarty, K.: Broadcast electrode-addressing for pin-constrained multi-functional digital microfluidic biochips. In: 45th ACM/IEEE Design Automation Conference (DAC'08), pp. 173–178 (2008). doi:10.1109/TCAD.2011.2116250

22. Yeh, S.H., Chang, J.W., Huang, T.W., Ho, T.Y.: Voltage-aware chip-level design for reliability-driven pin-constrained EWOD chips. In: International Conference on Computer-Aided Design (ICCAD'12), pp. 353–360 (2012). doi:10.1145/2429384.2429461
23. Yen, J.Y.: Finding the k shortest loopless paths in a network. J. Manag. Sci. **17**(11), 712–716 (1971)
24. Yuh, P.H., Yang, C.L., Chang, Y.W.: Bioroute: a network-flow based routing algorithm for digital microfluidic biochips. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (ICCAD'07), pp. 752–757 (2007). doi:10.1109/TCAD.2008.2006140

Chapter 11
Quantum Dot Cellular Automata: A Promising Paradigm Beyond Moore

Kunal Das, Arijit Dey, Dipannita Podder, Mallika De and Debashis De

Abstract The quantum dot cellular automata (QCA) is a promising paradigm to overcome the ever-growing needs in size, power and speed. In this chapter we explore charge-confined low-power optimum logic circuit design to enhance the computing performance of a novel nanotechnology architecture, the quantum dot cellular automata. We investigate robust and reliable diverse logic circuit design, such as hybrid adders and other binary adder schemes, among them bi-quinary and Johnson–Mobius, in QCA. We also examine zero-garbage lossless online-testable adder design in QCA. Multivalued logic circuit design, with potential advantages such as greater data storage, fast arithmetic operation, and the ability to solve nonbinary problems, will be important in multivalued computing, especially in the ternary computing paradigm.

11.1 Introduction

It is well known that the complementary metal oxide semiconductor (CMOS) technology-based digital computers conceived two innovative ideas, in one idea information are represented with binary '0' and binary '1' and another one is that electronic charge state is used to represent the information in terms of current switch. The CMOS provides micro scale computing with high-density and low-power very large-scale integrated circuit (VLSI). However, such technology was found to have several drawbacks like high leakage of current, power dissipation in terms of heat, and limitation of speed in GHz range. Moreover, this technology has arrived at

K. Das (✉) · A. Dey · D. Podder
B. P. Poddar Institute of Management and Technology, Kolkata, India
e-mail: kunaldasqca@gmail.com

M. De
Dr. Sudhir Chandra Sur Degree Engineering College,
Surer math, Kolkata, West Bengal, India

D. De
Department of Computer Science and Engineering, West Bengal University
of Technology, Salt Lake City, Kolkata, India

its limitation as per Moore's Law, i.e., unit cost is shrinking as number of circuit components rises. Every 18 months number of circuit components become double [1] as well as the industry is now facing an increasing important trends of "More-than-Moore," reported in the Semiconductor Industries Association's International Roadmap for Semiconductors [2]. Researchers have to find out a strong alternative of CMOS technology for VLSI design. Nanotechnology was found as a strong alternative and have computational intelligence in electronic design, subject to some confusion and controversy and complicated by the fact that there are naturally occurring nano-sized materials and other nano-size particles, in the range from 1 μm down to 10 Å. Nanotechnologies won its existence in development within the field of microelectronics. Nanomechanical computing elements are scalable in terms of input size and depth of propagation path analyzed using a bounded continuum model. Boolean logic functions of NOT, AND, OR, and XOR are realized. Nanotechnology should not be viewed as a single technology that only affects the specific area. It compensates the limitation in many existing technologies. Quantum dot cellular automaton (QCA) is an emerging research domain in nanotechnology [3–10]. Quantum dots are semiconductors confined in all three dimensions of space or alternatively, it can be stated that quantum dot is a simple charge container and it is three dimensionally confined [8]. The promising alternative of CMOS paradigm is the quantum dot cellular automata (QCA) which is used to represent the information in binary '1' and binary '0' in terms of electronic charge configuration. In 1993, C.S. Lent et al. first introduced the theoretical quantum dot cellular automata [3] and in early 1999, C.S. Lent et al. described the experimental approach to design QCA cell with GaAs [8]. The dynamic behavior of QCA was discussed with the help of the Hart tree approximation [4]. Quantum mechanics is also involved in finding out the cell size and dot radius of a single QCA cell. Hence, QCA became research interest to establish as strong CMOS alternative. During last decades, in nanotechnology era, an exhaustive research has been carried out in this domain. QCA is still in infancy stage, need lots of study for QCA logic circuit design. The low-power reversible logic circuit design, tile-based logic circuit design as well as its defect analysis are prime problem domain. The ternary computing with QCA is the most challenging task in this domain since no such improvement is noticed. The multivalued computing, specifically ternary computing is an emerging domain of research due to the potential advantages like greater data storage capability, faster arithmetic operations, better support for numerical analysis, application of nondeterministic and heuristic procedures, communication protocol and effective solution for nonbinary problems [11–18].

Rest of the chapter is organized as follows. In Sect. 11.2, the classical adder circuit designs in QCA are discussed. The lossless reversible and conservative logic circuit design and computation are explored in Sect. 11.3. The ternary computing in QCA is an intelligent step toward the computational aspect in Micro–Nano electronics era. The realization of ternary QCA is explored in Sect. 11.4. Finally, the important result and effectiveness are discussed in Sect. 11.5. A few burning challenges in QCA are also discussed as an open problem toward the researcher in this Micro–Nano electronics area.

11.2 Classical Adder Circuit Design

Addition is a fundamental operation for any digital system, digital signal processing, or control system. Adders are also very important component in digital systems because of their extensive use in other basic digital operations, such as subtraction, multiplication, and division. Rest of the section are organized as follows. In Sect. 11.2.1, the binary adder design in QCA is explored. In Sect. 11.2.2, the decimal adder design and implementation in QCA is demonstrated. The comparison is also made with the different adders in terms of complexity, area, delay, and cost.

11.2.1 Binary Adder

In digital electronics the adder circuits, i.e., half adder and full adder are used as basic building block to add binary numbers. Parallel adders compute the addition of multiple bit binary number, one of which is renowned as ripple carry adder. It is constructed by cascading full adders (FA) blocks in series. The carryout of one stage is fed directly to the carry-in of the next stage. In QCA technology, ripple carry adder has designed previously in [19].

11.2.1.1 Carry Look Ahead Adder

A carry look ahead adder improves speed by reducing the time required for determining carry bits. It calculates one or more carry bits ahead the sum, reduces the delay to calculate the result of the higher order bits. In QCA the carry output can be defined as follows using the majority voter where a, b, Cin defined as inputs of adder circuit and majority voter defined as m (a, b, c) = ab + bc + ca.

$$\mathrm{Cout} = \mathrm{a.b.Cin} + \mathrm{a.b.}\overline{\mathrm{Cin}} + \mathrm{a.}\bar{\mathrm{b}}\mathrm{.Cin} + \bar{\mathrm{a}}\mathrm{.b.Cin}$$

$$\mathrm{Cout} = \mathrm{a.b.Cin} + \mathrm{a.b.}\overline{\mathrm{Cin}} + \mathrm{a.b.Cin} + \mathrm{a.}\bar{\mathrm{b}}\mathrm{.Cin} + \mathrm{a.b.Cin} + \bar{\mathrm{a}}\mathrm{.b.Cin}$$

$$\mathrm{Cout} = \mathrm{a.b.}(\mathrm{Cin} + \overline{\mathrm{Cin}}) + \mathrm{b.Cin}(\mathrm{a} + \bar{\mathrm{a}}) + \mathrm{a.Cin}(\mathrm{b} + \bar{\mathrm{b}})$$

$$\mathrm{Cout} = \mathrm{a.b} + \mathrm{b.Cin} + \mathrm{a.Cin}$$

$$\mathrm{Cout} = \mathrm{m}(\mathrm{a, b, Cin})$$

The majority function definition for the Sum output is as follows:

$$\mathrm{Sum} = \mathrm{a.b.Cin} + \bar{\mathrm{a}}\mathrm{.}\bar{\mathrm{b}}\mathrm{.Cin} + \bar{\mathrm{a}}\mathrm{.b.}\overline{\mathrm{Cin}} + \mathrm{a.}\bar{\mathrm{b}}\mathrm{.}\overline{\mathrm{Cin}}$$

$$\mathrm{Sum} = (\mathrm{a.b.Cin} + \bar{\mathrm{a}}\mathrm{.}\bar{\mathrm{b}}\mathrm{.Cin}) + (\mathrm{a.b.Cin} + \bar{\mathrm{a}}\mathrm{.b.}\overline{\mathrm{Cin}}) + (\mathrm{a.b.Cin} + \mathrm{a.}\bar{\mathrm{b}}\mathrm{.}\overline{\mathrm{Cin}})$$

$$\mathrm{Sum} = (\bar{\mathrm{a}}\mathrm{.b} + \bar{\mathrm{a}}\mathrm{.Cin} + \mathrm{b.Cin})(\mathrm{a.}\bar{\mathrm{b}} + \bar{\mathrm{b}}\mathrm{.Cin} + \mathrm{a.Cin}) + (\mathrm{a.}\bar{\mathrm{b}} + \bar{\mathrm{b}}\mathrm{.Cin} + \mathrm{a.Cin})(\mathrm{a.b} + \mathrm{b.}\overline{\mathrm{Cin}} + \mathrm{a.}\overline{\mathrm{Cin}})$$

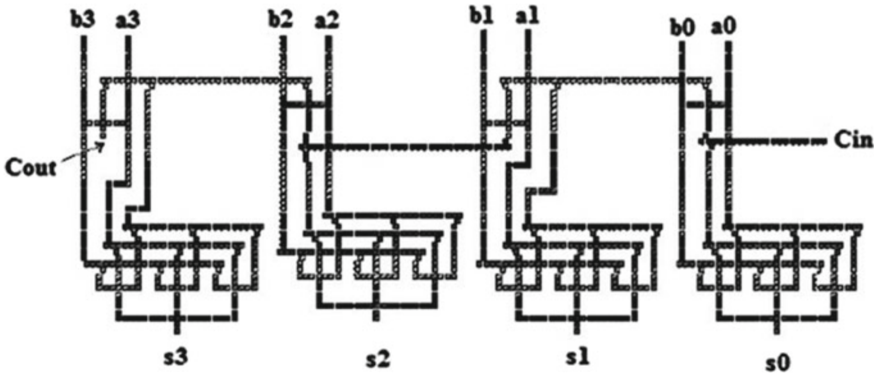


Fig. 11.1 QCA layout of a 4-bit carry look ahead adder

$$+ (\bar{a}.b + \bar{a}.Cin + b.Cin)(a.b + a.\overline{Cin} + b.\overline{Cin})$$

$$Sum = m(\bar{a}, b, Cin).m(a, \bar{b}, Cin) + m(a, \bar{b}, Cin).m(a, b, \overline{Cin})$$
$$+ m(\bar{a}, b, Cin).m(a, b, \overline{Cin})$$

$$Sum = m(m(\bar{a}, b, Cin), m(a, \bar{b}, Cin), m(a, b, \overline{Cin})),$$

n-bit adder can be designed by arranging typical *n*-structures of single full adder with carry look ahead, vertically in a column. A 4-bit carry look ahead adder design is shown in Fig. 11.1.

11.2.1.2 Brent–Kung Adder

Prefix adder is a special kind of parallel adder which reduces carry computation to a “prefix” computation [20]. Brent–Kung adder in QCA paradigm, mentioned in [19] has lower complexity than the other prefix adders. The small shaded circles in the prefix graph of a 16-bit Brent–Kung adder shown in Fig. 11.2 represent the associative operator “∘”.

The general formulation of prefix adders in terms of the associative operator “∘” defined as follows:
(Note that ∘ is also the fundamental carry operation)

$$(G_i, P_i) \circ (G_j, P_j) = (G_i + (P_iG_j), P_iP_j) \tag{11.2.1}$$

Let genarator $g_i = a_ib_i$ and propagator $p_i = a_i + b_i$.

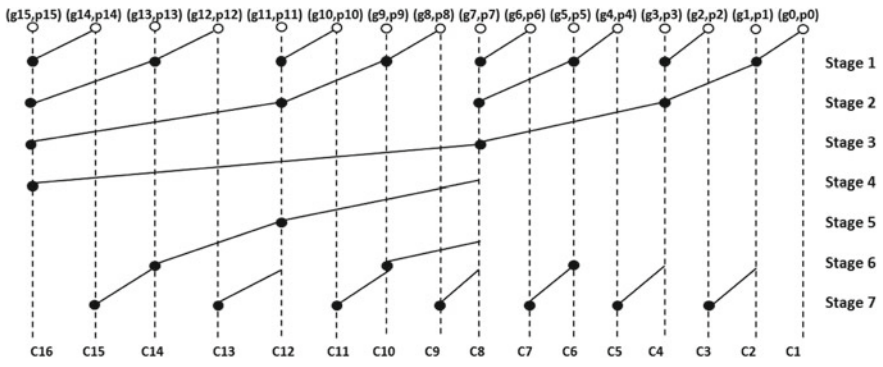

(g15,p15) (g14,p14) (g13,p13) (g12,p12) (g11,p11) (g10,p10) (g9,p9) (g8,p8) (g7,p7) (g6,p6) (g5,p5) (g4,p4) (g3,p3) (g2,p2) (g1,p1) (g0,p0)
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Stage 7
C16 C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1

Fig. 11.2 16-bit Brent–Kung adder prefix graph

In particular, (11.2.2)–(11.2.5) apply to all forms of prefix adders:

$$(c_1, 0) = (g_0, p_0) \circ (c_0, 0) \tag{11.2.2}$$

$$(c_2, 0) = (g_1, p_1) \circ [(g_0, p_0) \circ (c_0, 0)] \tag{11.2.3}$$

$$(c_3, 0) = (g_2, p_2) \circ [(g_1, p_1) \circ (g_0, p_0) \circ (c_0, 0)] \tag{11.2.4}$$

$$(c_4, 0) = [(g_3, p_3) \circ (g_2, p_2)] \circ [(g_1, p_1) \circ (g_0, p_0) \circ (c_0, 0)] \tag{11.2.5}$$

where c_0, c_1 are defined as input carry and output carry of stage 1 respectively. The equations for the Brent–Kung prefix adder can be developed using the following:

$$(g_{i:j}, p_{i:j}) = (g_i, p_i) \circ (g_{i-1}, p_{i-1}) \ldots \circ (g_{j-1}, p_{j-1}) \circ (g_j, p_j) \tag{11.2.6}$$

Suppose m is an integer defined as $j < m < i$; then (11.2.6) can be rewritten as shown in (11.2.7):

$$(g_{i:j}, p_{i:j}) = (g_{i:m}, p_{i:m}) \circ (g_{m-1:j}, p_{m-1:j}) \tag{11.2.7}$$

Using (11.2.7), we can rewrite (11.2.3) as

$$c_2 = (g_{1:0}, p_{1:0}) \circ (c_0, 0)$$

In general, c_{i+1} can be expressed as

$$c_{i+1} = (g_{i:0}, p_{i:0}) \circ (c_0, 0)$$

If initial carry $c_0 = 0$, then $c_{i+1} = g_{i:0}$.

In general, for an n-bit Brent–Kung adder, number of majority gates require to implement is $8n - 3log_2(n) - 4$ [19].

11.2.2 Decimal Adder

Decimal adder is required for the computers or calculators that perform arithmetic operations directly in the decimal number system. The very renowned decimal adder is BCD adder [21], which is also implemented in QCA technology.

11.2.2.1 Parallel Decimal JMC Adder

Apart from renowned BCD adder, Johnson–Mobius coded adder was designed in [22]. The encoding technique is mentioned in Table 11.1. The unique property of this code is that in the decimal JMC, the digit 5–9 can be obtained just by inverting the bits representation of 0–4.

The block diagram of the decimal parallel Johnson–Mobius adder is shown in Fig. 11.3. The parallel left barrel twisted-ring rotator and a decoder in the path of the second operand B is used. The truth tables for the barrel twisted-ring rotator and decoder are given in Tables 11.2 and 11.3 respectively.

Table 11.1 Johnson–Mobius code encoding technique

Digit	Johnson–Mobius code
0	0 0000
1	0 0001
2	0 0011
3	0 0111
4	0 1111
5	1 1111
6	1 1110
7	1 1100
8	1 1000
9	1 0000

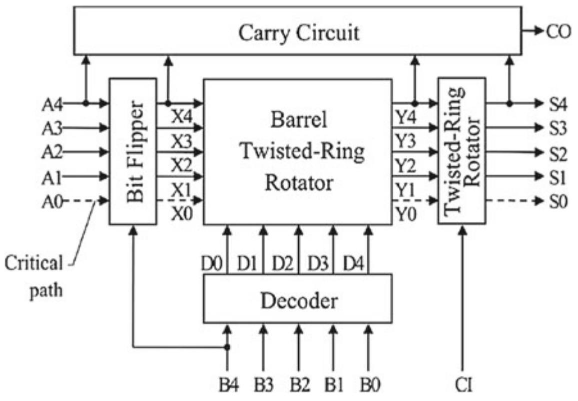


Fig. 11.3 Block diagram of second parallel decimal Johnson–Mobius coded adder

Table 11.2 Truth table of barrel twisted-ring rotator

D0	D1	D2	D3	D4	Y4	Y3	Y2	Y1	Y0
1	0	0	0	0	X4	X3	X2	X1	X0
0	1	0	0	0	X3	X2	X1	X0	$\overline{X4}$
0	0	1	0	0	X2	X1	X0	$\overline{X4}$	$\overline{X3}$
0	0	0	1	0	X1	X0	$\overline{X4}$	$\overline{X3}$	$\overline{X2}$
0	0	0	0	1	X0	$\overline{X4}$	$\overline{X3}$	$\overline{X2}$	$\overline{X1}$

Table 11.3 Truth table of the decoder

B4	B3	B2	B1	B0	D0	D1	D2	D3	D4
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	1	0	0	0
0	0	0	1	1	0	0	1	0	0
0	0	1	1	1	0	0	0	1	0
0	1	1	1	1	0	0	0	0	1
1	1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	1	0	0	0
1	1	1	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1

```
Algorithm 2.2:Serial Decimal Johnson–Mobius coded adder

IF (B4 = 1), THEN
INVERT A4 A3 A2 A1 A0;
INVERT B3 B2 B1 B0;
2) CALCULATE D (B3, B2, B1, B0);
3) D-bit TWISTED-RING ROTATE A4 A3 A2 A1 A0;
4) IF (CI = 1), THEN
TWISTED-RING ROTATE A4 A3 A2 A1 A0;
END,
Comment:
Where A4 A3 A2 A1 A0 is the JMC of the first operand, B4 B3 B2 B1 B0 is the
JMC of the second operand, D = {0, 1, 2, 3, 4} is the shift amount, and CI is the
carry-in bit.
If the INVERT (I) or D-bit TWISTED-RING ROTATE+TWISTED RING
ROTATE (DTR+TR), changes the MSB of the first operand (MSBF) A4 from "1"
to "0." then the carry-out bit CO will be set to "1" [23]
```

11.2.2.2 Serial Decimal JMC Adder

The serial decimal adder in QCA, proposed in [23], gives a good impact in serial data processing in QCA technology. The algorithm of the serial addition algorithm is mentioned in Algorithm 2.2. The serial data is represented using strobe signals for the I/O data: Strobe A, Strobe B, Strobe S, and Strobe C are the time gates of the first operand, the second operand, the sum, and the carryout bit respectively. The block diagram of the serial adder is shown in Fig. 11.4a. Here the operation unit processes the first operand and control unit generate control signals according to the bit values of the second operand.

At first MSB of operand A and B are transferred. The MSB of the second operand B4 is stored in a 1-bit register (BR) which is implemented by a 2-to-1 multiplexer (Fig. 11.4b), controls serial-bit flippers (SBF). The SBF is shown in Fig. 11.4c. The lower bits of the second operand are stored in a 4-BR. The outputs of decoder provide the shift amount D that is encoded by the unitary code. The switching bits D are determined as follows:

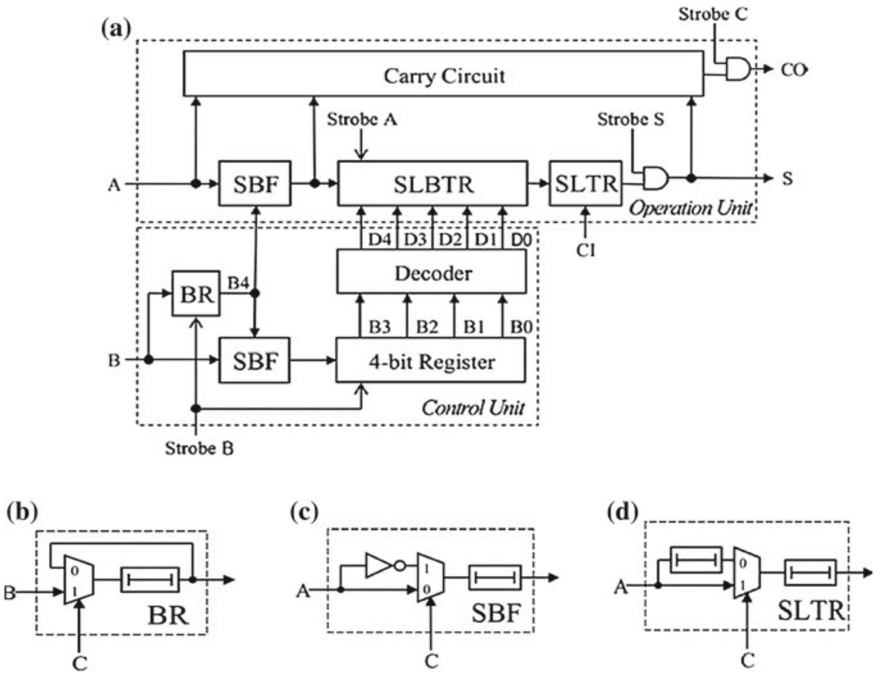


Fig. 11.4 Block diagram of **a** Serial decimal Johnson–Mobius coded adder, **b** Serial-bit flipper, **c** 1-bit register, **d** Serial left twisted-ring rotator

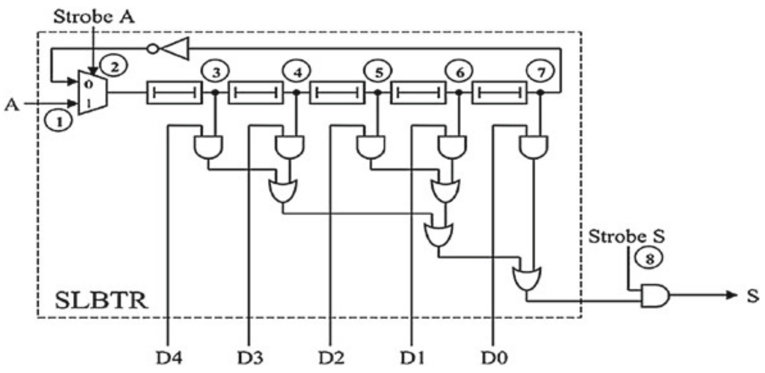


Fig. 11.5 Block diagram of serial left barrel twisted-ring rotator

$$D0 = \overline{B0}$$

$$D1 = B0\overline{B1}$$

$$D2 = B1\overline{B2}$$

$$D3 = B2\overline{B3}$$

$$D4 = B3$$

The serial left barrel twisted-ring rotator (SLBTR) is controlled by Strobe A. It is also noticed from the block diagram of SLBTR, mentioned in Fig. 11.5 that it contains shift register implemented by five delay elements. If Strobe A = 1, then the serial code of the first operand enters into the shift register whereas if Strobe A = 0, then the shift register functions as a Johnson counter. In this way, it calculates the serial code of sum S.

The serial left twisted-ring rotator (SLTR) which contains a delay element and a 2-to-1 multiplexer, rotates the operand A by getting CI = 1, shown in Fig. 11.4d. And carry circuit calculates the carryout bit according to

$$CO = [\overline{A4}(t+1)A4(t)]_1 V[\overline{A4}(t+1)A4(t)]_{DTR+TR}$$

11.2.2.3 Bi-quinary Coded Parallel Decimal Adder

Another parallel decimal adder is discussed in this section with 6-bit bi-quinary encoding technique, mentioned in Table 11.4. In this code the quinary components repeat themselves by getting inverted binary bit. The Algorithm 2.3 is suggested for this adder [24].

Table 11.4 Bi-quinary encoding technique

Decimal digit	543210
0	0 0 0 0 0 1
1	0 0 0 0 1 0
2	0 0 0 1 0 0
3	0 0 1 0 0 0
4	0 1 0 0 0 0
5	1 0 0 0 0 1
6	1 0 0 0 1 0
7	1 0 0 1 0 0
8	1 0 1 0 0 0
9	1 1 0 0 0 0

```
Algorithm 2.3:Bi-quinary coded Parallel Decimal Adder

1) CALCULATE D (A4, A3, A2, A1, A0, B4, B3, B2, B1, B0);
2) D5c = CORRECT D5 (A5, B5, D5);
3) CALCULATE Co (A5, B5, D5c);
4) SHIFT ();
END,
PROCEDURE SHIFT ()
1) IF (CI = 1), THEN
LEFT ROTATE D4, D3, D2, D1, D0;
2) IF (D4 = 1 AND CI = 1), THEN
INVERT D5c;
3) IF (D4=1 AND D5=1), THEN
INVERT Co;
END,
```

Where A5 A4 A3 A2 A1 A0 is the first operand, B5 B4 B3 B2 B1 B0 is the second operand,
D {D5, D4, D3, D4, D1, D0} is the result of sum, produced by the quinary components of operands.CI is the carry-in bit. D5c is the binary component of sum after correction.[24]

The block diagram based on this algorithm is shown in Fig. 11.6.

At first the sum, D{D5, D4, D3, D4, D1, D0} is calculated by sum generator which processes the quinary components of the second operand (addend), B{B4, B3, B2, B1, B0}, under the control of the first operand (augend) bits, A{A4, A3, A2, A1,A0}. The truth table of Sum generator is shown in Table 11.5.

The MSB of D will be corrected by correction block. The function of the correction block is D5c = (A5 $\oplus$ B5 $\oplus$ D5).

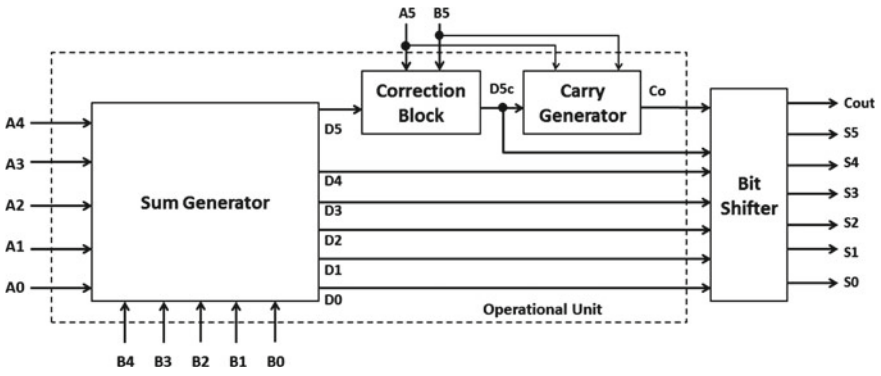


Fig. 11.6 Block diagram of bi-quinary coded parallel decimal adder

Table 11.5 Truth table of sum generator

A4	A3	A2	A1	A0	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	B4	B3	B2	B1	B0
0	0	0	1	0	B4	B3	B2	B1	B0	B4
0	0	1	0	0	B4+B3	B2	B1	B0	B4	B3
0	1	0	0	0	B4+B3+B2	B1	B0	B4	B3	B2
1	0	0	0	0	B4+B3+B2+B1	B0	B4	B3	B2	B1

Then the carry generator generates the carry of the quinary components. The function is $\text{Co} = \Big((\text{A5} + \text{B5})\Big)\overline{\text{D5c}} + \text{A5B5}$.

And finally if Cin = 1 all the previously calculated output are shifted in special manner to produce the actual output, otherwise the outputs remain same. The block diagram is shown in Fig. 11.7. If Cin = 1, then the quinary components of the sum

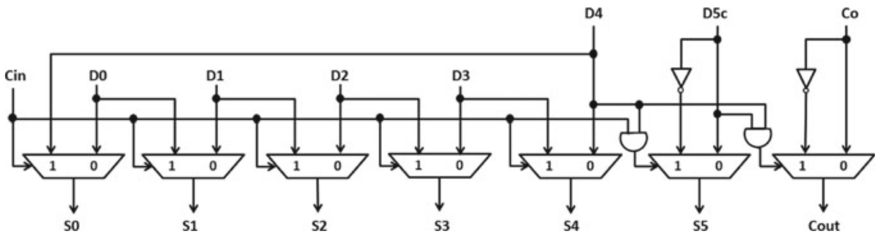


Fig. 11.7 Block diagram of bit shifter

shift toward left. Again D5 will be inverted at the condition Cin = 1 and D4 = 1 and Co will be inverted at the condition D5 = 1 and D4 = 1. The block consists of seven multiplexer and two AND gates and two inverters.

QCA layouts of the functional block for D5 and D4 bits. The sum generators are shown in Fig. 11.8a, b respectively. Similarly D3, D2, D1, D0 can be implemented. The correction block, carry generator, and bit shifter are implemented, and the QCA layouts are shown in Fig. 11.8c–e respectively.

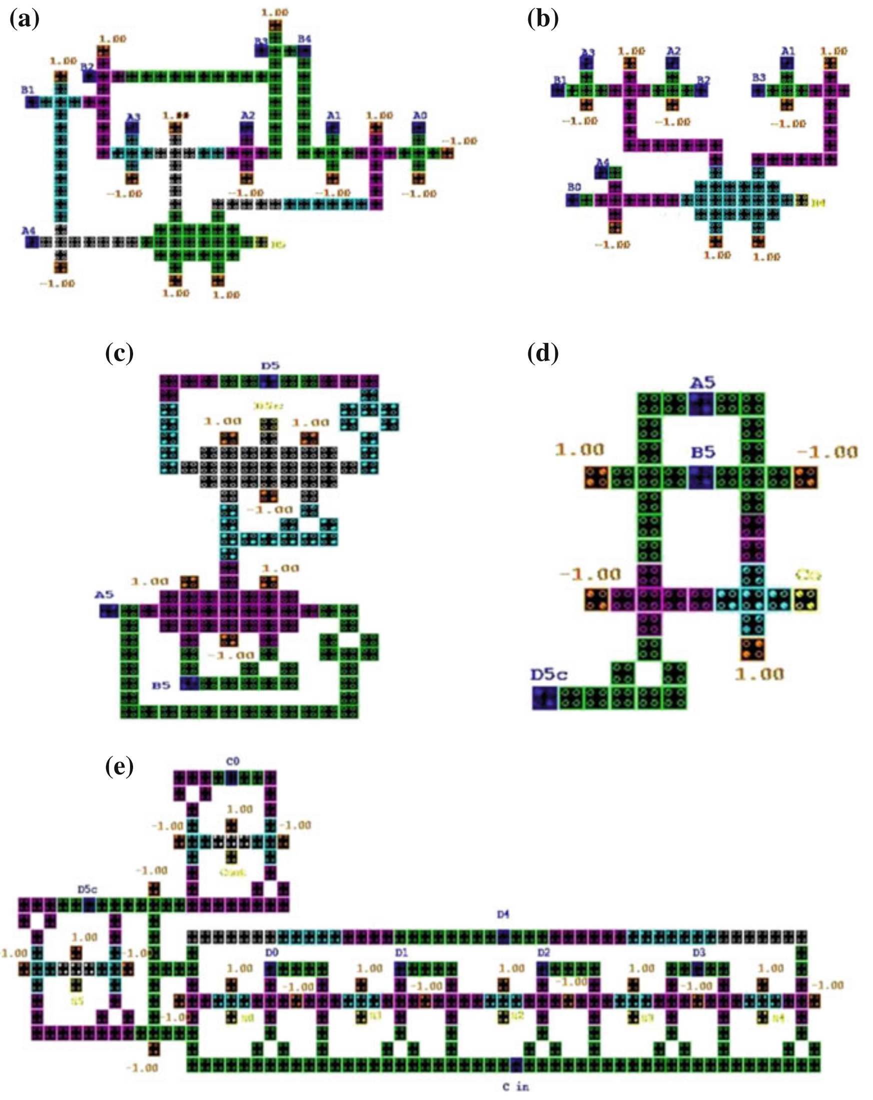


Fig. 11.8 QCA layout: **a** and **b** are output of sum generator, D5 and D4 respectively, **c** correction block, **d** carry generator, **e** bit shifter

11.2.2.4 Simulation and Result

All simulation results validates that the proposed adder works properly. Let us assume A = 8, B = 4, Cin = 1. Bi-quinary representation of A, B are 10100, 01000. D5, D4, D3, D2, D1, D0 are calculated using sum generator. The simulation result of Sum generator is shown in Fig. 11.9a. The simulation result of correction block and

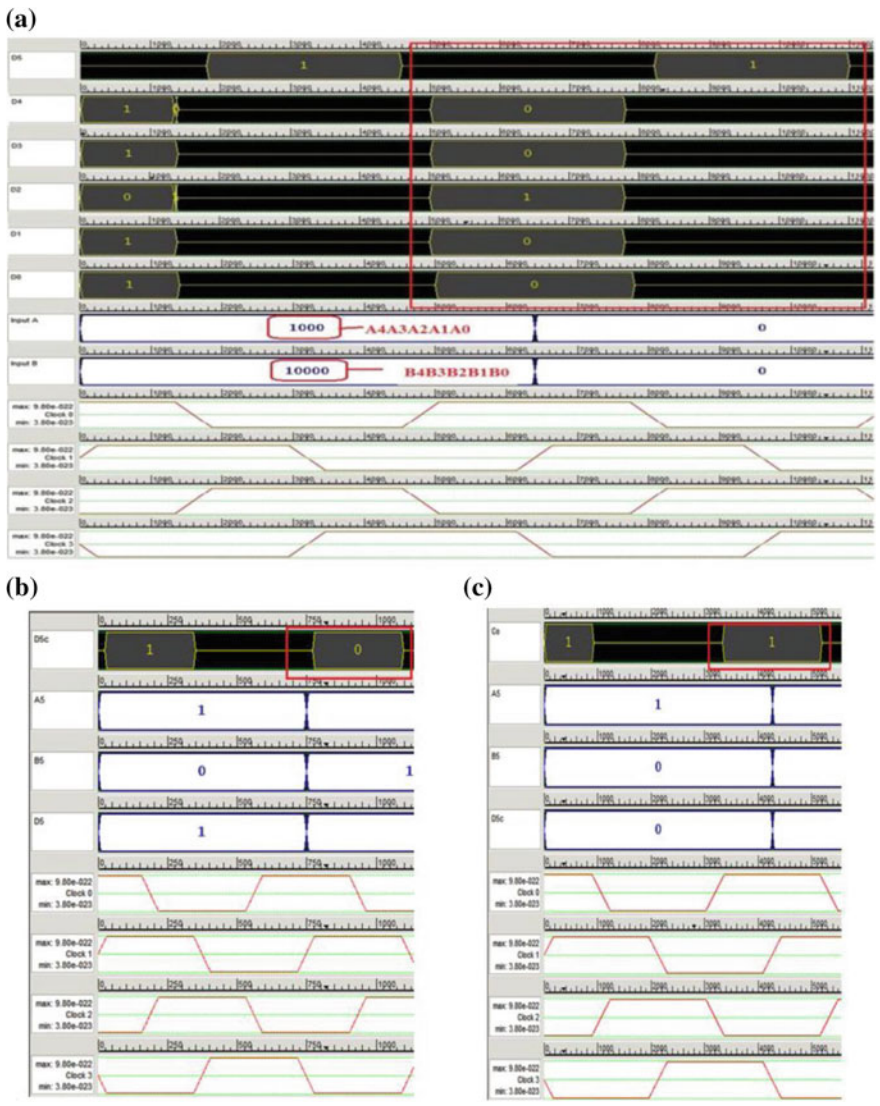


Fig. 11.9 Simulation result for QCA layout designed with QCA designer **a** sum generator, **b** correction block, **c** carry generator, **d** bit shifter

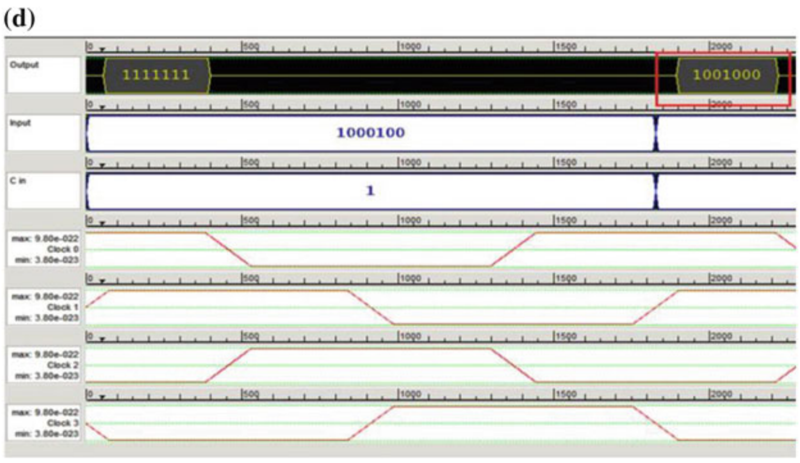


Fig. 11.9 (continued)

carry generator are shown in Fig. 11.9b, c respectively. Finally the shifting block executes, and provides the output 1001000 as shown in Fig. 11.9d.

11.2.2.5 Comparisons of Decimal Adders

All blocks of the bi-quinary coded decimal adder consist of 1266 cells with required area 2.14 μm^2. The total propagation delay of all blocks of the proposed adder is six clock cycles. The comparison with previously proposed adders is mentioned in Table 11.6. Complexity (number of cells), required area (size of QCA cell $=$ 18 nm $\times$ 18 nm, center-to-center distance $=$ 20 nm), propagation delay (latency), and cost function are given for each variant. It is known from [25] that $Cost = Area \times Delay \times Power$. It is observed from Table 11.6 that the bi-quinary coded decimal adder require lower complexity and minimal area among all parallel adders. It can reduce the delay also (Fig. 11.10).

Table 11.6 QCA adder comparison

Variant	Different adder	Complexity (cell)	Area (μm^2)	Delay	Cost
1	[a]Brent–Kung adder [19]	1782	1.498	2.5 CLK	6673.6
2	Parallel BCD adder [21]	1348	2.28	8 CLK	24588
3	Parallel JMC adder [22]	3560	4.00	7 CLK	99680
4	Serial JMC adder [23]	1130	1.77	10 CLK	20001
5	Bi-quinary coded decimal adder [24]	1266	2.14	6 CLK	16255

[a]Binary adder, while other adders are decimal adder

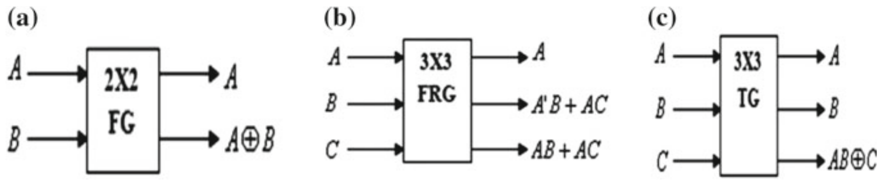


Fig. 11.10 Block diagram of **a** Feynman Gate, **b** Fredkin Gate, **c** Toffoli Gate

11.3 Reversible and Conservative Adder Circuit Design

The conventional computer system uses the irreversible technique, i.e., once the output is generated from the logic block, the input bits are lost and the power is retained in the system. In this regard, reversible computing plays an important role in the VLSI design. In VLSI, micro scale computing with high density and low power is provided by the complementary metal oxide semiconductor (CMOS). However, this technology suffers from several drawbacks like high leakage of current, power dissipation in terms of heat and limitation of speed in GHz range as well as this technology has arrived at its limitation as per Moor's Law when unit cost is falling as number of components per circuit rises [1]. Reversible computing is one of the possible solutions to perform computation with almost zero power dissipation. In the low power nanocomputing era, reversible and conservative logic gate design is emerging as an important area of research. Launder [26] has shown that for irreversible logic computation each bit information loss produces $K_B T \ln 2\,J$ of heat energy, where K_B is Boltzman's constant and T is the absolute temperature at which computation is performed. Bennett [27] has proved zero power dissipation in case of reversible logic computation. In reversible logic gate the mapping between input vector I_V and output vector O_V is objective, i.e., each input yields to a distinct output (one-to-one mapping). Basically, Feynman Gate [27], Toffoli Gate [28], Fredkin Gate [29] commonly perform as reversible logic gate.

Reversible logic gate have several key features, like (a) minimum number of garbage output, (b) minimum input constants, (c) minimum circuit level, (d) minimum number of gates [29]. However, one of the key features of reversible gate is to recover bit loss. But reversibility feature is not able to identify bit error in the circuit. The fault-tolerant circuit can be obtained by means of parity. In earlier several proposals of fault-tolerant circuit by parity preserving reversible gate or conservative logic gate have been introduced. Conservative logic gate is inputs from input vector I_V and outputs from output vector O_V are mapped in a way where parity of inputs in I_V and outputs in O_V are parity preserving, i.e., number of 1's present in each input and number of 1's present in output must be same. In early report [30] an algorithm for $K \times K$ conservative logic has been introduced, where the inputs are defined in $(K + 1)$ number of sets in terms of parity. The following three algorithms define reversible logic design, conservative reversible logic design, and conservative logic design [30].

```
Algorithm 3.1: Reversible logic Design
Initialization:
    1.  Input vector $I_v = \{I_0, I_1, I_2,\ldots, I_{k-1}\}$
    2.  Output vector $O_v = \{\phi\}$
Loop1:
    3.  i = 0 to k-1
Loop2:
    4.  j = 0 to k-1
    5.  choose $I_j$ for Output vector $O_i$
    6.  if success  Oi = Ij
    7.  remove $I_j$ from $I_v$ and exit Loop2
    8.  end of Loop1
Final Output:
    9.  $O_v$ becomes final output vector.
    10. end
```

```
Algorithm 3.2: Conservative Reversible logic Design

    1.  Initialization:
    2.  Input vector
         $Iv = \{I_0,I_1,I_2,\ldots,I_{k-1}\}$
    3.  Output vector $Ov=\{\phi\}$
    4.  Loop1:
    5.  i=0 to k-1
    6.  Loop2:
    7.  j=0 to k-1
    8.  choose $I_j$ for Output vector Oi such that numbers of 1's(Ij)= No's of 1's(Ii)
    9.  if success  $O_i=I_j$
    10. remove $I_j$  from $I_v$ and exit Loop2
    11. end of Loop1
    12. Final Output:
    13. Ov becomes final output vector.
    14. End
```

```
Algorithm 3.3: Conservative logic Design
Initialization:
Input vector Iv =  {I0, I1, I2,…., Ik-1}
Output vector Ov= {$\phi$}
 Set Input vector set  S0 = {I0} and Sk= {Ik-1}
i = 1
Set Input vector Si  for all Iv where 'i' is number of 1's in Iv
Loop1:
j = 0 to $\lfloor k/2 \rfloor$
Numbers of element in set Si is   ni= (k+2j-1)
Numbers of element in set Sk+1-i  is nk+1-i = (k+2j-1)
increment i by 1
end of Loop1
Loop2:
i =1 to k-1
j =1and count =1
if numbers 1's of Ij=i then Set Ij in Si
increment j and count by 1
if ni equals to count then goto Loop2
else goto step 13
end of Loop2
Loop3:
i = 0 to 2k-1
Set Oi equals to any element from S0, S1,...,Sk ( same element allowed to take more than once) as numbers of 1's in Ii must be equal to numbers of 1's in Oi.
end of Loop3
Final Output:
Ov becomes final output vector.
end
```

11.3.1 Conservative Full Adder

The testable reversible logic gate is reported to design reversible full adder in [11, 12]. The garbage count optimization or zero garbage lossless circuit design has become prime research problem. In this context, a conservative logic gate is introduced to design a fault-tolerant, lossless zero garbage full adder [31]. To design the architecture of conservative full adder one PCLG block is required as shown in Fig. 11.11a, which is more effective than other previously reported full adder in terms of size of the architecture [29]. Here a 4×4 mapping technique is used to design the effective architecture of the full adder. The input vector I_V (A, B, C, D) are mapped to the outputs in output vector O_V ($P = B \oplus C \oplus D, Q = BC + BD + CD, R = BC + BD + CD, S = A$) as shown in Fig. 11.11a. Classification is made of inputs in I_V using the concept of set and an algorithm is defined [Algorithm 3.3]. In this work, the sets are defined in terms of presence of 1's in I_V. 4×4 mapping technique is used to design the conservative logic (PCLG), so that the required numbers of sets are 5 (S0, S1, S2, S3, S4) (see Algorithm 3.3) [30]. Another 3×3 PCLG is designed to implement zero garbage full adder. In the Fig. 11.11b, the block diagram of 3×3 PCLG is shown. The input vector I_V (A, B, C) are mapped to the outputs in output vector O_V ($P = A \oplus B \oplus C$, $Q = AB + BC + CA$, $R = AB + BC + CA$). The classification of set for 3×3 and 4×4 PCLG are shown in Tables 11.7 and 11.8 respectively. A tester reversible logic gate (TRLG) is also discussed as a tester block to test conservative logic gate as shown in Fig. 11.12a. The PCLG is universal, i.e., a PCLG block can implement the three basic gates. A single PCLG block is needed for implementing conservative full adder which generates zero garbage count.

11.3.2 Online Testing

In this section, the online testing strategy is explored to test the conservative full adder. In earlier some reports [32–34], the online testing strategy is introduced for reversible logic gate test in quantum computing. In this regards, a 5×5 reversible logic gate known as TRLG is designed and implemented to test the conservative full adder PCLG without any other computation, i.e., online test is performed. The TRLG is also designed with QCA designer [35] and simulated. The input vector

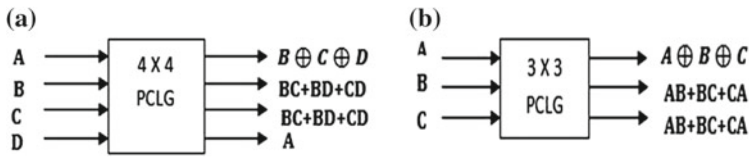


Fig. 11.11 Block diagram of **a** 4×4 conservative logic gate (PCLG), **b** 3×3 conservative logic gate (PCLG)

Table 11.7 Input vector showing set for 3 × 3 PCLG

I_V	Inputs	I_V Set	Outputs	O_V	O_V Set
	ABC		PQRS		
X0	000	S0	000	X0	S0
X1	001	S1	100	X4	S1
X2	010	S1	100	X4	S1
X3	011	S2	011	X3	S2
X4	100	S1	100	X4	S1
X5	101	S2	011	X3	S2
X6	110	S2	011	X3	S2
X7	111	S3	111	X7	S3

Table 11.8 Input vector showing set for 4 × 4 PCLG

I_V	Inputs	I_V Set	Outputs	O_V	O_V Set
	ABCD		PQRS		
X0	0000	S0	0000	X0	S0
X1	0001	S1	1000	X8	S1
X2	0010	S1	1000	X8	S1
X3	0011	S2	0110	X6	S2
X4	0100	S1	1000	X8	S1
X5	0101	S2	0110	X6	S2
X6	0110	S2	0110	X6	S2
X7	0111	S3	1110	X14	S3
X8	1000	S1	0001	X1	S1
X9	1001	S2	1001	X9	S2
X10	1010	S2	1001	X9	S2
X11	1011	S3	0111	X7	S3
X12	1100	S2	1001	X9	S2
X13	1101	S3	0111	X7	S3
X14	1110	S3	0111	X7	S3
X15	1111	S4	1111	X15	S4

I_V (A, B, C, D, E) of the reversible logic is mapped to the output vector O_V ($P = A \oplus B \oplus C \oplus D \oplus E, Q = A, R = B, S = C, T = D$) as shown in Fig. 11.12a. The two pair rail of 4 × 4 PCLG and TRLG is shown in Fig. 11.12b to test online 4 × 4 PCLG. If the PCLG block is fault free then its parity of input and the parity of output of TRLG should remain same, so that parity will be preserved. When TRLG input E = 0, the tester will perform the online testing if we railed with 4 × 4 PCLG as shown in Fig. 11.12b. The tester output will produce same parity output as input parity i.e. if input I_V is even parity or odd parity, the output 'P' of TRLG will produce

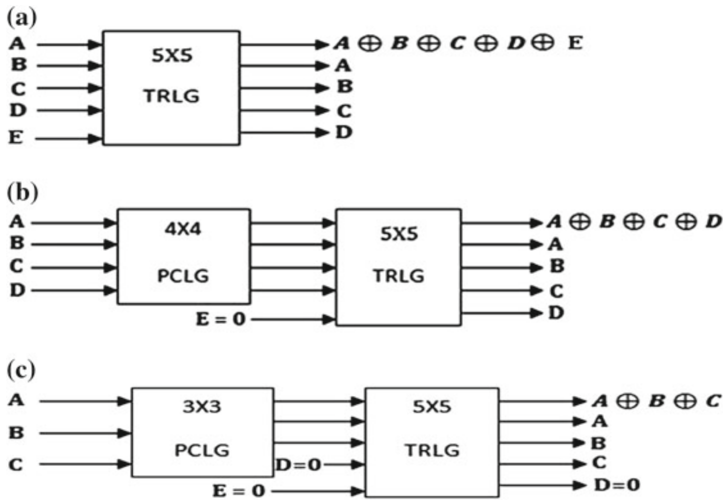


Fig. 11.12 **a** Block diagram of TRLG **b** Block diagram of online testing of 4 × 4 PCLG using 5 × 5 TRLG **c** Block diagram of online testing of 3 × 3 PCLG using 5 × 5 TRLG

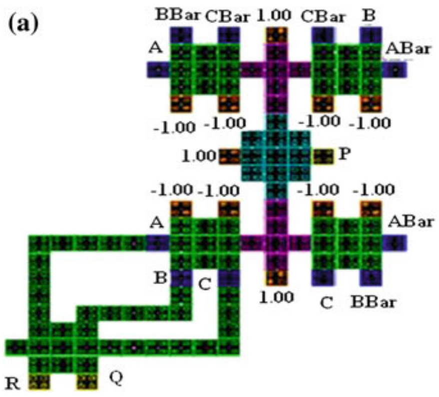
same parity and corresponding input vector I_V (A, B, C, D) will reflect with other output (Q, R, S, T). Similarly, the TRLG can test any 4 × 4 conservative logic gate (CLG) as well as 3 × 3 CLG. Figure 11.12c shows the 3 × 3 PCLG testability with two pair railed with 5 × 5 TRLG. In this test the tester sets input D as well as input E to zero, i.e., for D = E = 0., the tester will perform the online testing.

11.3.3 Simulation

The design, implementation, and simulation of two PCLGs and TRLG architecture is made by QCA designer [35]. Vector table is used for input vector I_V to design both the 3 × 3 PCLG and 4 × 4 PCLG. Figure 11.13a, b show the designing architecture of 3 × 3 PCLG and 4 × 4 PCLG respectively in QCA designer. The simulated result of 4 × 4 PCLG and 3 × 3 PCLG are shown in Fig. 11.14a, b respectively.

11.3.4 Comparative Study

The lossless conservative full adder design is compared with Fredkin gate, Toffoli gate in context of number of gates, number of garbage outputs. Table 11.9 gives the comparison in terms of number of CLG/RLG gate and number of garbage outputs for lossless full adder implementation. All the 13 standard functions can be



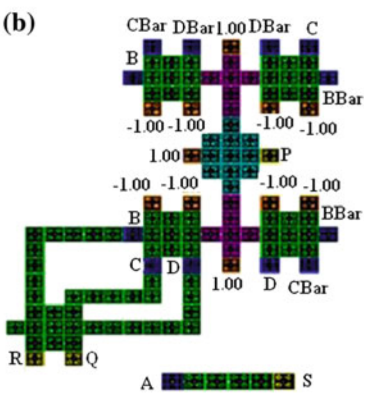
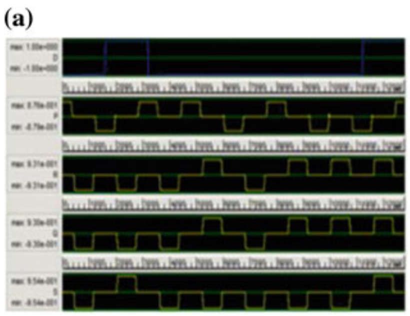


Fig. 11.13 **a** 3 × 3 PCLG QCA layout designed with QCA designer, **b** 4 × 4 PCLG QCA layout designed with QCA designer



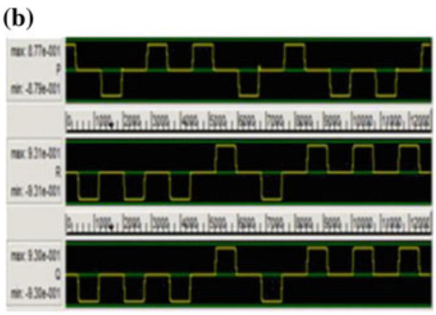


Fig. 11.14 **a** Simulation result of 4 × 4 PCLG, **b** simulation result of 3 × 3 PCLG

Table 11.9 Comparison of lossless full adder design

	No of gates	Garbage output
Fredkin gate	5	4
Toffoli gate	4	2
In [33]	2	1
In [34]	1	2
In PCLG [31]	1	0

implemented by the conservative full adder PCLG. Table 11.10 shows that to implement 13 standard functions [30] Fredkin uses total 41 gates with 136 clocks where as the present work shows that using 4 × 4 PCLG those 13 functions can be implemented with only 21 gates and 26 clocks, which produce 50 % reduction in terms of number of gates and 80 % reduction in clocking zones.

Table 11.10 Conservative logic design of 13 standard functions

	Standard functions	#Fredkin	#CLK	#4X4P CLG	#CLK
1	F = ABC	2	8	2	2
2	F = AB	1	4	1	1
3	F = ABC + AB′C′	3	12	1	3
4	F = AB + BC	2	8	2	3
5	F = AB + A′B′C	5	16	2	2
6	F = AB + A′B′C′	4	12	2	3
7	F = ABC + A′BC′ + AB′C′	6	16	1	3
8	F = A	1	4	1	1
9	F = AB + AC + BC	5	16	1	1
10	F = AB + AB′	1	4	2	2
11	F = A′B + BC + A′B′C′	6	16	4	3
12	F = AB + A′B′	2	8	1	1
13	F = ABC + A′B′C + AB′C′ + A′BC′	3	12	1	1
	Total	41	136	21	26

11.4 Ternary QCA Cell and Ternary Logic Implementation

Now it is established fact that quantum dot cellular automat (QCA) can be an alternative of promising and empirical CMOS technology. Researchers are mostly concentrating on binary logic representation in QCA, while it may not be suitable to think all aspects by means of binary logic. Multivalued logic especially ternary logic has potential advantages like greater data storage capability, faster arithmetic operations, better support for numerical analysis, application of nondeterministic and heuristic procedures, use of communication protocols and better effective solution for nonbinary problems [11–18]. QCA platform is being utilized for ternary logic implementation [11–18]. It is really in infancy stage, several more research works need to establish ternary logic implementation. Lebar Bajec et al. introduced ternary QCA (tQCA) and reported in [12]. The tQCA can also be classified into semiconductor-based tQCA and metal island tQCA like bQCA. The tQCA cell is realized by means of eight QDs confined within a square-shaped cell and two extra electrons confined within the cell. Eight QDs are arranged in a ring shape within square cell as shown in Fig. 11.15a. tQCA can have five state polarization ‘-1.00,’ ‘$+1.00$,’ ‘1/2,’ ‘1/2’ and no polarization represented by ‘A,’ ‘B,’ ‘C,’ ‘D,’ and ‘N’ respectively and are shown in Fig. 11.15b. Details of tQCA have been discussed in Sect. 11.4.1. Quantum mechanical model to adiabatic switching, the cell-to-cell interaction and some elementary logic gate implementation as straight wire, inverter, and majority voter have been reported in [13, 14]. Material choice to fabricate the device as well as architectural/circuit proposal for design the tQCA cell is essential requirement for fabrication of device. To establish the tQCA as a logic design

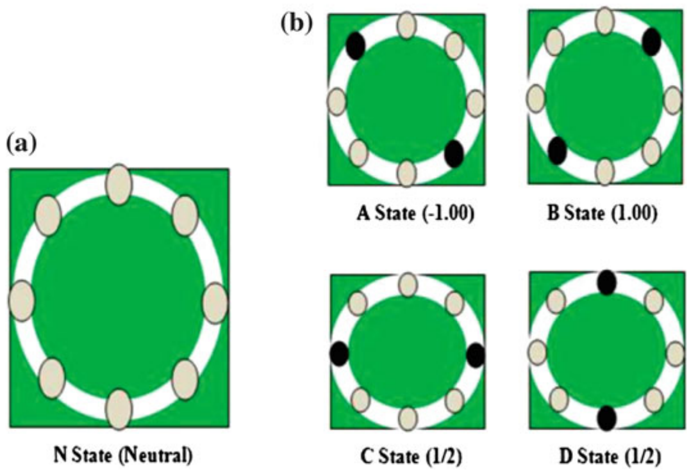


Fig. 11.15 **a** Eight quantum dot arranged in ring within square-shaped tQCA cell shows no polarization state and **b** tQCA cell with 'A' state (−1.00), 'B' state (+1.00), 'C' state, or 'D' state (1/2)

paradigm, the attention toward fabrication for this device is necessary. It also to be noted that the limitations for such device design are temperature as well as several physical parameters for material, size of cell, quantum dot size, etc., are obviously a challenging aspect toward design of the device. Rest of the section is organized in the following subsequence; Sect. 11.4.1 is dedicated to explore the basic ternary logic and tQCA, quantum mechanical model to study the different parameter that must be taken care during fabrication of tQCA cell.

11.4.1 Ternary Logic and tQCA

A quantum dot cellular automata cell is configured with four QDs made with semi-conductor or metal island or molecular redox. The four dots are positioned at the corner of a square-shaped cell. The two extra electrons are confined within the cell. The tunnel junction between two dots plays a role for tunneling electron from one dot to another dot. It is a bi-stable system; the single electron can tunnel from one QDs to neighbor QDs. Practically, each QD can have 'M' number of electrons and it is considered that there will be two extra electrons within a cell. Therefore, total number of electrons present in the system is $(4M+2)$. In bQCA cell, initially the inter dot barrier is low, it is known as 'Relax' phase. Due to stimulus or clock interaction inter dot barrier gets high and electron tunnel through the tunnel junction and attend a definite polarity known as switch phase. In the hold phase, the dots hold polarity for a definite time. Finally, in release phase dots release the polarity. These are the four clocking phase in bQCA.

The tQCA cell differs from bQCA cell in number of QD present in the cell and polarization state. There are eight numbers of QD arranged in a ring/circular shape within the square-shaped cell, shown in Fig. 11.15. The diameter of ring must be equal to length of a square-shaped cell. As a result R = L/2, where 'R' is radius of ring/circle and 'L' is length of square-shaped cell. Therefore, maximum distance of separation between two QD is L. The two extra electrons similar to bQCA are confined within the cell. The internal circuitry of tQCA cell is made of tunnel junction and junction capacitance. Due to coulomb interaction, the tQCA cell is being polarized. Five possible state of polarity marked as 'A,' 'B,' 'C,' or 'D' and 'N' are shown in Fig. 11.15.

11.4.1.1 Quantum Mechanical Model

Quantum dot is a charge container, it can hold 'M' number of electrons and it has three-dimentional confinement. The charge on each QD is quantized and equals to 'Me' where 'e' is electron charge in ev. Now, between two QDs, when tunneling occurs by amount 'e,' associate change in energy can be expressed in terms of capacitance 'C' between two QDs. The extra charge 'e' changes the electrostatic potential by $E_C = e^2/C$. The tunneling time 'τ' can be expressed as $\tau = R_t, C$, where R_t is the resistance of tunnel junction. It can be calculated from Heisenberg uncertainty relation

$$\Delta E\tau = (e^2/C)R_tC\rangle h \tag{11.4.1}$$

which implies $R_t \gg h/e^2$, i.e.,

$$R_t \gg R_k \tag{11.4.2}$$

The tunnel resistance must be greater than Quantum Hall resistance, i.e., $R_t \gg 25.813\,\mathrm{K\Omega}$ which is known as Von Klitzing constant. The capacitance C_a can be expressed with respect to ε_r and radius of QDs as

$$C_a = 8\pi\varepsilon_0\varepsilon_r R \tag{11.4.3}$$

where R is radius of QD and ε_0, ε_r represent permittivity of free space and relative permittivity respectively.

To calculate the radius of QD to form tQCA cell at room temperature, i.e., at 300 K, three temperature regimes are to be considered.

(i) $e^2/C \ll K_BT$ where discreteness of charge cannot be discerned.
(ii) $\Delta E \ll K_BT \ll e^2/C$ in classical coulomb blockade regime.
(iii) $K_BT \ll \Delta E \ll e^2/C$ in quantum coulomb blockade regime, where only few levels participate in transport.

Now, the necessary condition for the correct behavior of multiple (eight) quantum dot system is

$$K_B T \ll (3/8)\frac{h^2}{m_e L^2}$$

As a result, we have 'L,' the radius of quantum dot as

$$L = \left[(3/8)\frac{h^2}{m_e K_B T}\right]^{1/2} \tag{11.4.4}$$

The tunneling rate of tQCA cell system can be expressed as

$$\Gamma_{\pm} = \frac{\pm\Delta E}{e^2 R_T(\exp\frac{\pm\Delta E}{K_B T} - 1)} \tag{11.4.5}$$

where ΔE, particle level spacing can be expressed as $\Delta E = h/\tau$,

R$_t$ is tunneling resistance. The tunneling energy can be calculated in ev as

$$\gamma = \frac{\Delta E}{2} \tag{11.4.6}$$

The polarization on single tQCA cell can be derived from

$$\Gamma_{\pm} = \frac{\Gamma_{\mu}}{\Gamma_{+} + \Gamma_{-}} \tag{11.4.7}$$

The cell-to-cell response function is defined as

$$P_2 = \frac{2}{1 + \{\beta P_1 + \sqrt{\beta^2 P^2 + 1}\}^2} - 1 \tag{11.4.8}$$

where P_1 is the polarization of first cell and

$$\beta = e^2/4\pi\varepsilon_0(2 - \sqrt{2})/4(\gamma L)^{-1}$$

or

$$\beta = \frac{0.2109}{\gamma L} \tag{11.4.9}$$

Here, 'L' is length between two dots in nm. Now, differentiating to find out the slope of cell-to-cell response function at origin

$$\frac{dP_2}{dP_1}\,|_{P_1=0} = -\beta \tag{11.4.10}$$

So, the saturation value can be expressed as

$$P^{sat} = \frac{2}{1 + \{\sqrt{(\beta^2 + 1)} - \beta\}} - 1 \tag{11.4.11}$$

In tQCA, the good switching is obtained for $\beta > 1$. To meet this requirement, the exhaustive calculations on different materials are performed and the result obtained from the MATLAB program. There may be several choices of materials; we concentrated on only GaAs, Ge, and Si and the calculations have been performed on those materials only. The result obtained is shown in Figs. 11.16, 11.17, 11.18 and 11.19, which provide detailed knowledge about the fabrication of tQCA cell. In the Fig. 11.16, it is shown that for Si, the polarization of tQCA cell effect due to distance of separation between two QDs for temperature range 300–285 K. Physically 5 nm distance of separation may be adopted during fabrication for 6.36 nm radius of QDs. The change in tunneling energy and tunneling time for diverse temperature are explored in Figs. 11.17 and 11.18. As a result, it is concluded that better switching can be obtained with silicon (Si) but the quantum dot size has to decrease if working temperature is increased [36].

The capacitance of corresponding quantum dot will decrease simultaneously which is responsible for increasing the charging energy. Since, energy uncertainty or energy of level splitting of quantum dot must lie between the charging energy and thermal energy, there is a need to choose a suitable tunneling resistance to solve the problem. It is observed that the tunneling resistance decreases with the increase in temperature. The tunneling rate also increases with particle level spacing (ΔE). In case of switching, the tunneling energy and the length between two dots play an important role. To get better switching, the term β should be greater than 1. In case of Si, at temperatures 3 and 30 K, 20 nm length (L) between the dots gives β

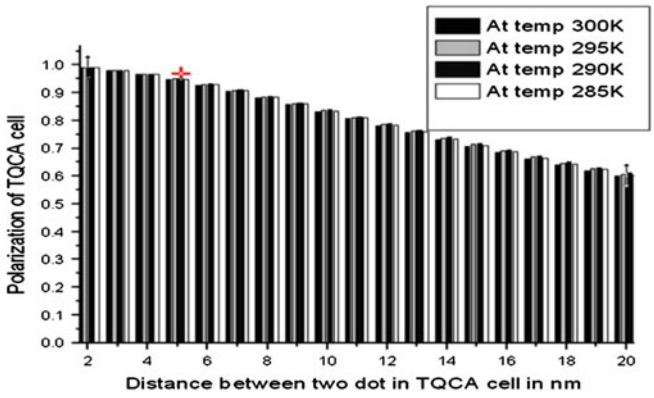


Fig. 11.16 Effect of polarization due to distance of separation between two dots in tQCA cell at 300, 295, 290, and 285 K for Si. '+' sign denotes that the polarization obtained for 5 nm distance of separation between two dots

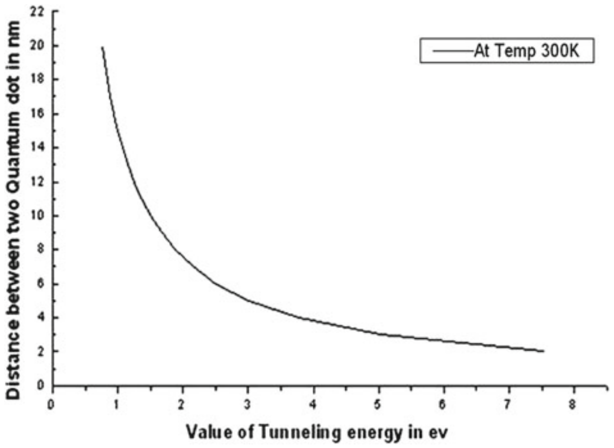


Fig. 11.17 Change in tunneling energy due to distance of separation between two dots in tQCA

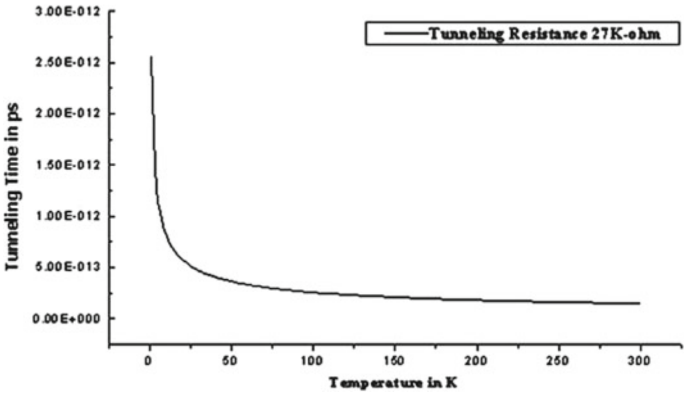


Fig. 11.18 High temperature effects on tunneling time

values greater than 1 (27.5 and 5.19 respectively) and the value of Psat is quite good. However, at 300 K, the length 20 nm gives $\beta = 0.77$, hich is less than one. So L $=$ 5 nm is taken and got the value of $\beta = 3.01$ along with $P_{sat} = 0.9491$.

For Ge and GaAs the charging energies (16.05×10^{-3} ev and 6.75×10^{-3} ev respectively) of dot at 300 K become less than the thermal energy, so the discreteness of charge cannot be discerned. It is concluded that for tQCA switching, Ge and GaAs should not be used. These can only be used at a lower temperature. Using Si at 300 K, the experiment of tQCA switching may be accomplished if the tunneling resistant is just greater than the Quantum Hall resistant.

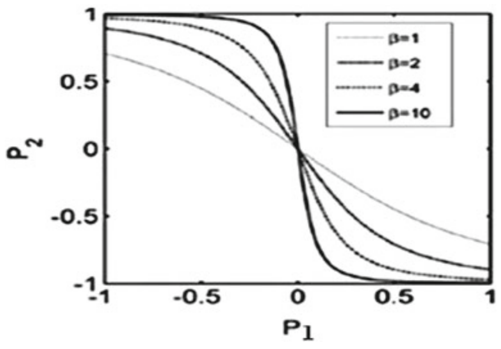


Fig. 11.19 Cell-to-cell response function for different values of $\beta = 1, 2, 4, 10$

11.5 Results and Discussions

In this chapter, the models for physical realization of semiconductor-based tQCA have been discussed. Detailed guidelines for material choice required for fabrication of tQCA cell as well as diverse parameter choice have also been presented. In Sect. 11.4.1.1, quantum mechanical approaches to solve the numerical problem have been described. The result indicated the limitation that the tunneling resistance must be greater than quantum hall resistance, i.e., Rt $\gg$ 25.813 kΩ. The operating temperature is room temperature if the dot size is less than 6.6 nm and distance of separation between two dots must be 5 nm. Otherwise, the device has to operate at low temperature that will require high cooling system. Silicon can be the choice to fabricate the device. The dot size must be such that it can have definite tunneling time, which also affects the polarization. Value of β must be greater than one for better switching phenomenon. As a result, the best choice for diverse temperature domain is silicon (Si). Design of tQCA cell may also be promising step toward the tQCA cell fabrication. It will open a strong alternative to represent the information in ternary logic and to design the arithmetic logic circuits for ternary computing with quantum dot cellular automata.

In Sect. 11.3, presents a new lossless and effective architecture of full adder using conservative logic design. A new reversible logic TRLG is demonstrated for online testing of the advanced PCLG. In this work, a conservative full adder is designed to optimize the garbage output. The online testing strategy is also described with an online tester TRLG gate. This tester can also be applicable to test online any 2×2, 3×3, 4×4 conservative logic gate. Finally, this work can conclude that zero garbage count, testable lossless conservative full adder design can be applicable to implement advanced cryptographic architecture.

Hence, it can be concluded that QCA is a promising alternative of CMOS technology in the era of ‘More-than-Moore’ technological trends. The computational intelligence of this emerging technology is also proved by different research activities in micro–nano electronics development. QCA is in infancy stage of research, more research work is necessary for the development of QCA as a paradigm shift in this 'More-than-Moore' technological trends. A few burning problem is listed below, which are the open challenge to the researcher in the development of QCA as paradigm shift. Input and output device is a fundamental requirement for computational device design. No such improvement is noticed so far; it can be a good problem domain for research. Tri-state buffer is hardly possible in QCA design. Hence, a challenge remains to the researcher to find out a solution for this problem. Crossover in QCA device or circuit design is quite natural problem in any 2D fabrication. Coplaner crossover with single clock and multiple clocks are recognized as a solution of crossover problem, but clock conflict problem also arise in large circuit design. Design of robust and reliable QCA circuit design is expected since QCA device fabrication is in nanoscale. Therefore, quantum dot cellular automata is a promising research area and a paradigm in the revolution of advanced computing technology, an alternative in the 'More-than-Moore' technological trends.

References

1. Moore, G., et al.: Cramming more components onto integrated circuits. Electronics **38**(8), 35–39 (1965)
2. International Technology Roadmap for Semiconductors: Semiconductor industries as association, San Jose, CA. http://public.itrs.net (2001)
3. Lent, C.S., et al.: Quantum dot cellular automata. Nanotechnology **4**, 49–57 (1993)
4. Tougaw, P.D., et al.: Dynamic behavior of quantum cellular automata. J. Appl. Phys. **80**(8), 4722–4736 (1996)
5. Lent, C.S., et al.: Bistable saturation in coupled quantum dots for quantum cellular automata. Appl. Phys. Lett. **62**, 7–14 (1993)
6. Timler, J., Lent, C.S.: Power gain and dissipation in quantum dot cellular automata. J. Appl. Phys. **91**(2), 823–831 (2002)
7. Tougaw, P.D., et al.: Logical devices implemented using quantum cellular automata. J. Appl. Phys. **75**(3), 1818–1825 (1994)
8. Amlani, I., et al.: Experimental demonstration of electron switching in a quantum-dot cellular automata (QCA) cell. Superlattices Microstruct. **25**(1–2), 273–278 (1999)
9. Amlani, I., et al.: Digital logic gate using quantum-dot cellular automata. Appl. Phys. Lett. **74**, 2875 (1999)
10. Macucci, M., et al.: A QCA cell in silicon on insulator technology: theory and experiment. Superlattices Microstruct. **34**, 205–211 (2003)
11. Bajec, L., et al.: Towards the bottom-up concept: extended quantum-dot cellular automata. Microelectron. Eng. **83**(4), 1826–1829 (2006)
12. LebarBajec, I.L., et al.: The ternary quantum-dot cell and ternary logic. IOP Nanotechnol. **17**(8), 1937–1942 (2006)
13. Pecar, P., et al.: Solving the ternary QCA logic gate problem by means of adiabatic switching. Jpn. J. Appl. Phys. **47**(6), 5000–5006 (2008)
14. Pecar, P., et al.: Adiabatic pipelining: a key to ternary computing with quantum dots. IOP Nanotechnol. **19**(49), 495401 (2008)
15. Bajec, I.L., Pecar, P.: Two-layer synchronized ternary quantum-dot cellular automata wire crossings. Nanoscale Res. Lett. **7**, 221 (2012). doi:10.1186/1556-276X-7-221
16. Primoz, P., Bajec, I.L.: The key elements of logic design in ternary quantum-dot cellular automata. In: Calude, C.S. et al. (eds.) Unconventional Computation, pp. 177–188. Springer, Berlin (2011)

17. Pecar, P., et al.: Solving the ternary QCA logic gate problem by means of adiabatic switching. Jpn. J. Appl. Phys. **47**(6), 5000–5006 (2008)
18. Pecar, P., et al.: Adiabatic pipelining: a key to ternary computing with quantum dots. IOP Nanotechnol. **19**(49), 495401 (2008)
19. Vikramkumar, P., Sridharan, K.: Low complexity design of ripple carry and Brent-Kung adders in QCA. IEEE Trans. Nanotechnol. **11**(1), 105–119 (2012)
20. Keren, I.: Computer Arithmetic Algorithms. AK Peters, Ltd., Natick (2002)
21. Kharbash, F., Chaudhry, G.M.: The design of quantum-dot cellular automata decimal adder. In: IEEE International Multitopic Conference, INMIC (2008)
22. Gladshtein, M.A.: The signal propagation delay reduction of the combinational adder of decimal digits encoded by the Johnson-Mobius code. Autom. Control Comput. Sci. **44**(2), 103–109 (2010)
23. Gladshtein, M.: Quantum-dot cellular automata serial decimal adder. IEEE Trans. Nanotechnol. **10**(6), 1377–1382 (2011)
24. Podder, D., et al.: Realization of bi-quinary coded decimal adder in quantum dot cellular automata. Emerging Trends in Computing and Communication, pp. 353–361. Springer, India (2014)
25. Oklobdzija, V. (ed.): The Computer Engineering Handbook. CRC Press, Boca Raton (2002)
26. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**(3), 183–191 (1961)
27. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**(6), 525–532 (1973)
28. Feynman, R.: Quantum mechanical computers. Opt. News **11**, 11 (1985)
29. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theor. Phys. **21**, 219 (1982)
30. Das, K., De, D.: Characterization, test and logic synthesis of novel conservative and reversible logic gates for QCA. Int. J. Nanosci. **9**(03), 201–214 (2010)
31. Dey, A., et al.: Online testable conservative adder design in quantum dot cellular automata. In: Emerging Trends in Computing and Communication, pp. 385–393. Springer, India (2014)
32. Ma, X., et al.: Testing reversible 1D arrays for molecular QCA. In: 21st IEEE International Symposium on Defect and Fault Tolerance VLSI Systems (2006)
33. Vasudevan, D.P., et al.: Reversible-logic design with online testability. IEEE Trans. Instrum. Meas. **55**, 2 (2006)
34. Thapliyal, H., Nagarajan, R.: Reversible logic based concurrent error detection methodology for emerging nanocircuits. In: 10th IEEE Conference on Nanotechnology (IEEE-NANO) (2010)
35. Walus, K.: ATIPS laboratory QCA designer homepage. ATIPS Laboratory, University Calgary (2002)
36. Das, K., et al.: Realization of semiconductor ternary quantum dot cellular automata. Micro Nano Lett. **8**(5), 258–263 (2013)

Chapter 12
Smart Videocapsule for Early Diagnosis of Colorectal Cancer: Toward Embedded Image Analysis

Quentin Angermann, Aymeric Histace, Olivier Romain, Xavier Dray, Andrea Pinna and Bertrand Granado

Abstract Wireless capsule endoscopy (WCE) enables screening of the gastrointestinal tract by a swallowable imaging system. However, contemporary WCE systems have several limitations—battery, low processing capabilities, among others—which often result in low diagnostic yield. In this chapter, after a technical presentation of the components of a standard WCE, the authors discuss the related limitations and introduce a new concept of smart capsule with embedded image processing capabilities based on a boosting approach using textural features. We discuss the feasibility of the hardware integration of the detection–recognition method, also with respect to the most recent FPGA technologies.

12.1 Introduction

12.1.1 Colorectal Cancer and Related Imaging Technics

Colorectal cancer (CRC) is the first cause of death by cancer in developed countries, with an estimated incidence of 728.550 cases worldwide in 2008, with fatal outcome in 43 % of cases. Overall, CRC is the third more frequent cancer after lung cancer and

Q. Angermann · A. Histace (✉) · O. Romain
ETIS UMR CNRS 8051, ENSEA-University of Cergy-Pontoise,
6 Av. du Ponceau, 95000 Cergy, France
e-mail: aymeric.histace@u-cergy.fr

Q. Angermann
e-mail: quentin.angermann@u-cergy.fr

O. Romain
e-mail: olivier.romain@u-cergy.fr

X. Dray
ETIS Lab and APHP Hôpital Lariboisière, Paris Diderot, Paris 7 University, Paris, France

A. Pinna · B. Granado
LIP6 Lab, University Pierre Et Marie Curie, CNRS, Paris, France

M. Fakhfakh et al. (eds.), *Computational Intelligence in Digital and Network Designs and Applications*, DOI 10.1007/978-3-319-20071-2_12

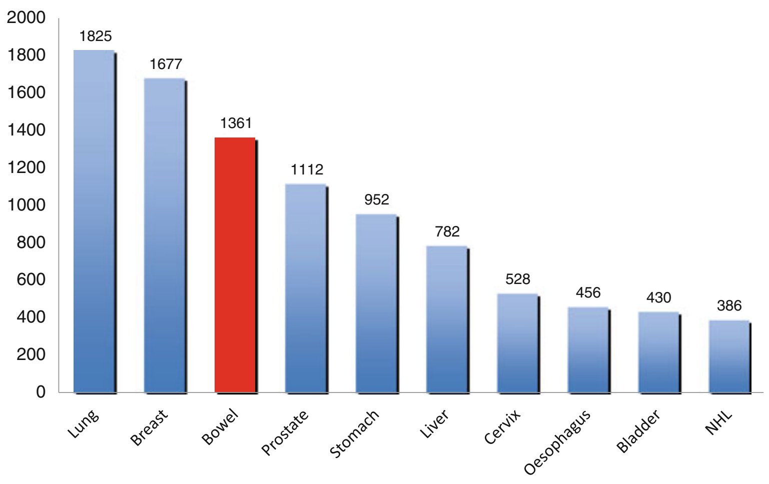


Fig. 12.1 Worldwide Cancer Incidence (expressed in thousands) in 2012. *Source* World Health Organization

breast cancer [1] (see Fig. 12.1 for comparisons with other type of cancer in 2012). Prevention of CRC by detection and removal of preneoplastic lesions (colorectal adenomas) is therefore of paramount importance and has become a worldwide public health priority.

Currently, colonoscopy is the the "gold standard" technique for diagnosis of colorectal adenoma and cancer. Using a videoendoscope, gastroenterologists can perform and record a complete examination of the colon in order to detect and to remove suspicious tissular structures like adenomas which degenerescence could lead to cancer. Because colonoscopy is performed under general anesthesia, mini-invasive techniques such as computed-tomography-based colonography and wireless capsule endoscopy (WCE) have been developed. Both techniques are currently considered valid alternative options to videocolonoscopy in patients with contraindication or low compliance to general anesthesia. WCE takes form of a pill equipped with a CCD or CMOS sensor, two batteries, and a RF (radiofrequency) transmitter, that enables the wireless identification of gastrointestinal abnormalities such as ulcers, blood, and polyps [2] with no need for hospitalization or sedation. In the last decade, WCE has become a breakthrough technology for diagnosis of small bowel pathologies [3]. Many fabricants such as Given Imaging, IntroMedic, and Olympus [4] have developed a variety of capsules for the complete examination of the gastrointestinal tract.

Practically speaking, after ingestion of the capsule, about 150,000 images are captured along the digestive tract and each of them are wirelessly transmitted to a wearable receiver and saved for a postponed physician's reading. Offline image processing enables the identification of gastrointestinal abnormalities (like the aforementioned polyps and adenoma) by the gastroenterologist.

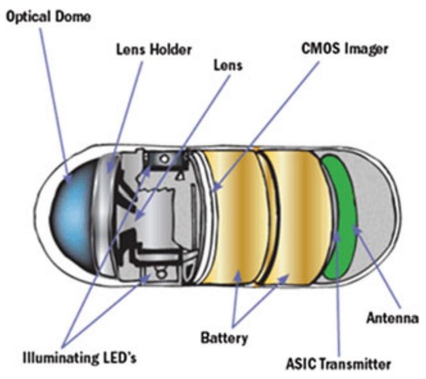


Fig. 12.2 Images and diagram showing the composition of a video capsule endoscope. References: Providence Park Hospital, Detroit, USA

In the following section, details about the technology of WCE are proposed to have a better understanding of the current issues related to this imaging device that explain its limited used in practical daily routine.

12.1.2 WCE Technology in Details

Figure 12.2 shows the usual classic component layout of WCE.

12.1.2.1 Battery

As it can be noticed, the battery is the element that occupies majority of available space in the capsule. The current autonomy is between 8 and 10 h by exam. Nevertheless, the average processing time from the mouth to the anus is more than 12 h, and, depending on the organism of the patient, sometimes even 24 h; under these conditions the capacity of this battery will be 55 mA @ 12h, a very low one to supply enough power to the capsule. Currently the two batteries used in the WCE are composed of silver oxide, the only material approved for clinical use, even if it is not the most efficiently solution. Improvements are quite limited in this area: If other batteries were used, lithium batteries can provide longer operating times or carbon nanotubes that will produce a good autonomy at the same time reduce the required space, the energy performance will be better. As an example, batteries composed by Zinc Air (ZnO_2) will increase about four times the capacity and the weight will be reduce in 20 %.

12.1.2.2 Image Sensor

Both CMOS and CCD sensors are used in WCE. The main characteristics of these technologies are:

- CMOS (Complementary Oxide Silicone): A CMOS imager converts the charge into voltage within each pixel, using an array of pixels to convert light into electronic signals. This signal is weak and needs amplifying to an usable level, and then each pixel has its own amplifier circuit. The result is a lower chip count, an increased reliability, a reduced power consumption, and a more compact.
- CCD (Charge Coupled Device): The CCD imager transfers the charge from the pixel, created by photoelectric conversion, through a bucket relay transfer to the imager output stage. The charge transfer is almost complete, which means that noise is rare, but a high voltage differential is required to improve transfer efficiency, which increases power consumption.

The most recent WCE of Given Imaging company uses a CMOS sensor "Aptina MT9S526".

12.1.2.3 Transmission

The transmission is one of the most important but consuming element of the capsule, because it has to process about at least 150,000 images per exam (often more), this module is nearly 100 %-active during the all exam. Currently, the capsules could use either one of these systems for image transmission to the wearable data logger:

- By RF transmitter (Zarlink ZL70340 E3 for instance): composed of an oscillator and a control circuit that send images with a variable rate 800/400/200 kbps and a carrier frequency of 402–405 MHz, the nominal current is 5 and 1 mA in passive mode.
- By electric field propagation: this system is a more experimental one used on the MIRO capsule [5]. Based on a novel human telemetry technology known as electric field propagation, it uses the human body as a conductive medium for data transmission. The transmitters are a pair of gold plates coated on the surface of the capsule, reducing drastically the power consumed compared with existing communication devices that use RF transmission technology. Thanks to a longer operation time, this capsule takes advantage of the surplus energy to produce more image data; the autonomy of this capsule is between 10 and 12h but remain barely used by clinicians.

12.1.2.4 Power Consumption of WCE

Energy consumption has become a very important performance in a wide range of electronic systems, from embedded systems based on batteries to smart sensor as WCE for health applications. With the increasing size and complexity of these systems, it is imperative to take into account the energy consumption aspect from the start of the design process to avoid a lower autonomy. While Modeling at system

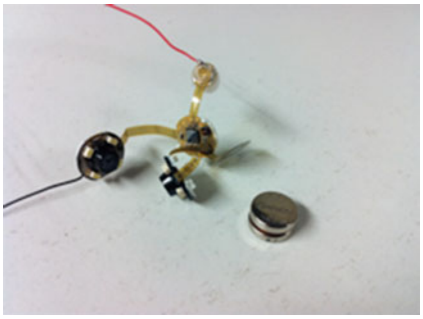


Fig. 12.3 Layout of a PillCam Small Bowell 2 composed of two CMOS sensors and a RF Zarlink chip. Two batteries supplies the WCE in energy

level, the power consumption represents a key issue that will permit to optimize the architecture with respect to the specifications. Both for current and future intelligent WCE, a power consumption model would allow finding solutions for optimizing the RF transmission, as for example, by filtering the data transmitted. However, modeling the power consumption at the system level is difficult since the details of the architecture (CMOS sensor, LEDs, processor, RF transmitter, etc. see Fig. 12.3) are not yet gave in the literature. In addition, the lack of information on the layout and the manufacturing process makes the analysis of the interconnection and the static power consumption impossible.

In the last decade, numerous methods were proposed for the power consumption estimation of mixed components (processors, memories, digital IP, and RF system) [6–13]. These methods can be divided into two classes: The analytical methods for which the power consumption is often modeled as a mathematical function depending on the size of the circuit and its switching activity. In empirical methods, the measurement of power consumption is associated with each instruction or couple of instructions [14, 15] associated to the processor, or, to the activity of the system [16]. These power consumption models, at system level, are widely used for the power consumption estimation of digital block [17–19].

Second type of methods appears more adapted to our problem. Thus, to model the power consumption of the WCE at system level, we used an empirical approach. The power consumption model estimates the instantaneous power consumption as a function of operating parameters, such as the operating state of the WCE. For Pillcam SB2, there are three operating states (see Figs. 12.4, 12.5, 12.6 for illustration of the instantaneous current consumption of each mode). The strategy is to measure the power consumption at these various operating states. Then the power consumption model is built based on a continuous function that interpolates these measures.

For these three operating modes, the instantaneous current profiles are similar. Just the acquisition-time is different. LEDs are power supplied with a stable tension of 3.1 V and the images are acquired by the CMOS sensor, and transmitted to the data logger of the patient wirelessly.

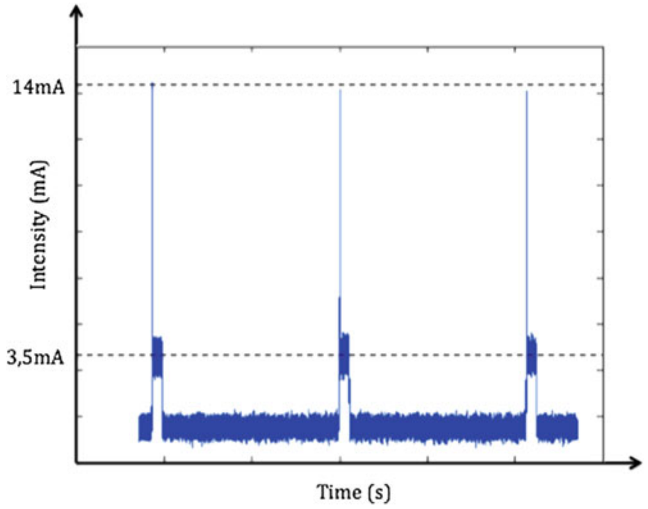


Fig. 12.4 Instantaneous current consumption (mA) for the slow operating mode (one image every 4 s)

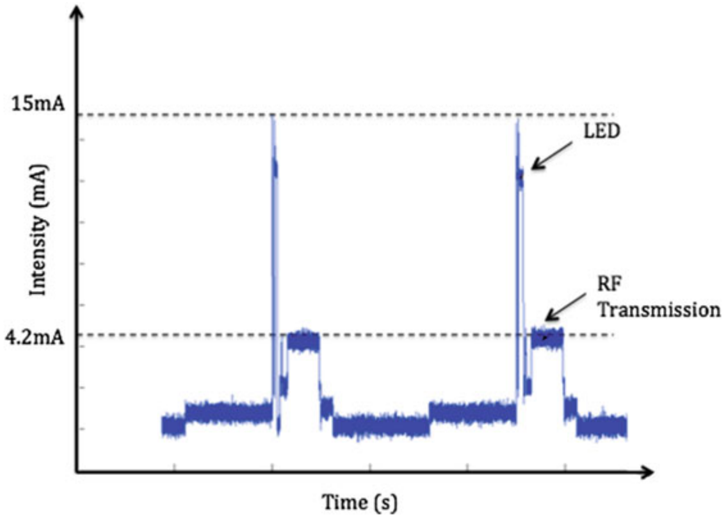


Fig. 12.5 Instantaneous current consumption (mA) for the medium operating mode (one image every 4 s)

During a standard exam, statistically the medium operating mode (4 images/second) is the most used by the WCE. The instantaneous power consumption is given in Table 12.1.

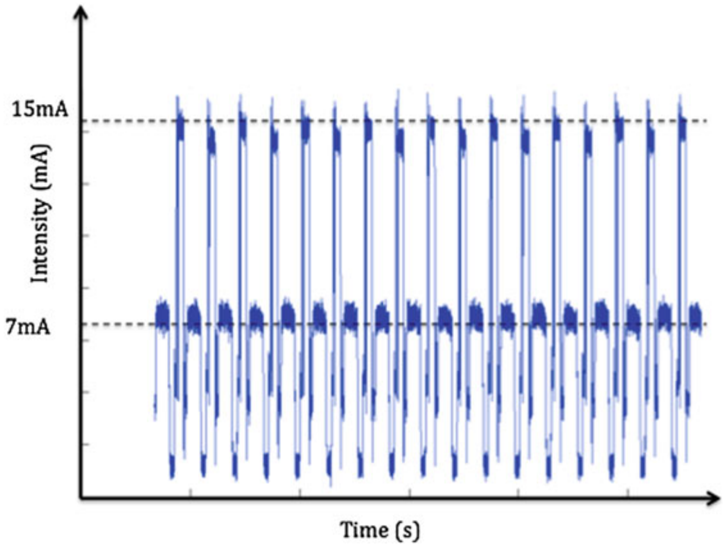


Fig. 12.6 Instantaneous current consumption (mA) for the speed operating mode (35 image every second)

Table 12.1 Detailed power consumption of a typical WCE

LEDs + CMOS (mA/s)	RF Transmitter (mA/s)	Total (mA/s)
11.00	76.92	**87.92**

Table 12.2 Technical comparisons of the existing WCE

	PillCam SB	EndoCapsule	MiroCam	Sayaka	OMOM	Vector
Size (mm)	26 × 11	26 × 11	24 × 11	23 × 9	28 × 13	26 × 11
Weight (gr)	3.7	3.8	3.3		6	
See Angle	140	145	150	360	140	
LED	4	6	6	4	6	4
Rate (fps)	2	2	2–3	30	0.5, 1, 2	19
Image Sensor	CMOS	CCD	CMOS		CCD	QVGA
Autonomy	8	8–10	9–11		8	5

12.1.2.5 Discussion About WCE

In Table 12.2, a comparison of the different existing technologies of WCE is proposed.
Current main issues of WCE are [4]:

- The complete analysis of the 150,000+ images is time consuming for physicians, and even for experienced ones, WCE diagnoses are sometimes challenging.
- The transmission of the 150,000+ images, that represents 80 % of the overall energy consumption of the embedded batteries, limits to 8 h the autonomy of the classic WCE, whereas 12 h are necessary to scan the complete intestinal tract.
- A recent study comparing diagnostic capabilities of videoendoscopy and of WCE shows that the average detection rate is around 80 % polyps per patient [3, 20]. Thus, the improvement of polyp detection and classification capabilities of WCE is expected from gastroenterologists.
- Processing capabilities of WCE are limited to transmit raw images. No "intelligence" is currently embedded into the imaging device itself.

In the context of early diagnosis of colorectal adenoma and cancer, a new generation of WCE must be proposed [21] (see Fig. 12.9 for illustration) that will permit an in situ detection of the polyps and, consequently, to only emit the images which are important for the final diagnosis. The expected benefits are twofold:

1. An increase of the battery lifetime up to 12 h considering the fact that, except for particular pathologies, only a small percentage of the 150,000 images would contain polyps and will be consequently transmitted (see Fig. 12.7).
2. A facilitated offline final diagnosis for the clinician considering the law amount of transmitted data after in situ hardware processing and the possibility to highlight particular regions of interest within the images that possibly contain a polyp.

In Fig. 12.8, a comparison between the usual clinical workflow and the expected one with proposed WCE is showed.

The main challenge related to this smart WCE is to embed the image processing scheme within the WCE itself. It requests low complexity algorithms that compelled to high hardware constraints in terms of computational resources. The performance has to remain high to satisfy the clinician expectations.

In the context of polyp detection for early diagnosis of CRC, in [21, 22], a first prototype demonstrator equipped with an active stereo vision sensor was proposed

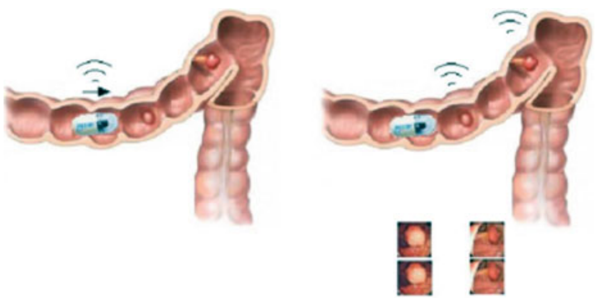


Fig. 12.7 Comparison between continuous transmission (*left*) and intelligent one (*right*)

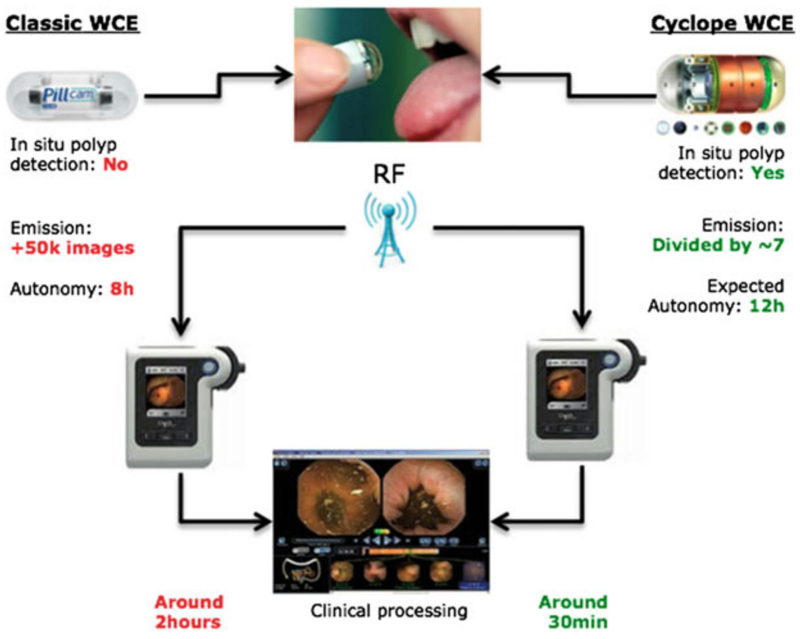


Fig. 12.8 Comparison of the classic clinical workflow (*left*) and the expected one using Cyclope WCE (*right*), with corresponding improvement in *green*

to detect protuberating polyps within the colon. The proposed embedded detection algorithm used a SVM classifier trained on robust 3D feature descriptors. The detection/recognition algorithms was embedded in a FPGA Virtex 5 and the overall detection performance was very promising with a global classification rate of 97 % on an in vitro dataset consisting of 111 polyps (40 adenomas and 81 hyperplasias) made in silicon. Nevertheless, it appears that for real case examinations, 3D features are not sufficient to detect the large variety of polyp shapes that can be very flat at an early evolution stage.

In this chapter, we focus on the 2D analysis of coloscopy images in order to investigate other possibilities than 3D shape characterization of polyps to improve capabilities of WCE. As in [22], a particular attention is given to propose a global detection/classification scheme that can be integrated into the "Cyclope"—WCE architecture shown in Fig. 12.9 and more precisely, by taking benefits of the FPGA block.

The remainder of this article is organized as follows: a state of the art on detection of polyps in videocolonoscopy using 2D features is proposed in Sect. 12.2. In Sect. 12.3, the proposed approach is detailed. Experimental results using algorithms implemented in C++ are given and commented in Sect. 12.4. In Sect. 12.5, a particular focus on hardware implementation (FPGA) is proposed, and conclusions are given in the last section.

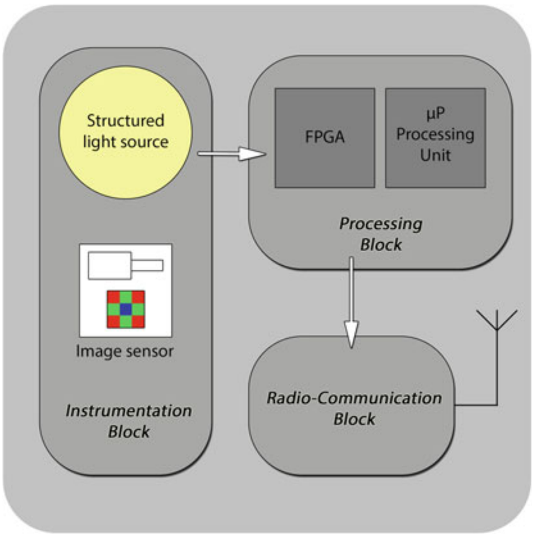


Fig. 12.9 Block Diagram of the “Cyclope WCE”

12.2 Related Works

Several previous references have considered the detection of intestinal polyps in videocolonoscopy images in the last few years ([23–28] among recent ones). They are mainly divided into two categories: based on geometric features of the polyps (size and shape) and based on textural features.

In the framework of “Cyclope project,” we focused our attention on four particular recent contributions.

In [24], Bernal et al. authors propose a study made on videoendoscopy images. They developed a region descriptor based on the depth of valleys (SA-DOVA). Resulting algorithm, divided into several steps, including region segmentation, region description, and region classification, is characterized by promising detection performance (see Table 12.3).

In [25], Figueiredo et al. assume that polyps show up as protrusions that can be detected using the local curvature of the image. Consequently, a method based on the mean and geometric curvature of the WCE image is proposed. The main drawback of the proposed approach is the strong dependance on the protrusion measure of the polyp to identify potential candidates. The consequence is that if a polyp is not protruding enough from the surrounding mucosal folds, it may be missed.

In [26], Karargyris and Bourbakis propose an algorithm for WCE images mainly based on Log Gabor filters and Susan edge detector. Based on the geometric information of the resulting detected ROI, a level-set segmentation is then initialized for an accurate delineation of the polyps. On the considered WCE image database (10 polyps and 40 non-polyps), the method gives satisfying results but authors highlight that taking into account of the texture- or color-based features within the detection/classification scheme would significantly increase related performance.

Table 12.3 Main characteristics of four of the most recent references of the literature

Authors	Main principle	Classification performance	Database
Bernal [24]	Geometry	Sensitivity 89 % Specificity 98 %	300 videocolonoscopy images containing a polyp (**freely available**)
Figueiredo [25]	Geometry	No indicated performance	17 WCE videos of 100 images each, containing example of polyps (10), flat lesions, diverticula, bubbles, and trash liquids
Karkargyris [26]	Geometry	Sensitivity 100 % Specificity 67.5 %	50 WCE images (10 polyps and 40 non-polyps)
Kodogiannis [27]	Texture	Sensitivity 97 % Specificity 94 %	140 WCE images (70 polyps and 70 non-polyps)

Finally, Kodogioannis and Boulougoura [27] propose a texture-based approach. Authors introduce a new texture-based features computed from the chromatic and achromatic spectra of the Region of Interest (ROI) that may contain a polyp. For classification, a neurofuzzy scheme is proposed. Main result is that the textural information is of first importance for the discrimination between polyps and non-polyps.

Table 12.3 summarizes the main principle and the obtained performance of these four main contributions.

All four presented approaches for polyp detection and classification are definitely of primary interest but may not fully compel to the hardware constraints of Cyclope architecture (the detection algorithm is to be embedded in the FPGA block of limited resources) since all developed methods were designed mainly for an offline use by the clinician and can fully benefit from the high computing capabilities of the last-generation processors: As a consequence, the related processing schemes include possible demanding algorithms like active contour segmentation [26], blob detector [24], or local curvature estimation [25], that have not been proved yet to be easily embedded on a "low" resource hardware like FPGA. Moreover, it also appears that image databases used for performance estimation are size-limited and/or not freely available for possible comparison, except in the case of [24], more particularly when considering WCE images.

Taking benefits of the aforementioned references, and taking into account the heavy hardware constraints of "Cyclope" WCE, a learning-based polyp detection approach using texture-based descriptors is introduced in next section. In order to compare related performance to the most recent literature, we will use for illustration the database freely provided by [24].

12.3 Proposed Method

12.3.1 Principle

The proposed method is inspired from the psychovisual methodology used by the physician when doing an endoscopic examination: First, a detection of the Regions of Interests (ROI) that may contain a polyp is performed using shape and size features extracted from the image. This first preselection allows a first fast scanning of the image. Once the ROI are detected, a second analysis based on texture (homogeneity, granularity, coarseness…) is achieved. Practically speaking, we propose a global scheme for the detection/classification of possible polyps divided into two steps:

1. Considering the geometric step of the proposed approach, simple image processing tools make possible the detection of circular/elliptical shape like the Hough transform for instance.
2. The texture-based classification is the main keypoint of the global scheme since the rejection of most of the false positive preselected ROI have to be performed at this stage. To achieve this, we propose to design an ad hoc classifier based on a boosting-based learning process using textural features.

The global scheme of this approach is summarized in Fig. 12.10. Each step is detailed in the following sections.

12.3.2 Geometric Features

As mentioned before, the first useful characteristics for detection are size and shape of the candidate structures. More precisely, a detection algorithm based on the circular form of the polyps is considered. Instead of considering a local curvature estimation

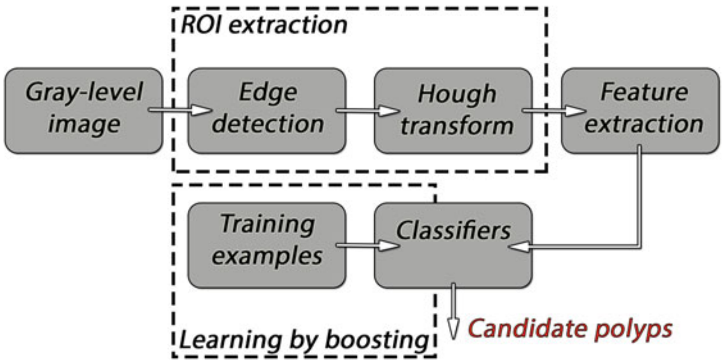


Fig. 12.10 Proposed diagram for the detection of polyps within videoendoscopy images

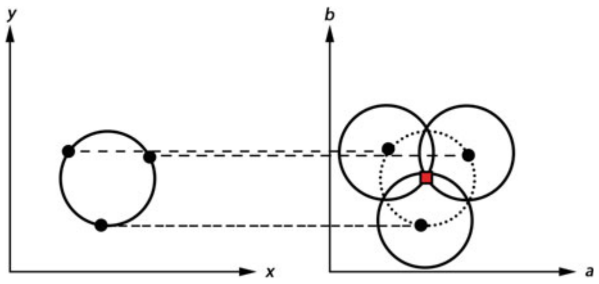


Fig. 12.11 Principle of Hough transform for circular patterns detection. On the *left*, three circular patterns in the image domain, on the *right*, these *three circles* corresponds to the *red dots* in the parameter domain (coordinates of the center (a, b) for a given value of radius R)

or the Log Gabor filtering, as suggested in [26], the circular Hough transform is used for three reasons; first, processing remains simple and efficient; second, all polyps must be detected even if numerous false positive ROI are also considered; third, the Hough transform can be FPGA embedded like shown in [29] for an *in situ* and real-time detection. A discussion on that particular point is provided in the related section. In order to handle with different polyp sizes, we consider a research interval for the radii of the extracted circles. Figure 12.11 illustrates the main principle of Hough transform for circular patterns.

12.3.3 Texture-Based Features

For the texture-based analysis of predetected ROI, the co-occurrence matrix [30] is used to discriminate textural patterns of polyps and non-polyps. Main advantage of co-occurrence matrix is in their fixed dimensions only depending on the grayscale resolution of images: as a consequence whatever is the dimensions of the candidate ROI, the size of the matrix remains the same, which is of first interest when considering the hardware implementation constraints (mainly memory) we have to deal with. Moreover, the textural discrimination capabilities of co-occurrence matrices remain of high efficiency even on grayscale images [31] and could be implemented on FPGA [32] with possible limited memory resource, the three-color channels being not necessary.

Basically, the co-occurence matrix $\mathscr{MC}_{\Delta x,\Delta y}(i, j)$ shows how often a pixel of gray-level value i occurs either horizontally, vertically, or diagonally to adjacent pixels of value j:

$$\mathscr{MC}_{\Delta x,\Delta y}(i, j) = \sum_{p=1}^{n}\sum_{q=1}^{m}\begin{cases}1, & \text{if } I(p,q) = i \text{ and } I(p+\Delta x, q+\Delta y) = j\\ 0, & \text{otherwise}\end{cases} \tag{12.1}$$

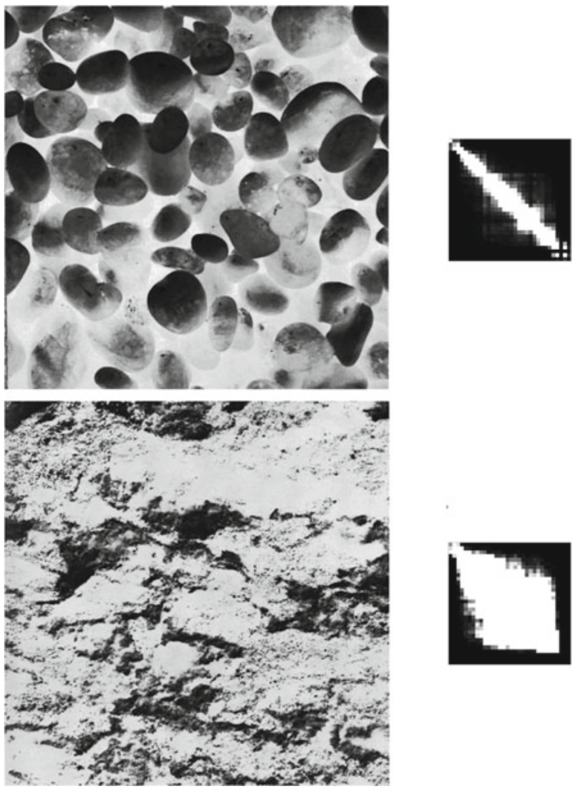


Fig. 12.12 Illustration of the computation of cooccurrence matrices for two different texture images

Figure 12.12 shows an illustration of two cooccurrence matrices obtained for two different kinds of texture (taken from the Brodatz databasis).

Size of the co-occurrence matrices is related to the grayscale resolution of input image. In Fig. 12.12, a subsampling from 256 to 64 was down to visually highlight the difference in the dispersion of the values around the diagonal of the matrix for the considered textures.

Twenty-six features (known as the Haralick's features [31]) are then extracted from each of the computed matrices. Are included: Contrast, Correlation, Entropy, Cluster Prominence, Cluster Shade, Dissimilarity, Homogeneity, Autocorrelation, Maximum probability, among other parameters. (see Eqs. (12.2)–(12.4) for illustration of the first three parameters).

$$Contrast = \frac{1}{K}\sum_{k=0}^{N-1} k^2 \sum_{|i-j|=k} \mathscr{MC}(i,j), \tag{12.2}$$

$$Correlation = \frac{1}{K\sigma_x\sigma_y}\sum_{i,j} ijMC(i,j) - \mu_x\mu_y, \tag{12.3}$$

$$Entropy = -\frac{1}{K}\sum_{i,j}\mathscr{MC}(i,j)\log\left(\frac{\mathscr{MC}(i,j)}{K}\right), \tag{12.4}$$

with K the number of elements of $\mathscr{MC}(i,j)$ and

$$\begin{aligned}
\mu_x &= \frac{1}{K}\sum_{i,j} i * \mathscr{MC}(i,j),\\
\mu_y &= \frac{1}{K}\sum_{i,j} j * \mathscr{MC}(i,j),\\
\sigma_x^2 &= \frac{1}{K}\sum_{i,j}(i-\mu_x)^2\mathscr{MC}(i,j),\\
\sigma_y^2 &= \frac{1}{K}\sum_{i,j}(j-\mu_y)^2\mathscr{MC}(i,j).
\end{aligned}$$

Since the texture-based classification is performed using a boosting-based algorithm, no limitations about the number of parameters is considered, main idea being to let the learning process converge to the best classification solution without any prior information.

12.3.4 Classification

"Boosting" is a machine-learning algorithm for supervised learning (see [33] among other publications of the same authors). It consists of the accumulation and constant learning of weak classifiers (a weak classifier is considered slightly correlated (a little better than chance) with the true classification), that once combined together generate a strong classifier, well-correlated with the ground truth provided by the expert. In the framework of our proposed approach, we use the boosting-based method of [34] setup in attentional cascade (Cascade Adaboost). This configuration allows us to create a strong classifier which performance can be priorly set up in order to optimize the sensibility of the classification along with the specificity. For illustration of the overall learning algorithm, see Fig. 12.13 in which F_i and D_i stand for the maximum authorized False Positive Rate and the minimum acceptable detection rate, respectively, computed for each iteration of the process using the given f and d performance ratio, and F_{target} the global false positive rate.

If the learning process related to boosting-based algorithms is time consuming, it is important to note that once the optimal classifier is computed offline, the classification step is very fast and fully compatible with a hardware implementation as shown by application to real-time face detection [34] embedded in cameras.

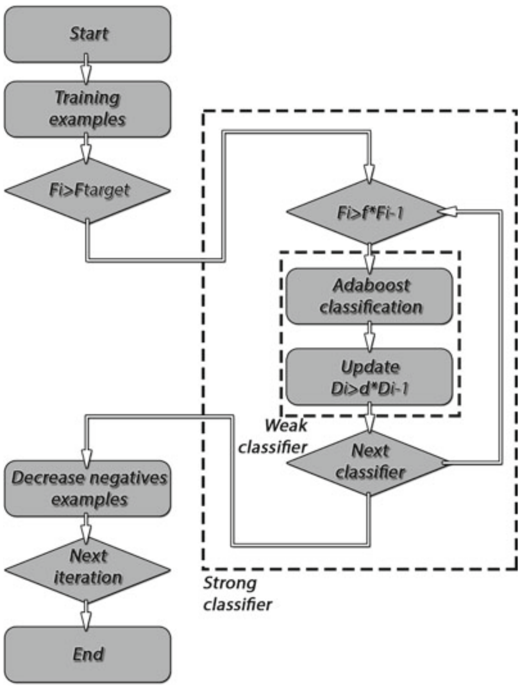


Fig. 12.13 Flow diagram shows how the Cascade Adaboost is performed. F_i and D_i stand for the maximum authorized False Positive Rate and the minimum acceptable detection rate, respectively, computed for each iteration of the process using the given f and d performance ratio, and F_{target} the global false positive rate

In our particular case, the considered weak classifiers are based on a set of truncated binary decision trees (bootstrapping) built from the 24 textural parameters on the dedicated learning database.

12.3.5 Data

Tests were performed using the database proposed by J. Bernal from the Universitat Autonoma de Barcelona [24], which consists of 300 videoendoscopy images presenting with one single polyp each identified and segmented by a specialist. The data are courtesy made available by authors. To our knowledge, in the particular framework of colorectal polyp detections, this is currently the only existing online database with a sufficient amount of examples to be statistically meaningful. Figure 12.14 shows some example of polyps extracted from the database.

To build the learning database, each image of the main dataset was subdivided into five thumbnails by the gastroenterologist, as shown in Fig. 12.15. A first ROI corresponds to the polyp (a), and the other four to non-polyps (b)–(e). The resulting learning/testing database is then composed of a total of 1500 images, with 300 images of polyps and 1200 images of non-polyps, the labeling being performed, once again, by a specialist.

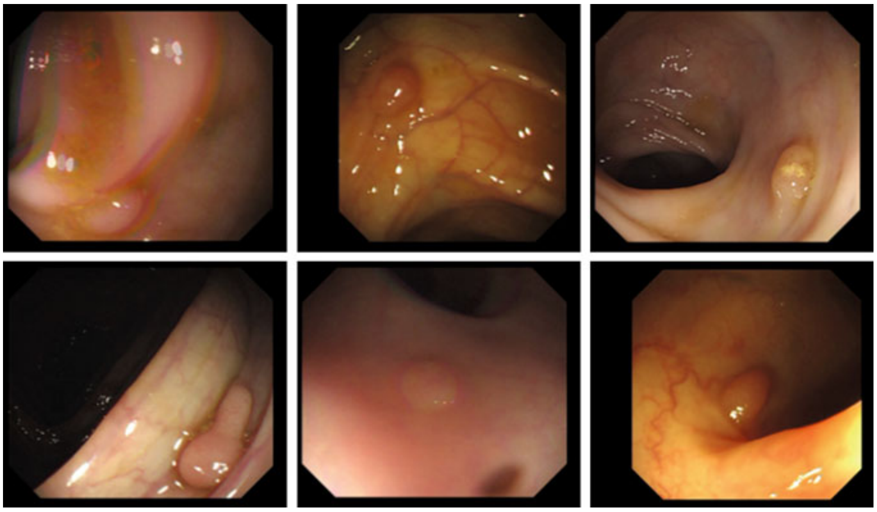


Fig. 12.14 Examples of polyps extracted from the database of [24]

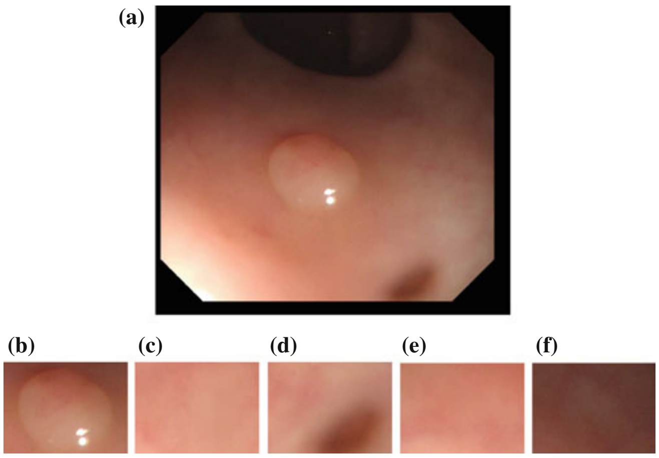


Fig. 12.15 Example on how the learning/testing database is generated from the original data of [24]: (**a**) Original image, (**b**) positive example, (**c**–**f**) negative examples

12.3.6 Performance Evaluation

To proceed to performance evaluation of the proposed boosting-based method, three measures are usually considered meaningful and complementary: the sensitivity, the specificity, and the false positive rate (FPR) respectively defined by

$$Sensitivity = \frac{TP}{TP + FN}\ , \tag{12.5}$$

$$Specificity = \frac{TN}{TN + FP}\ , \tag{12.6}$$

$$FPR = \frac{FP}{FP + TN}\ , \tag{12.7}$$

with TP, FN, TN, FP standing for true positive, false negative, true negative, and false positive, respectively.

12.4 Experiments

In order to compel with a real-time constraint analysis, all the image processing scheme previously prsented was implemented in C++ using the OpenCV library. The main objective is to show that the complexicity of the developed algorithm is in accordance with a possible hardware implementation and to develop a context-based library of functions that could be translated into VHDL codes using for instance HMS-Vivado-like pipeline.

Performance of each step of the process is proposed in the following sections.

12.4.1 ROI Detection Using Circular Hough Transform

In Table 12.4 the detection performance of the Hough transform on the aforementioned original database of [24] is shown and compared to the Log Gabor filtering proposed by [26].

We provide here the best obtained results considering the sensibility rate for an ad hoc setup of the Hough transform circle detection threshold and for a research interval of the radii between 40 and 80 pixels.

We do not provide here usual Receiving Operating Curve (ROC) since we do not control the number of detected FP for a given threshold: Depending on the quality of the original image, number of FP can be very important (see Fig. 12.17c for illustration).

Table 12.4 Comparison of the detection sensitivity and specificity of the Hough transform and the Log Gabor filtering approach of [26] on the original database of Bernal et al.

	Sensitivity (%)	Specificity (%)
Hough transform	94	15
Log Gabor	42	89

At this stage, it can be noticed that the simple Hough transform allows a good detection of ROI containing a polyp even if the assumption made on the shape could be consider as restrictive since polyps are more elliptical than circular most of the times.

Moreover, if the value of specificity is low, the next classifying step will allow to improve the overall method performance.

12.4.2 Learning-Based Classification Performance Using Texture Features

For these experiments, the ad hoc generated polyp/non-polyp database were divided into two subgroups: A first one composed of 1000 images (200 images of polyps and 800 of non-polyps) for the learning process and a second group for testing composed of the remaining 500 images. In order to obtain classification performance statistically meaningful, the drawing of the elements of both learning and testing databases were randomly made and presented quantitative results correspond to the average value obtained on 100 different configurations.

In a first experiment, different kinds of methods for classification were compared: Learning Vector Quantization technic (LVQ) [35], classic Adaboost, and finally Attentional Boosting (cascade adaboost). In terms of performance, as long as, contrary to cascade adaboost, it is not possible to set the obtained performance for LVQ or classic Adaboost, we privileged the balance between “Sensibility” and “Specificity.” The results of this experimentation are shown in Table 12.5.

As it can be noticed, among the different classification technics used, Cascade Adaboost provides the best compromise between “Sensibility” and “Specificity”. If LVQ leads to a good classification of True Positive examples, the total amount of FPR remains too important considering the fact that 10 % of the polyps are misclassified.

In a second experiment, we tested the influence of the size of the cooccurrence matrix on obtained performance. The objective was to be able to reduce at its minimum the size of the cooccurrence matrix related to the grayscale resolution of the input image (default value: 8 bits), keeping in mind an hardware implementation on FPGA. More precisely, in Fig. 12.16, the performance (sensibility and specificity) for each considered resolution (256, 128, 64, 32, 16) is shown.

Table 12.5 Performance comparisons among different types of classification approaches

Type adaboost	Sensitivity (%)	Specificity (%)	FPR (%)
Real adaboost	77	92.5	7.5
Attentional	91	95.2	4.8
LVQ classification	92	86	14
[24]	89	98	2

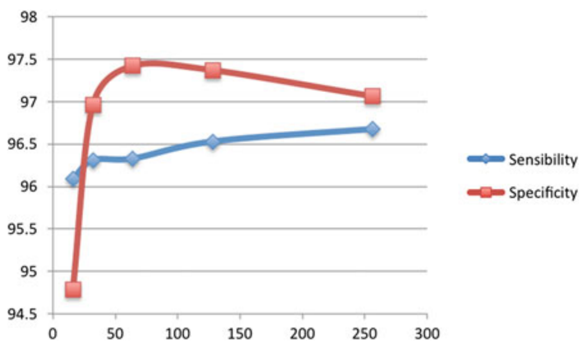


Fig. 12.16 Performance of the boosting process with respect to the resolution of the coocurrence matrix (256, 128, 64, 32, 16): Sensitivity is shown in *blue* and sensibility in *red*

Table 12.6 Average performance of the Cascade Adaboost learning process with a "Sensibility" set to a minimum of 99 %

Cascade Adaboost	Sensibility	Specificity	FPR
Mean (%)	99,5	86.1	13.9
Standard deviation	0.00	0.07	0.07

One can notice that the global performance remains at an acceptable rate even with a limited number of gray levels (only 16). Nevertheless, to ensure a maximum detection ratio a resolution of 5bits (32 gray-levels) was selected.

Finally, in a last experiment, only Cascade Adaboost is considered with a setting of the performance parameters (F_i and D_i of Fig. 12.7) chosen in order to have a "Sensibility" the closer to 100 %, whatever "Specificity" will be. This scenario fits better the expectations of radiologists who do not wish to miss possible polyps. For this experiment, in accordance with previous tests, the numer of gray level of the cooccurrence matrices is fixed to 32. Performance is shown in Table 12.6.

Table 12.6 shows that a high"Sensibility" is an objective that can be reached with a cascade adaboost setting of the learning process. Of course the "FPR" rate increases, but finally not that much considering the fact that for 100 polyps detected, only 14 more will be showed as possible candidates to the radiologist.

12.4.3 Examples of Detection and Classification Results

In Fig. 12.17 some examples of detection/classification are shown. ROI that are skirted by a non-bolded plain rectangle are the ROI candidate issued from the Hough transform step of the proposed approach. ROI skirted by a bold plain rectangle are those which are effectively identified as a polyp after the texture-based classification.

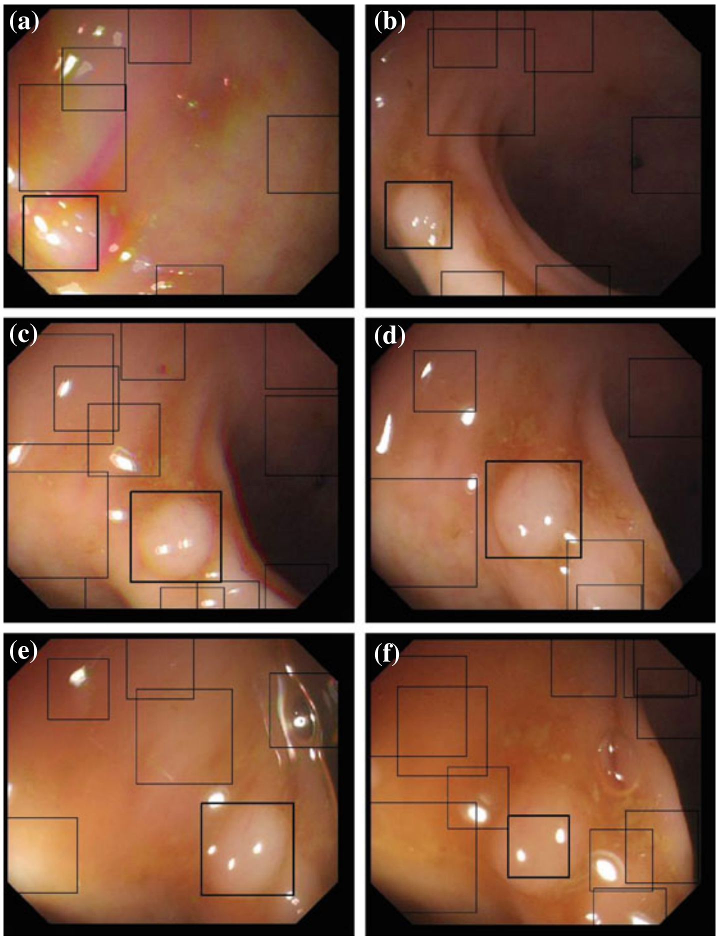


Fig. 12.17 Detection examples: ROI that are skirted by a *non-bolded plain rectangle* are the ROI candidate issued from the Hough transform step of the proposed approach. ROI skirted by a *bold plain rectangle* are those which are effectively identified as a polyp after the texture-based classification

12.4.4 Simulation and Time Processing

The developments have been performed under OpenCV environment in order to first establish the feasibility of this new approach and reduce the time development hardware. For comparison, a Matlab version of the code was also used. In both cases, no optimization of the code has been realized. Table 12.7 summarizes the obtained performance in terms of time processing using a iCore7 at 2.8 GHz.

In addition, a real condition test was performed in the gastroenterology department of Hôpital Lariboisière in September 2013, showing that on the usual machine used,

Table 12.7 Average Time Processing in terms of the language development used

Language	Average Time Processing per Image
Matlab	2.5 s
OpenCV	40 ms

a frame rate of only three or four image per second was reached due to the less important computational resources when compared to the machines used in our lab.

If compatible with a classic WCE performance in terms of frame rate (three images per second), these results show that there is a real need to go for VHDL implementation of the algorithms in order to ensure a 24 frames per second rate that will probably be the next standard of WCE.

In the following section, such a VHDL implementation is discussed with a positioning regarding the most recent state of the art.

12.5 Towards an Integrated Hardware Implementation

Currently, the Cyclope project only implemented in the hardware SVM-based classification of 3D object with an FPGA Virtex II-pro [21]. Nevertheless, the algorithms proposed here can be also be implemented on this platform considering the recent literature. Elhossini [36] proposed a memory-efficient architecture for implementing Hough Transform on FPGAs. The proposed architecture enables storing the Hough Transform space on the FPGA's memory blocks with no need for accessing external memory while processing large size images in real time with a 30 frame per second rate. Others hardware implementations allow to optimize the computation of angle using Cordic algorithms (COordinate Rotation DIgital Computer) [37], the time processing, the type (circle and/or line), etc. The main issue is to develop a Hough Transform architecture that minimize the memory space with a great precision and a high parallelism. Table 12.8 summarizes the technical aspects of the five main contributions in the field.

Optimized embedded architecture-based FPGA for an efficient and fast computation of gray-level co-occurrence matrices (GLCM) and Haralick textures features for use in high throughput image analysis applications where time performance is critical have been already studied. The three main contributions [32, 42, 43] focus on the design of hardware processor that makes possible to compute four distances (1, 2, 3, and 4 pixels) and four angles (0°, 45°, 90°, and 135°) in parallel. The main difference among these approaches lies in the strategy to address the neighboring pixel. Table 12.9 summarizes the performance of the three main recent contributions obtained on Virtex-II and Virtex-5 FPGA.

Boosting classification has been also implemented in hardware on FPGA [44]. According to the related work on the hardware implementation of Hough Transform,

Table 12.8 Main characteristics of five of the most recent references of the literature

Authors	Year	Frame size	Memory	Rate (FPS)	FPGA
Jau Ruen [38]	2006	256 × 256	Ext. 1.6 Mb	0.2	Altera Stratix 1
Souki [39]	2008	320 × 240	Ext. 4 Mb	0.3	Altera Cyclone 2
Geninatti [40]	2009	44 × 46	Ext. 8 Mb	30	Xilinx Spartan 3
Hardzeyeu [41]	2008	500 × 400	Ext.	800	Xilinx Virtex 2
Elhossini [36]	2012	800 × 600	Int. 250 kb	30	Xilinx Virtex 2

Table 12.9 Main characteristics of related work

Feature	Sieler [42]	Iakovidis [32]	Tahir [43]
Year	2010	2007	2004
FPGA	Virtex-5	Virtex II	Virtex II
Ref	XC5VLX50T	XCV2000E	XCV2000E
Frequency (MHz)	56.3	38.2	50
Area (FPGA used) (%)	21.9	45	59
Time processing (128 × 128)	2.4 ms	x	1.756 ms
Ext. Mem	x	800 kb	327 kb
Int. Mem (kb)	327	83.2	81.9

co-occurence matrices computation and boosting classification, it is feasible to embed our approach on FPGA circuit.

This first step is a prerequired one toward an ASIC design embedded in the WCE, but also to be able to precisely estimate the energy consumption related to the hardware implemented detection/classification algorithms. In a previous work, we demonstrated that 75 % of the power consumption of a smart RF sensor are due to the RF power budget [16]. Moreover, [45] showed the transmitter power consumption is a nonlinear function of the data rate and concluded that to increase the battery life of a smart sensor, the amount of data should be reduced. The overall gain of an intelligent transmission versus a continuous transmission in a standard WCE on one hand (see Fig. 12.7) depends on the estimation of the number of images (polyps and false positives) that will be transmitted, and on the other hand on the power consumption due to their processing.

Currently, we can make only an estimation of this overall gain based on hypothesis, because of the integrated circuit has not been yet designed and the in vivo experiments not achieved. We are working on the hardware implementation of the 2D classification on a FPGA-based platform. By considering the above state of the art, the fact that an FPGA also consumes 12 times more dynamic power than an equivalent ASIC on average [46] and the power consumption of Virtex 5, we can estimate that the power consumption due to the processing will be approximatively under the hundred of

μW. This feature is less than the power consumption of the usual eight white LED used for illumination and the RF transceiver.

Moreover, during a standard examination, around 50 000 images are sent to the data logger with a frame rate of 4 fps up to 35 fps. By considering the same examination with the possible presence of ten polyps and a FPR of 13.9 %, only 6960 images will be sent and a 7 factor can been won on the overall power consumption of the transmission RF.

12.6 Conclusion

In this Chapter, we introduced an embeddable method for polyp detection in videoendoscopic examinations. The entire detection chain combines geometric and textural features for polyp characterization: if the first geometric step remains simple with the use of the Hough transform, the textural features computed from co-occurrence matrices are integrated within a boosting-based approach making possible to achieve good classification performance similar to those of the most recent state-of-the-art article. At last, the complete developed detection/classification scheme is in accordance with a hardware implementation which is of primary importance for possible in situ application using WCE.

References

1. Parkin, M.C., Shin, F.J., Forman, B.F.: Globocan 2008 v1.2, cancer incidence and mortality worldwide: Iarc cancerbase no. 10. International Agency for Research on Cancer (2008)
2. Moglia, A., Menciassi, A., Dario, A., Cuschieri, A.: Capsule endoscopy: progress update and challenges ahead. Nat. Rev. Gastroenterol. Hepatol. **6**, 352–362 (2009)
3. Spada, C., Hassan, C., Munoz-Navas, M., Neuhaus, H., Deviere, J., Fockens, P., Coron, E., Gay, G., Toth, E., Riccioni, M.-E., Carretero, C., Charton, J.-P., Van Gossum, A., Wientjes, C.A., Sacher-Huvelin, S., Delvaux, M., Nemeth, A., Petruzziello, L., Prieto de Frias, C., Mayershofer, R., Aminejab, L., Dekker, E., Galmiche, J.-P., Frederic, M., Johansson, G.W., Cesaro, P., Costamagna, G.: Second-generation colon capsule endoscopy compared with colonoscopy. Gastrointest. Endosc. **74**(3), 581–589 (2011)
4. Bergwerk, A., Fleischer, D., Gerber, J.: A capsule endoscopy guide for the practising clinician: technology and troubleshooting. Medline **66**(6), 1188–1195 (2007)
5. Bang, S., Park, J.Y., Jeong, S., Kim, Y.H., Shim, H.B., Kim, T.S., Lee, D.H., Song, S.Y.: First clinical trial of the MiRo capsule endoscope by using a novel transmission technology: electric-field propagation. Gastrointest. Endosc. **69**(2), 253–259 (2009)
6. Najm, F.N.: A survey of power estimation techniques in VLSI circuits. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **2**(4), 446–455 (1994)
7. Benini, L., Hodgson, R., Siegel, P.: System-level power estimation and optimization. In: Proceedings of the 1998 International Symposium on Low Power Electronics and Design, ISLPED'98, pp. 173–178, New York. ACM (1998)
8. Bogliolo, A., Benini, L.: Robust RTL power macromodels. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **6**(4), 578–581 (1998)

9. Gupta, S., Najm, F.N.: Power modeling for high-level power estimation. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **8**(1), 18–29 (2000)
10. Gupta, S., Najm, F.N.: Analytical models for RTL power estimation of combinational and sequential circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **19**(7), 808–814 (2000)
11. Gupta, S., Najm, F.N.: Energy and peak-current per-cycle estimation at RTL. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **11**(4), 525–537 (2003)
12. Anderson, J.H., Najm, F.N.: Power estimation techniques for FPGAs. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **12**(10), 1015–1027 (2004)
13. Buyuksahin, K.M., Najm, F.N.: Early power estimation for VLSI circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **24**(7), 1076–1088 (2005)
14. Naehyuck, C., Kwanho, K., Gyu, L.H.: . Cycle-accurate energy measurement and characterization with a case study of the ARM7TDMI [microprocessors]. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **10**(2), 146–154 (2002)
15. Lee, H., Lee, K., Choi, Y., Chang, N.: Cycle-accurate energy measurement and characterization of FPGAs. Analog Integr. Circuits Signal Process. **42**(3), 239–251 (2005)
16. Suissa, A., Romain, O., Denoulet, J., Hachicha, K., Garda, P.: Empirical method based on neural networks for analog power modeling. IEEE Trans. Comput. Aided Des. Integ. Circuits Syst. **29**(5), 839–844 (2010)
17. Burch, R., Najm, F.N., Yang, B.S.P., Trick, T.N.: A Monte Carlo approach for power estimation. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **1**(1), 63–71 (1993)
18. Nemani, M., Najm, F.N.: High-level area and power estimation for VLSI circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **18**(6), 697–713 (1999)
19. Sreeramaneni, R., Vrudhula, S.B.K.: Energy profiler for hardware/software co-design. In: Proceedings of 17th International Conference on VLSI Design, pp. 335–340 (2004)
20. Eliakim, R., Yassin, K., Niv, Y., Metzger, Y., Lachter, J., Gal, E., Sapoznikov, B., Konikoff, F., Leichtmann, G., Fireman, Z., Kopelman, Y., Adler, S.N.: Prospective multi center performance evaluation of the second generation colon capsule compared with colonoscopy. Endoscopy **41**, 1026–1031 (2009)
21. Kolar, A., Romain, O., Ayoub, J., Viateur, S., Granado, B.: Prototype of video endoscopic capsule with 3-D imaging capabilities. IEEE Trans. Biomed. Circuits Syst. **4**(4), 239–249 (2010)
22. Ayoub, J., Granado, B., Mhanna, Y., Romain, O.: SVM based colon polyps classifier in a wireless active stereo endoscope. In: 2010 IEEE EMBC, pp. 5585–5588 (2010)
23. Liu, M., Lu, L., Bi, J., Raykar, V., Wolf, M., Salganicoff, M.: Robust large scale prone-supine polyp matching using local features: a metric learning approach. In: Fichtinger, Gabor, Martel, Anne, Peters, Terry (eds.) Medical Image Computing and Computer-Assisted Intervention. Lecture Notes in Computer Science, vol. 6893, pp. 75–82. Springer, Berlin (2011)
24. Bernal, J., Sanchez, J., Vilariño, F.: Towards automatic polyp detection with a polyp appearance model. Pattern Recognit. **45**(9), 3166–3182 (2012)
25. Figueiredo, P.N., Figueiredo, I.N., Prasath, S., Tsai, R.: Automatic polyp detection in pillcam colon 2 capsule images and videos: preliminary feasibility report. Diagn. Therapeutic Endosc. **182435**, 1–7 (2011)
26. Karargyris, A., Bourbakis, N.: Identification of polyps in wireless capsule endoscopy videos using log gabor filters. In: IEEE Workshop LiSSA, pp. 143–147 (2009)
27. Kodogiannis, V., Boulougoura, M.: An adaptive neurofuzzy approach for the diagnosis in wireless capsule endoscopy imaging. Int. J. Inf. Technol. **13**, 46–56 (2007)
28. Karkanis, S.A., Iakovidis, D.K., Maroulis, D.E., Karras, D.A., Tzivras, M.: Computer-aided tumor detection in endoscopic video using color wavelet features. IEEE Trans. Inf. Technol. Biomed. **7**(3), 141–152 (2003)
29. Tagzout, S., Achour, K., Djekoune, O.: Hough transform algorithm for FPGA implementation. Signal Process. **81**(6), 1295–1301 (2001)
30. Davis, L.S., Johns, S.A., Aggarwal, J.K.: Texture analysis using generalized co-occurrence matrices. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-1**(3), 251–259 (1979)

31. Haralick, R.M.: Statistical and structural approaches to texture. Proc. IEEE **67**(5), 786–804 (1979)
32. Iakovidis, D.K., Maroulis, D.E., Bariamis, D.G.: FPGA architecture for fast parallel computation of co-occurrence matrices. Microprocess. Microsyst. **31**(2), 160–165 (2007)
33. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**(3), 297–336 (1999)
34. Viola, S., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE CVPR Conference, pp. 511–518 (2001)
35. Kohonen, T.: Chapter learning vector quantization. The Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge (1995)
36. Elhossini, A., Moussa, M.: Memory efficient FPGA implementation of Hough transform for line and circle detection. In: CCECE, pp. 1–5 (2012)
37. Volder, J.E.: The CORDIC trigonometric computing technique. IRE Trans. Electron. Comput. **EC-8**(5), 335–339 (1959)
38. Ruen, J.J., Shie, M.S., Chen, C.: A circular Hough transform hardware for industrial circle detection applications. In: IEEE Conference on Industrial Electronics and Applications, pp. 1–6 (2006)
39. Souki, M.A., Boussaid, L., Abid, M.: An embedded system for real-time traffic sign recognizing. In Proceedings of the 3rd International Design and Test Workshop. IDT 2008, pp. 273–276 (2008)
40. Geninatti, S.R., Benavidez-Benitez, S.R., Hernandez-Calvino, M., Guil-Mata, N., Gomez-Luna, J.: FPGA implementation of the generalized Hough transform. In: Proceedings—2009, International Conference ReConFigurable Computing and FPGAs, pp. 172–177 (2009)
41. Hardzeyeu, V., Klefenz, F.: On using the Hough transform for driving assistance applications. In: 2008 International Conference on Intelligent Computer Communication and Processing, pp. 91–98 (2008)
42. Sieler, L., Tanougast, C., Bouridane, A.: A scalable and embedded FPGA architecture for efficient computation of grey level co-occurrence matrices and Haralick textures features. Microprocess. Microsyst. **34**(1), 14–24 (2010)
43. Tahir, M.A., Bouridane, A., Kurugollu, F.: An FPGA Based Coprocessor for the Classification of Tissue Patterns in Prostatic Cancer. Volume 3203 of Lecture Notes in Computer Science, pp. 771–780. Springer, Berlin (2004)
44. Mitéran, J., Matas, J., Bourennane, E., Paindavoine, M., Dubois, J.: Automatic hardware implementation tool for a discrete Adaboost-based decision algorithm. EURASIP J. Appl. Signal Process. **1035–1046**, 2005 (2005)
45. Wang, A.Y., Sodini, C.G.: On the energy efficiency of wireless transceivers. In: IEEE International Conference on Communications 2006, vol. 8, pp. 3783–3788 (2006)
46. Kuon, I., Rose, J.: Measuring the gap between FPGAs and ASICs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **26**(2), 203–215 (2007)