IFIP AICT 508

Thomas Hollstein
Jaan Raik
Sergei Kostin
Anton Tšertov
Ian O'Connor
Ricardo Reis
(Eds.)

VLSI-SoC: System-on-Chip in the Nanoscale Era – Design, Verification and Reliability

24th IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2016
Tallinn, Estonia, September 26–28, 2016
Revised Selected Papers

Springer

IFIP Advances in Information and Communication Technology 508

Editor-in-Chief

Kai Rannenberg, Goethe University Frankfurt, Germany

Editorial Board

TC 1 – Foundations of Computer Science
Jacques Sakarovitch, Télécom ParisTech, France

TC 2 – Software: Theory and Practice
Michael Goedicke, University of Duisburg-Essen, Germany

TC 3 – Education
Arthur Tatnall, Victoria University, Melbourne, Australia

TC 5 – Information Technology Applications
Erich J. Neuhold, University of Vienna, Austria

TC 6 – Communication Systems
Aiko Pras, University of Twente, Enschede, The Netherlands

TC 7 – System Modeling and Optimization
Fredi Tröltzsch, TU Berlin, Germany

TC 8 – Information Systems
Jan Pries-Heje, Roskilde University, Denmark

TC 9 – ICT and Society
Diane Whitehouse, The Castlegate Consultancy, Malton, UK

TC 10 – Computer Systems Technology
Ricardo Reis, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

TC 11 – Security and Privacy Protection in Information Processing Systems
Steven Furnell, Plymouth University, UK

TC 12 – Artificial Intelligence
Ulrich Furbach, University of Koblenz-Landau, Germany

TC 13 – Human-Computer Interaction
Marco Winckler, University Paul Sabatier, Toulouse, France

TC 14 – Entertainment Computing
Matthias Rauterberg, Eindhoven University of Technology, The Netherlands

IFIP – The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the first World Computer Congress held in Paris the previous year. A federation for societies working in information processing, IFIP's aim is two-fold: to support information processing in the countries of its members and to encourage technology transfer to developing nations. As its mission statement clearly states:

> *IFIP is the global non-profit federation of societies of ICT professionals that aims at achieving a worldwide professional and socially responsible development and application of information and communication technologies.*

IFIP is a non-profit-making organization, run almost solely by 2500 volunteers. It operates through a number of technical committees and working groups, which organize events and publications. IFIP's events range from large international open conferences to working conferences and local seminars.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is generally smaller and occasionally by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is also rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

IFIP distinguishes three types of institutional membership: Country Representative Members, Members at Large, and Associate Members. The type of organization that can apply for membership is a wide variety and includes national or international societies of individual computer scientists/ICT professionals, associations or federations of such societies, government institutions/government related organizations, national or international research institutes or consortia, universities, academies of sciences, companies, national or international associations or federations of companies.

More information about this series at http://www.springer.com/series/6102

Thomas Hollstein · Jaan Raik
Sergei Kostin · Anton Tšertov
Ian O'Connor · Ricardo Reis (Eds.)

VLSI-SoC: System-on-Chip in the Nanoscale Era – Design, Verification and Reliability

24th IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2016
Tallinn, Estonia, September 26–28, 2016
Revised Selected Papers

Springer

Editors
Thomas Hollstein
Tallin University of Technology
Tallinn
Estonia

Jaan Raik
Tallinn University of Technology
Tallinn
Estonia

Sergei Kostin
Tallinn University of Technology
Tallinn
Estonia

Anton Tšertov
Tallinn University of Technology
Tallinn
Estonia

Ian O'Connor
Ecole Centrale de Lyon
Ecully
France

Ricardo Reis
Federal University of Rio Grande do Sul
Porto Alegre
Brazil

ISSN 1868-4238 ISSN 1868-422X (electronic)
IFIP Advances in Information and Communication Technology
ISBN 978-3-319-67103-1 ISBN 978-3-319-67104-8 (eBook)
DOI 10.1007/978-3-319-67104-8

Library of Congress Control Number: 2017952848

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This book contains extended and revised versions of the highest quality papers selected from those presented during the 24th IFIP/IEEE WG10.5 International Conference on Very Large Scale Integration (VLSI-SoC), a global System-on-Chip design and CAD conference. The 24th conference was held at the Park Inn in the Radisson Meriton Conference and Spa Hotel in Tallinn, Estonia, September 26–28, 2016. Previous conferences took place in Edinburgh, Scotland (1981); Trondheim, Norway (1983); Tokyo, Japan (1985); Vancouver, Canada (1987); Munich, Germany (1989); Edinburgh, Scotland (1991); Grenoble, France (1993); Chiba, Japan (1995); Gramado, Brazil (1997); Lisbon, Portugal (1997); Montpellier, France (2001); Darmstadt, Germany (2003); Perth, Australia (2005); Nice, France (2006); Atlanta, USA (2007); Rhodes, Greece (2008); Florianopolis, Brazil (2009); Madrid, Spain (2010); Kowloon, Hong Kong, China (2011); Santa Cruz, USA (2012); Istanbul, Turkey (2013); Playa del Carmen, Mexico (2014); and Daejeon, South Korea (2015).

The purpose of this conference, which was sponsored by IFIP TC 10 Working Group 10.5, the IEEE Council on Electronic Design Automation (CEDA), and by IEEE Circuits and Systems Society, with the cooperation of ACM SIGDA, is to provide a forum for the presentation and discussion of the latest scientific and industrial results and developments as well as future trends in the field of System-on-Chip (SoC) design, considering the challenges of modern nano-scaled state-of-the-art and future manufacturing technologies. The down-scaling of the feature sizes of modern semiconductor technologies imposes numerous new challenges on physical and system-level design of SoCs. Increasing reliability challenges demand new concepts in fault-tolerance and testing. While classically, based on the bathtub fault rate model, testing has been applied after manufacturing only, in the nano-era, built-in testing concepts have to monitor the system's health status during its lifetime. Sophisticated design and architectural measures are required to cope with wear-out, and system lifetime health management and active fault-resilience are also required. VLSI-SoC addresses these future challenges and provides an internationally acknowledged platform for scientific contribution and industrial progress within this field.

For VLSI-SoC 2016, 36 papers out of 93 submissions were selected for presentation at the conference.

Out of these 36 full papers presented at the conference, 11 papers were chosen by a Selection Committee to have an extended and revised version included in this book. The selection process of these papers considered the evaluation scores during the review process as well as the review forms provided by members of the Technical Program Committee and session chairs as a result of the presentations.

The papers in this proceedings volume have authors from Belgium, China, France, Germany, Hong Kong, Iran, Israel, Italy, Singapore, and the USA. The Technical Program Committee for the regular tracks comprised 95 members from 25 countries.

VLSI-SoC 2016 was the culmination of the work of many dedicated volunteers: paper authors, reviewers, session chairs, invited speakers, and various committee chairs. We thank them all for their contribution.

This book is intended for the VLSI community, mainly those persons who did not have the chance to attend the conference. We hope you enjoy reading this book and that you find it useful in your professional life and for the development of the VLSI community as a whole.

August 2017

Thomas Hollstein
Jaan Raik
Sergei Kostin
Anton Tšertov
Ian O'Connor
Ricardo Reis

Organization

General Chairs

Jaan Raik	Tallinn UT, Estonia
Ian O'Connor	ECL Lyon, France

Technical Program Chairs

Thomas Hollstein	Frankfurt UAS, Germany
Krishnendu Chakrabarty	Duke University, USA

Special Sessions Chair

Matteo Sonza Reorda	Politecnico di Torino, Italy

PhD Forum Chair

Mario Schölzel	Potsdam University, Germany

Finance/Local Arrangements Chair

Maksim Jenihhin	Tallinn UT, Estonia

Publicity Chairs

Ricardo Reis	UFRGS, Brazil
Masahiro Fujita	Tokyo University, Japan
Said Hamdioui	Delft University, Netherlands

VLSI-SoC Steering Committee

Manfred Glesner	TU Darmstadt, Germany
Salvador Mir	TIMA, France
Michel Robert	University of Montpellier, France
Chi-Ying Tsui	HKUST, Hong Kong, China
Matthew Guthaus	UC Santa Cruz, USA
Ricardo Reis	UFRGS, Brazil
Luis Miguel Silveira	INESC ID, Portugal
Fatih Ugurdag	Ozyegin University, Turkey

Publication Chairs

Anton Tsertov	Tallinn UT, Estonia
Sergei Kostin	Tallinn UT, Estonia

Registration Chair

Siavoosh Payandeh Azad	Tallinn UT, Estonia

Web Chair

Tarmo Robal	Tallinn UT, Estonia

Local Organizing Committee

Lembit Jürimägi	Tallinn UT, Estonia
Siavoosh Payandeh Azad	Tallinn UT, Estonia

Technical Program Committee

Analog and Mixed-Signal IC Design

Jerzy Dabrowski (Co-chair)	Linköping University
Makoto Nagata (Co-chair)	Kobe University
Kenichi Okada	Tokyo Institute of Technology
Tsung-Hsien Lin	National Taiwan University
Jacob Wikner	Linköping University
Rashad Ramzan	National University of Computer and Emerging Sciences-FAST-NU
Pawel Sniatala	Poznan University of Technology
Robert Szczygiel	AGH - University of Science and Technology

System Architectures, NoC, 3D, Multi-core and Reconfigurable

Michael Hübner (Co-chair)	Ruhr-Universität Bochum
Dirk Stroobandt (Co-chair)	Ghent University
Jiang Xu	Hong Kong Universtiy of Science and Technology
Wim Vanderbauwhede	University of Glasgow
Ulya Karpuzcu	University of Minnesota
Joao Cardoso	FEUP/Universidade do Porto
Jishen Zhao	University of California
Leandro Indrusiak	University of York
Radu Teodorescu	Ohio State University

CAD, Synthesis, and Analysis

Peeter Ellervee (Co-chair)	Tallinn University of Technology
Ricardo Reis (Co-chair)	UFRGS
Masahiro Fujita	University of Tokyo
Takashi Kambe	Kinki University
Bei Yu	Chinese University of Hong Kong
Tiziano Villa	University of Verona
Srinivas Katkoori	University of South Florida
Jari Nurmi	Tampere University of Technology
Johnny Öberg	Royal Institute of Technology
Juha Plosila	Turku University

Prototyping, Verification, Modeling, and Simulation

Graziano Pravadelli (Co-chair)	University of Verona
Ian Harris (Co-chair)	University of California Irvine
Eli Arbel	IBM Haifa Lab
Anupam Chattopadhyay	Nanyang Technological University
Matthieu Moy	Verimag
Rob Aitken	ARM Ltd.
Goerschwin Fey	University of Bremen and German Aerospace Center - DLR
Laurence Pierre	Université de Grenoble
Sandip Ray	Intel Corporation
Francis Wolff	Case Western Reserve University

Circuits and Systems for Signal Processing and Communications

Fatih Ugurdag (Co-chair)	Ozyegin University
Tobias Noll (Co-chair)	RWTH Aachen
Dajiang Zhou	Waseda University
Luc Claesen	University of Hasselt
Hm Bae	KAIST
Per Larsson-Edefors	Chalmers University of Technology
Jongsun Park	Korea University
Oscar Gustafsson	Linköping University

Embedded System Architectures, Design, and Software

Zebo Peng (Co-chair)	Linköping University
Vijaykrishnan Narayanan (Co-chair)	Pennsylvania State University
Lars Bauer	Karlsruhe Institute of Technology - KIT
Ing-Chao Lin	National Cheng Kung University

Zili Shao	Hong Kong Polytechnic University
Hai Li	University of Pittsburgh
Yu Wang	Tsinghua University
Paul Pop	Technical University of Denmark
Jason Xue	City University of Hong Kong
Ingo Sander	Royal Institute of Technology – KTH

Low-Power and Thermal-Aware Design

Aida Todri-Sanial (Co-chair)	LIRMM Montpellier
José L. Ayala (Co-chair)	Complutense University of Madrid
Nadine Azemard	LIRMM/CNRS
Masaaki Kondo	University of Tokyo
Mirko Loghi	Università di Udine
Jose Luis Abellan	Catholic University of Murcia
Marina Zapater	Universidad Politécnica de Madrid
Andrea Bartolini	University of Bologna

Memory Technologies, Circuits, and Systems

Lionel Torres (Co-chair)	LIRMM Montpellier
Yiran Chen (Co-chair)	University of Pittsburg
Jingtong Hu	Oklahoma State University
Wujie Wen	Florida International University
Elena Ioana Vatajelu	Politecnico di Torino
Weisheng Zhao	Spintronics Interdisciplinary Center, Beihang University
Danghui Wang	NorthWestern Polytechnical University
Jean-Michel Portal	Ecole Polytech' Marseille
Olivier Thomas	CEA-Leti
Nitin Chandrachoodan	IIT Madras
Jean-Michel Portal	IM2NP

Design for Variability, Reliability, and Testing

Bernd Becker (Co-chair)	University of Freiburg
Erik Larsson (Co-chair)	Lund University
Tony Kim	Nanyang Technological University
Sandeep Kumar Goel	TSMC
Saqib Khursheed	University of Liverpool
Stephan Eggersglüß	University of Bremen
Stefano Dicarlo	Politecnico di Torino
Emil Gizdarski	SYNOPSYS
Satoshi Ohtake	Oita University

Security

Lilian Bossuet (Co-chair)	University St. Etienne
Mihalis Maniatakos (Co-chair)	NYU Abu Dhabi
Paolo Maistri	TIMA Laboratory
Julien Francq	Airbus Defence & Space - CyberSecurity
Debdeep Mukhopadhyay	IIT Kharagpur
Jeyavijayan Rajendran	University of Texas at Dallas
Joseph Zambreno	Iowa State University
Xueyang Wang	Intel Corporation
Yier Jin	University of Central Florida

Contents

Enabling Internet-of-Things with Opportunities Brought by Emerging Devices, Circuits and Architectures

Xueqing Li(✉), Kaisheng Ma, Sumitha George, John Sampson, and Vijaykrishnan Narayanan(✉)

Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, PA 16802, USA
{lixueq, kxm505, sug241, sampson, vijay}@cse.psu.edu

Abstract. In recent years, the concept of Internet-of-Things (IoT) has attracted significant interests. Required by the applications, the IoT power optimization has become the key concern, which relies on innovations from all levels of device, circuits, and architectures. Meanwhile, the energy efficiency of existing IoT implementations based on the CMOS technology is fundamentally limited by the device physics and also the circuits and systems built on it. This chapter focuses on a different dimension, exploring how emerging beyond-CMOS devices, such as tunnel field effect transistor (TFET) and negative capacitance FET (NCFET), and the circuits and architectures built upon them, could extend the low-power design space to enable IoT applications with beyond-CMOS features.

Keywords: Internet-of-things · Emerging devices · Tunnel FET · Negative capacitance FET · Energy harvesting · Nonvolatile memory · Nonvolatile computing

1 Introduction

Improved sensing, signal processing, and communication has significantly changed the connection between humans and the world with the rise of intelligent devices being developed for the Internet-of-things (IoT) [1]. As designers seek to make these IoT systems smarter and more ubiquitous, high energy-efficiency has been the key to enhance both connectivity and IoT signal processing functionality. Cross-layer efforts in improving solid-state devices, and the circuits and systems built upon them, are the key to achieve the high energy efficiency demanded by an expanding future of IoT tasks.

Concurrently, the needs for portability and mobility, common in IoT applications, have driven devices toward battery and/or ambient energy harvesting power solutions [2]. In the past few decades, the power consumption of integrated circuits has been lowered significantly through the scaling of the CMOS technology together with signal processing techniques. Such achievement has made more and more IoT applications feasible while being powered with a modest battery capacity or ambient energy

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 1–23, 2017.
DOI: 10.1007/978-3-319-67104-8_1

harvester (e.g. a solar cell). However, further power reduction has become more and more challenging for conventional CMOS technology (including FinFET innovations) and the computation and communication methodologies built upon it. The conventional means of power reduction alongside CMOS scaling of using a lower supply voltage to reduce the dynamic power consumption while simultaneously reducing threshold voltages to provide sufficient computing speed causes exponentially increasing leakage power, which can now approach magnitudes similar to dynamic power. This fundamentally limits the expansion of functionality via CMOS scaling alone, especially when IoT devices are powered by batteries or harvested energy.

Battery-less IoT systems face further challenges in obtaining sufficient energy from the low and intermittent power source in the ambient environment [3]. Existing energy-harvesting circuits may encounter a low-input voltage that leads to a low power-conversion efficiency. Low harvested power not only limits the average amount of tasks being performed, but also increases the response latency, which is a key factor of quality-of-service (QoS). Meanwhile, distinct from conventional computing systems with a stable supply, the intermittency of harvested power also requires additional backup and restore operations, which consumes extra energy and time and carries the risk of losing computation progress if a backup operation is not carried out in time.

While these fundamental challenges have become a barrier when using CMOS technology, the advent of emerging technologies have brought new opportunities. These emerging technologies include emerging transistor devices, circuits, and architectures. The new opportunities can be seen, broadly, as advances in two key directions. Firstly, the Boolean switching behavior of some emerging transistors can replace the existing CMOS transistors in conventional computing approaches with substantially improved prospects for power scaling and low-voltage operation [4]. Secondly, certain emerging devices inherently support nonvolatile data storage and computing, enabling low-energy memory access and backup/restore operations.

There have been quite a few promising beyond-CMOS emerging devices, such as single-electron devices [5], spin-transfer-torque devices [6], the tunnel field effect transistor (TFET) [7], negative capacitance FET (NCFET, aka ferroelectric FET or FeFET) [8]. This chapter introduces two types of them, including TFET and NCFET. As promising beyond-CMOS candidates, these devices could work at a lower supply voltage to enable further power reduction in Boolean computation (without higher static leakage than CMOS). Meanwhile, the substantially novel features that they exhibit could also be captured to enable new computing architectures supporting nonvolatile data storage and computing.

The remainder of this chapter proceeds as follows. Section 2 investigates the challenges in designing energy-efficient IoT systems. Section 3 introduces the three types of emerging technologies, with more emphasis on their electrical characteristics. Section 4 describes how to make use of these emerging devices to design more energy-efficient IoT systems beyond those in CMOS. Section 5 discusses future research directions and Section 6 discusses key conclusions.

2 IoT Systems and Efficiency Bottlenecks

This section presents a model of a general IoT system, describes the functionality of each block, and analyzes the bottlenecks in each block considering existing optimization efforts.

2.1 A General IoT System

While there has been a relatively long history of using solar cells to power devices, recently published battery-less IoT system designs have been demonstrated with an increasingly wide range of power sources. Devices powered by harvested radio-frequency (RF) energy have been shown to be successful for applications including a glucose level sensor on a contact lens, a highway RFID pass, bio-signal sensors on animals or insects, etc. [2]. Their system functionality varies from a simpler signal recorder to a more complex in situ signal processor, such as one with EEG signal processing, and wireless transmission. The system feature size, operating range, performed tasks, circuit design and architecture implementation, should be optimized based on the amount of obtainable energy and other quality-of-service (QoS) requirements in the applications.

The system structure varies with the specific application requirements. A general battery-less IoT system could be built as shown in Fig. 1 using ambient-energy-harvesting techniques [2, 9]. While some blocks, such as sensors and interface, memory storage, and a digital signal processor and accelerators, can be similar to conventional designs with a stable power supply, there are extra and significantly different blocks when the system is battery-less and powered by ambient energy-harvesting techniques.

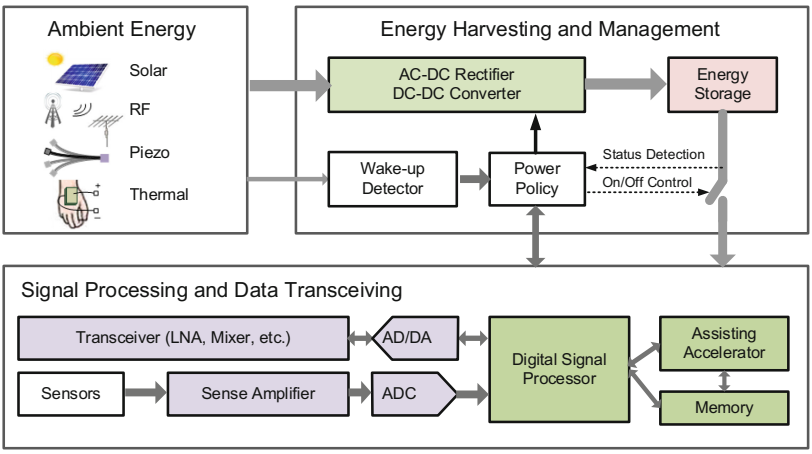


Fig. 1. A general battery-less IoT system powered by ambient-energy harvesting [9]

In addition to the external power sources, the energy-harvesting IoT system consists of two major blocks: the block of energy harvesting and management, and the block of signal processing and data transceiving. The energy harvester differs with the ambient energy source, and a wake-up receiver may be used in scenarios when an external triggering signal is used to switch the system between different power or operation modes. A temporary energy storage medium in the form of a capacitor is usually used to smooth the supply glitch and cover a temporary power income loss. As will be further discussed later, the power supply and management module, and the digital signal processing architecture for an energy-harvesting IoT system can be significantly different from conventional designs with a stable power supply. In fact, the overall system performance greatly relies on how these different blocks are built. The next sub-section (Sect. 2.2) will discuss more details of each block.

2.2 Bottlenecks and Existing Efforts

Energy sources and energy harvesting techniques. Solar, RF, piezoelectric and thermal gradients have been widely used ambient energy sources [23]. When the energy source does not directly provide the required DC voltage output, voltage converters and regulators are needed. For example, a rectifier is required to convert AC signals from an RF signal antenna and piezoelectric films. DC-DC converters can be used to convert the DC supply voltage to be higher or lower. Despite of the differences between these ambient power sources, there are three major challenges in energy harvesting and storage. The first challenge is the relatively low and varying energy density, intermittency, dependency of the efficiency on the load condition, and the unpredictability of these factors. Therefore, circuit optimizations such as tracking and adaptive operations [10] are usually required to mitigate these effects which significantly increases the design complexity. The second challenge is the low power-conversion efficiency (PCE) because of the weak power from the ambient environment. Such a weak power results in low-voltage operation and thus a high resistive energy loss with conventional CMOS technology [11, 12]. The third challenge is the leakage of the energy storage capacitors, which makes the approach of "short-time-computing, long-time-harvesting" less applicable in ultra-low input power scenarios.

Sensing, interface, and communication. While this can be similar to IoT systems with a stable supply, the increasing amount of data being transferred by the IoT devices, the relatively much lower energy budget, and the unpredictable power outages make the interface challenging. There is not yet a mature protocol to deal with frequent supply failures in IoT. Some techniques, such as passive communications (e.g. backscatter in [16]), are useful to reduce the power, but limited in the operation range, speed, and overall energy-efficiency when considering the power transmitter.

Digital signal processing. There are two main challenges in the design of digital signal processors. The first challenge, as introduced in Sect. 1, is that the slowing down of voltage scaling has become challenging because of increased leakage power. The question of how to build reliable and energy-efficient digital processing circuits under a

lower voltage has become a hot topic in device, circuit, and architecture research. The second challenge, which is a result caused by intermittent supply failure, is that the frequent backup-restore operations consume significant amounts of energy, limiting the overall forward computing progress. There has been some initial research on the optimization of nonvolatile processors (NVPs) recently, as will be discussed later, showing great potential to mitigate the impact of power intermittency [13–15]. Nevertheless, the study of signal processing algorithms, computing architectures for IoT systems is still insufficient for digital signal processing under an intermittent power supply.

Data storage. For IoT systems, especially in sensing applications, memory elements are needed to store data before they are processed and transferred. Future IoT data storage will be using more memory as the task complexity increases. While the required data storage volume varies with the application, the major challenge in data storage for energy-harvesting IoT systems is the energy efficiency in read and write access due to a low energy budget. This challenge is particularly critical for on-chip nonvolatile memory (NVM) designs, as recent research has revealed the advantage of integrated on-chip NVM to reduce access energy and delay [13–15, 17, 18]. It is likely that the co-design of data storage and signal processing architecture will be critical for overall energy efficiency, especially for some applications where memory access is the bottleneck due to frequent backup and restore operations [19].

Other issues. Other challenges, such as security and privacy [20], reliability, yield, etc., which are not covered by this chapter, will also be critical in future IoT systems.

3 Emerging Beyond-CMOS Devices

In this section, TFET and NCFET, as emerging beyond-CMOS devices, will be introduced and compared with conventional CMOS. At the device level, there are a few widely-used performance metrics to evaluate a device:

ON-state current (I_{ON}): drain current when the transistor is in the *ON* state. I_{ON} is usually measured with both the gate-source voltage (V_{GS}) and the drain-source voltage (V_{DS}) set to be equal to the supply voltage. A higher I_{ON} is equivalent to smaller on-state resistance, and is thus preferred for higher speed.

OFF-state current (I_{OFF}): drain current when the transistor is in the *OFF* state. I_{OFF} is usually measured with V_{GS} equal to zero and V_{DS} equal to the supply voltage. A lower I_{OFF} indicates larger off-state resistance, and is preferred for lower leakage current.

Subthreshold swing (SS): the required voltage change at the transistor gate to change the drain-source current by a decade in the subthreshold region. In conventional CMOS FETs, SS is limited by the thermionic emission of carriers, and is higher than 60 mV/decade at the room temperature. A transistor with a smaller *SS*, could be operating at a lower supply voltage, while providing the same I_{ON} and I_{OFF}. This capability reduces overall power consumption by reducing the dynamic power (as the voltage is lower). A smaller *SS* in analog and RF circuits is also preferred, because it also leads to higher $g_m I_D$ for higher gain and current efficiency:

$$\frac{g_m}{I_D} = \frac{\partial I_D}{\partial V_{GS}} \frac{1}{I_D} = \frac{\partial ln I_D}{\partial V_{GS}} = \frac{ln10}{ss} \tag{1}$$

Steep-slope devices: in this chapter, it is used to represent devices with *SS* lower than 60 mV/decade of conventional CMOS FETs at the room temperature.

3.1 TFET

TFET is essentially a gated p-i-n diode with reverse biasing and asymmetric doping [7]. There have been many types of reported TFET devices [7]. The double gate GaSb-InAs heterojunction TFET (HTFET) device has shown good balance between a steep slope and high I_{ON}, as shown in Fig. 2(a) [11]. When the gate bias voltage is low, the drain source current is small. This is because the wide energy barrier suppresses the probability of band-to-band tunneling (BTBT), as shown in Fig. 2(b). When the gate voltage is increased, the tunneling barrier is narrowed. As a result, the quantum-mechanical BTBT phenomenon creates an abrupt transition between the *ON* and *OFF* states as shown in Fig. 2(c), achieving a low *SS* at the room temperature as shown in Fig. 2(d).

In addition to the steep-slope switching characteristic, HTFET also exhibits some unique features shown in Fig. 2(e–f) [2, 11, 12, 21]. The first feature is the uni-directional tunneling that makes TFET conducting current almost drain-to-source only in a moderate voltage range. This originates from the asymmetric structure in HTFET. The second feature is the negative differential resistance (NDR), which appears in the negative V_{DS} range. The third feature is about the device capacitance. HTFET has less capacitance than

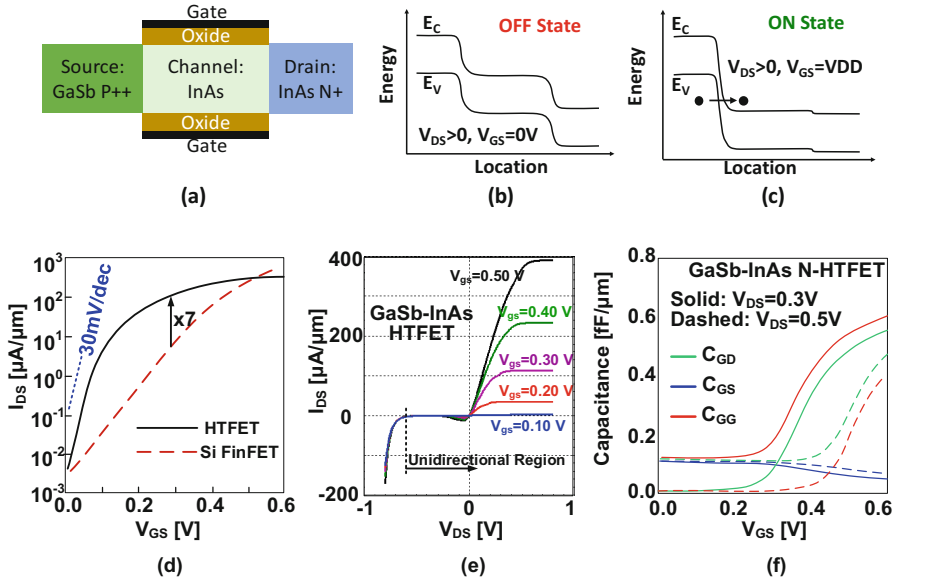


Fig. 2. HTFET: (a) Structure of an N-type HTFET; (b–c) Energy diagrams; (d, e) I_{DS}-V_{GS} comparison; (f) Capacitance [2, 11, 12, 21]

Si FinFET in the low voltage region, and more capacitance in the high voltage region. Table I summarizes some recent TFET experimental results. Device models for TFET are available for circuit SPICE simulations [11, 12, 22–25].

3.2 NCFET

A negative differential capacitor was predicted in 2008 to be stacked at the gate insulator in a MOSFET. By doing this, a small voltage change at the gate could create a larger change in the insulator surface potential, leading to a steeper switching behavior in the I_{DS} versus V_{GS} curves of the transistor [26]. Figure 3(a–b) shows the conceptual device structure and the equivalent gate capacitance network. Recently, there have been advances in both fundamental and experimental results [27–33]. Table 1 shows some recent NCFET results. Due to the challenge of integrating the ferroelectric layer, some early devices were shown with an external ferroelectric capacitor. Recent reported devices are capable of integrating the ferroelectric capacitor around a fin-structure gate.

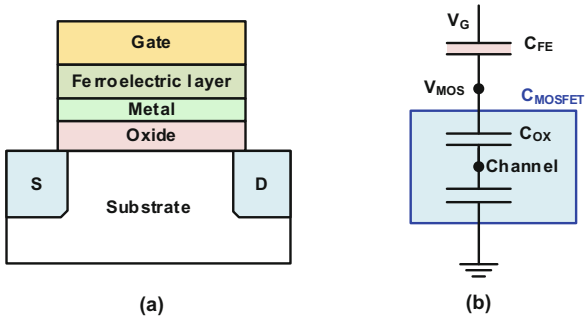


Fig. 3. NCFET: (a) Device structure; (b) Capacitance model [18]

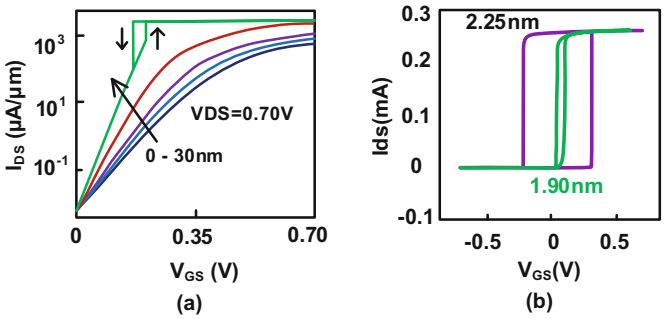


Fig. 4. NCFET simulated switching behavior versus ferroelectric layer thickness [18, 34, 36]

Table 1. Recent advances in TFET, NCFET, and PTD-based PhaseFET

	NCFET			TFET		
Source	[32] IEDM'15	[29] EDL'16	[30] EDL'16	[45] IEDM'14	[44] EDL'15	[43] VLSI'15
Structure	HfZrOFinFET	P(VDF-TrFE)	BiFeO3, FinFET	Si FinFET	III-V vertical	III-V vertical
I_{ON} (A/m)	-	100	1e-4–1e-6	-	8.4	275 N; 30 P
I_{OFF}	-	~5pA/m	1e-12–1e-14	I_{ON}/3e4 N; I_{ON}/2e6 P;	0.1nA/m	0.8nA/m N; 0.3nA/μm P
SS_{min} (mV/dec)	55–87	45–52 (2-4 w/ hysteresis)	8.5–11 P; 16–50 N	56 N;58 P;	64 N	55 N; 115 P;
Hysteresis	Depends	no	yes	no	no	no

Many ferroelectric materials, including PbTiO, BaTiO, Pb(ZrTi)O, HfZrO, etc., could exhibit negative capacitance [30]. The matching of the ferroelectric negative capacitance and the internal MOSFET gate capacitance is the key towards the performance of an NCFET. Thus, a proper capacitance tuning through ferroelectric material layer thickness and area is critical to the success of an NCFET process [30]. Figure 4 shows how the ferroelectric layer thickness affects the switching slope and hysteresis [18, 36]. As the ferroelectric layer thickness increases, *SS* reduces, and a hysteresis window gradually appears and then finally covers both positive and negative V_{GS} range. These characteristics of hysteresis, a steep slope, and their dependence on the ferroelectric material, have been explored in digital logic and memory circuit design [18, 34, 35].

4 New Opportunities Enabled by Emerging Devices, Circuits, and Architectures

This section shows how the IoT system bottlenecks could be mitigated by the opportunities enabled by these emerging devices.

4.1 Energy Harvesters and Sensors with Higher Efficiency

It is intuitive that, by increasing harvested energy from the same ambient environment, the number of performed tasks and functionalities could be increased in an energy-harvesting IoT system. Existing research results have shown that, by making use of the steep switching characteristics, energy harvesters based on these emerging devices could operate better than CMOS transistors in the low-voltage scenarios. Figure 5(a) is a conventional cross-coupled RF rectifier. Figure 5(b) is a conventional DC-DC charge pump. Figure 5(c) is an enhanced TFET DC-DC charge pump topology [11, 12]. The power conversion efficiency (PCE) comparisons in Fig. 6 shows how III-V heterojunction TFET (HTFET) based designs outperform those based on the Si FinFET technology.

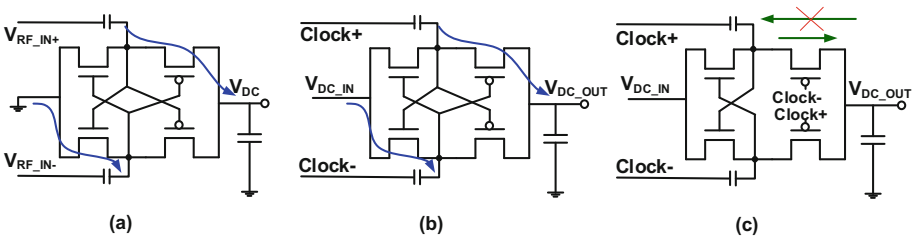


Fig. 5. Rectifier and DC-DC charge pumps: (a) Rectifier; (b) Conventional DC-DC charge pump; (c) Enhanced DC-DC charge pump in III-V HTFET [11, 12]

There are a few factors that lead to the improvement of power conversion efficiency when using HTFETs. The first factor is lower resistive power loss. When the input voltage is low, the resistive power loss limits the overall power conversion efficiency, and the designs in HTFET have less resistive power loss, leading to significant benefits. The second factor is lower capacitive power loss during charge redistribution when the input voltage is low. A combination of these two factors leads to a better transistor sizing strategy for the trade-off between the resistive power loss and switching capacitive loss. The third factor is the uni-directional tunneling conduction which leads to lower reverse power loss in a form of leakage current from the output to the input.

The uni-directional tunneling feature of HTFETs also enables a new circuit topology towards even higher efficiency. For example, in the enhanced HTFET DC-DC converter in Fig. 5, the gate control of the output p-type transistor is now controlled directly by the input clock signal, which enables doubled gate driving voltage and less resistive power loss.

By providing higher power conversion efficiency, HTFET significantly extends the IoT operating applications to lower energy-income scenarios. It is also noted that, from another aspect, an energy harvester itself could be treated as a sensor that senses the input power level. A higher PCE provided by HTFET also improves the sensing sensitivity, such as motion or vibration sensors and radiation sensors. Similar rectifier and DC-DC charge pump designs based on NCFET and PhaseFET, although there is no result reported, a higher PCE will not be a surprise.

4.2 Analog Processing and Communication

For analog processing and transceiver designs, the lowest achievable power consumption is determined by the trade-off between various specifications, including gain, speed and bandwidth, linearity and spectral performance (such as spurious-free dynamic range or SFDR, signal-to-noise + distortion ratio or SNDR, input-referred noise), etc. Figure 7 shows the evaluation results of TFET based designs, including A/D converter, sense amplifier and D/A converter.

Figure 7(a) evaluates a 6-bit 10-MS/s successive-approximate-register (SAR) A/D converter, and Fig. 7(b) shows how HTFET is capable of lowering the energy beyond the limit of CMOS [24]. Such a gain stems from higher current efficiency for both digital logic (lower dynamic power) and the comparator (higher g_m/I_D). Figure 7(c–d)

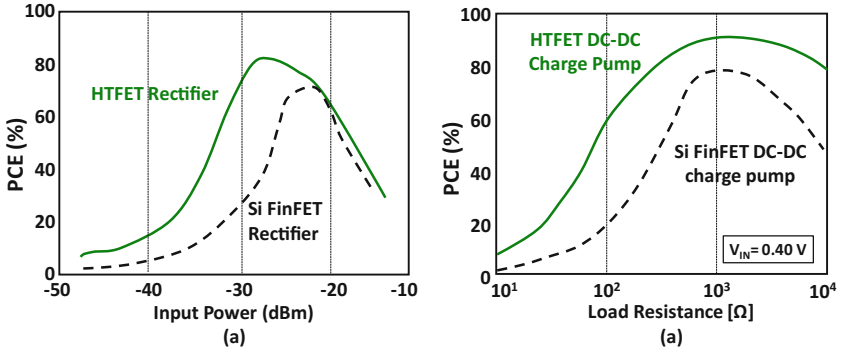


Fig. 6. PCE of rectifier in (a) and DC-DC charge pump in (b) [11, 12]

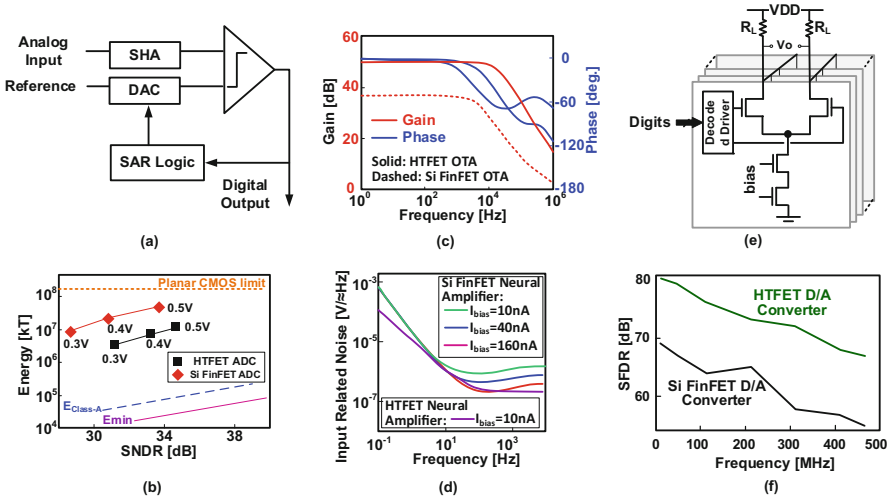


Fig. 7. Comparisons between HTFET and Si FinFET circuits: (a–b) a 6-bit SAR A/D converter and performance; (c–d) bio-signal sensing OTA gain and input referred noise versus frequency; (e–f) Current-steering D/A converter and its SFDR [22, 24, 25]

shows the performance evaluation of a low-noise bio-signal sense amplifier (LNA) [22]. Here HTFET based design also has a higher gain because of higher g_m/I_D. A higher gain also leads to the input referred noise reduction as by definition, the input referred noise is the output noise divided by the gain of the amplifier. Figure 7(d–e) shows the performance evaluation of a current-steering D/A converter [25]. HTFET shows a higher SFDR because of less transistor capacitance at the low voltage region, which leads to less coupled switching glitches and higher output impedance.

TFET based typical RF circuit designs were reviewed in [37], including RF LNA, mixer, frequency doubler, oscillators, etc. Substantial benefits are observed using

HTFET in low-voltage high-frequency circuits, with higher preferred nonlinearity for mixers, and higher transconductance and gain at low power and low current levels.

Considering that the above designs are widely used as a front-end and back-end block in IoT systems, as shown in Fig. 1, significant power saving could be achieved by adopting HTFET.

4.3 Energy-Efficient Volatile Digital Logic

There have been evaluations between TFET and conventional CMOS technologies on digital circuits, including combinational gates and adders, sequential gates like D flip-flops, and SRAM. TFET based designs are shown to outperform conventional CMOS in energy-delay especially with a low supply voltage, as shown in Fig. 8 [38, 39]. It is also noted that, when using TFET for pass-transistor logic, the device feature of uni-directional tunneling conduction affects the functionality and is handled with by either adding another parallel pass transistor for the other opposite direction conduction, or re-designing the circuit topology.

As the technology scales down to smaller dimensions, the parasitics and contact non-idealities play a more important role. Recently, an evaluation work considering parasitics indicates that, similar performance advantage by HTFET is still observed even with higher contact resistance due to a vertical structure [40]. Another work on processor design and evaluation shows that, with less energy per instruction (EPI), TFET based designs extends the design space when considering the thermal limit and the degree of parallelism, leading to higher performance [41].

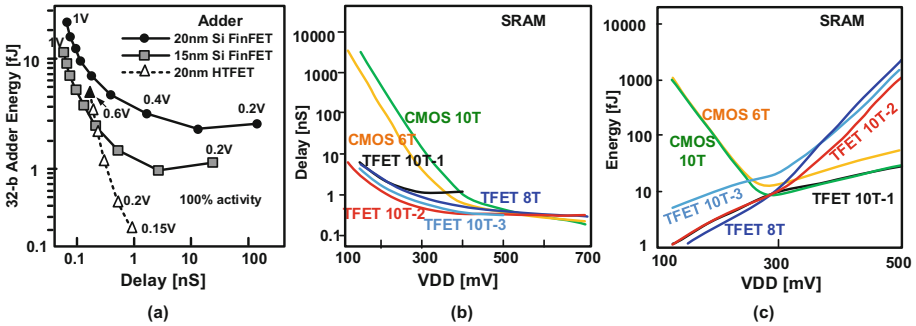


Fig. 8. TFET design examples: (a) a 32-b adder; (b–c) SRAM [38, 39]

Similarly, for NCFET, lower energy-delay has been observed for digital logic in low-voltage scenarios when operating with a moderate-to-high capacitive wire load, as shown in Fig. 9(a) [35]. The hysteresis in the positive V_{GS} region as shown in Fig. 5(c) could significantly improve the input noise margin by an amount of the hysteresis window width [34]. The theory of this could be understood as follows. Considering an NCFET inverter with n-type NCFET transistor and p-type conventional transistor, the n-type NCFET transistor will not turn on until the input voltage increases beyond the

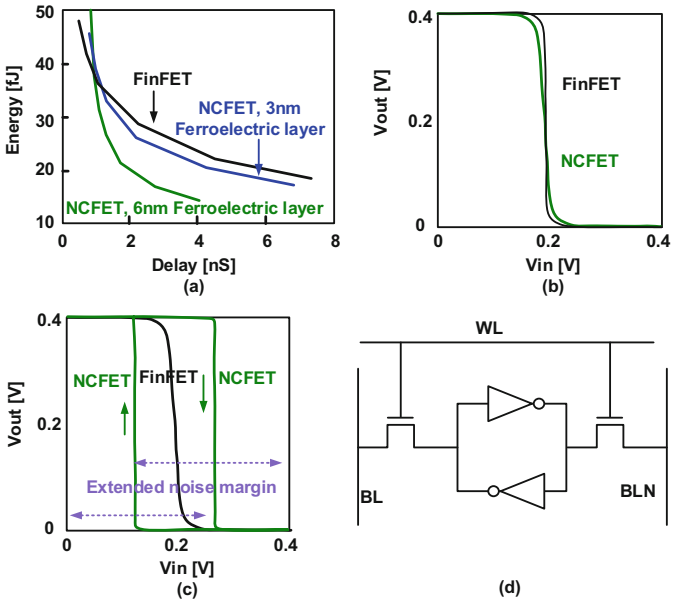


Fig. 9. NCFET evaluation [34, 35]: (a) Energy-delay for a Koggy-Stone adder; (b) Inverter input-output transfer function (NCFET has 16 nm ferroelectric layer thickness); (c) Inverter input-output transfer function (NCFET has 27 nm ferroelectric layer thickness) showing improved input noise margin with NCFET hysteresis; (d) NCFET SRAM with enhanced noise margin with NCFET hysteresis.

rising hysteresis edge, nor will it turn off until the input signal reduces beyond the falling hysteresis edge. This is illustrated in Fig. 9(c). The improved input noise margin of NCFET logic could also be used to build SRAM cells with enhanced noise margin, as shown in Fig. 9(d) [34].

4.4 Energy-Efficient Nonvolatile Logic and Memory Circuits

For IoT energy-harvesting applications where the power supply is intermittent, it is critical to sustain inter-process data during power outages. Therefore, on-chip non-volatile memory (NVM) becomes intriguing because of its non-volatility to avoid refreshing and its immunity to power failures. The possibility of on-chip memory access instead of out-of-chip access also reduces the energy consumption. Meanwhile, power-gating is very useful to further reduce the static leakage power of idle digital circuits, and NVM could be used to store the state of these idle circuits while turning off their power supply.

Furthermore, with on-chip NVM and associated sensing and control, a nonvolatile processor (NVP) could be built to back up the processor states and data, including memory, D flip-flops (DFF), registers, etc., into this NVM during power failures [14, 42, 48–54]. Such on-chip data backup and restore operations reduce the risk of losing computation progress. When compared with out-of-chip nonvolatile backup options,

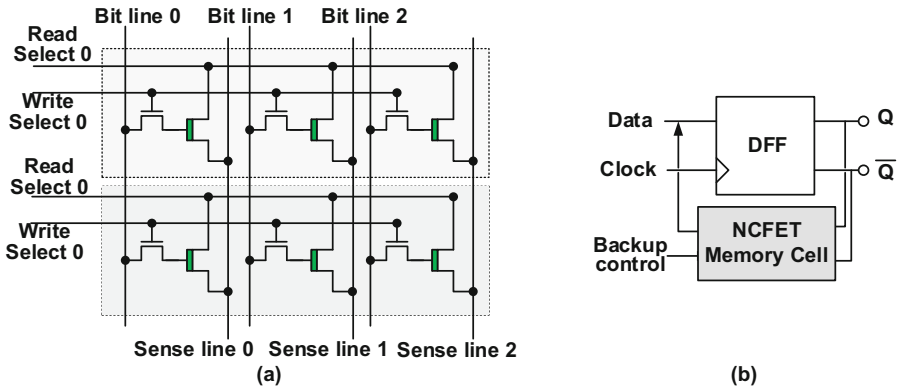


Fig. 10. NCFET circuits [18, 34]. (a) Two-transistor (2T) nonvolatile memory array; (b) Nonvolatile NCFET D flip-flop.

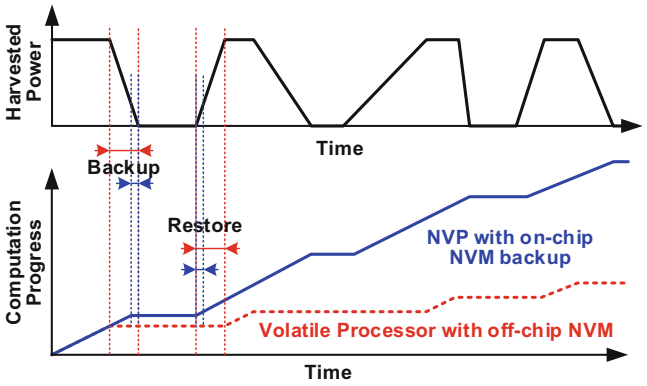


Fig. 11. Comparisons of computation progress between volatile processor with off-chip NVM and nonvolatile processor with on-chip NVM.

this on-chip backup solution has lower power, energy and interface overhead. Such an advantage enables more computational progress in power-supply-intermittent scenarios [13], as illustrated in Fig. 11.

With the tunable hysteresis in NCFETs, energy-efficient nonvolatile memory could be built. Figure 10(a) shows an NCFET NVM design based on an NCFET hysteresis tuned around $V_{GS} = 0$ V (see Fig. 3(d)) [18]. It is reported that this NCFET NVM exhibits improved access energy-delay. Different from existing nonvolatile memory devices such as ReRAM and STT-RAM, the NCFET itself is also a transistor. This provides opportunities of logic-in-memory process.

Attaching an NCFET nonvolatile bit storage to a conventional volatile DFF, a nonvolatile DFF with external backup and reset controls could also be built, as illustrated in Fig. 10(c) [34]. With a local nonvolatile memory cell, the backup and restore

energy becomes lower than that of the clustered nonvolatile memory backup solution in which long-distance data transmission is time and energy consuming.

More aggressively, by exploring the embedded logic and non-volatility in NCFET, an external-control-free intrinsically nonvolatile DFF is possible. Such an intrinsically NV-DFF could be built by replacing the slave latch of a conventional volatile CMOS master-slave DFF with one NCFET nonvolatile latch shown in Fig. 12. Making the NV-DFF intrinsically nonvolatile enables the removal of external controls, and makes fine-grained backup/restore operations in NVP and power-gating applications possible with more energy savings.

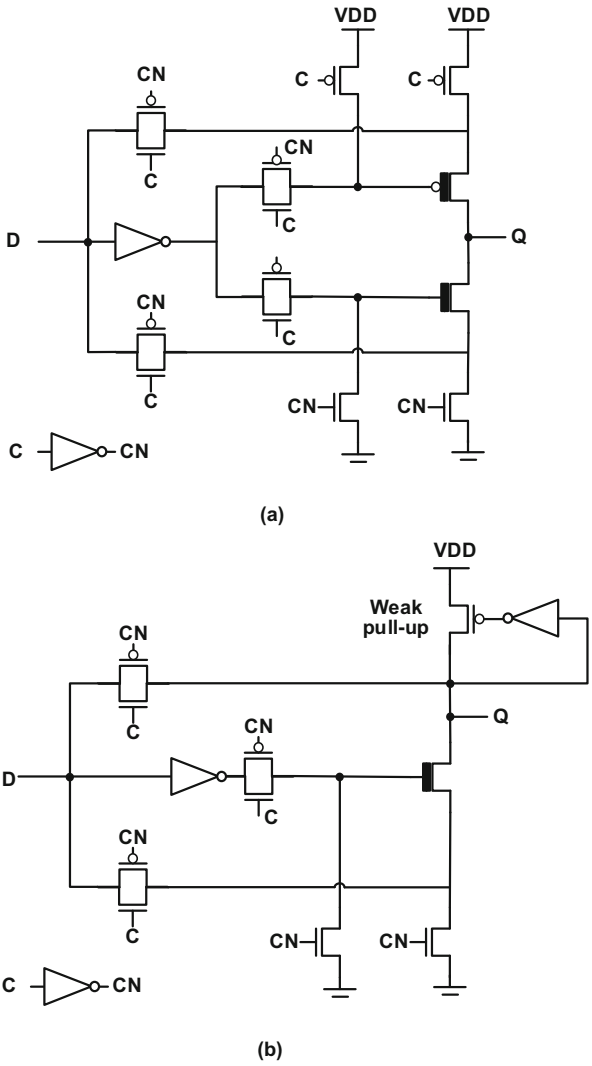


Fig. 12. NCFET nonvolatile latches (NCFET transistors are drawn with thick gates).

With the synergy of the low-voltage NCFET logic [35], NCFET nonvolatile memory array [18], and the NCFET NV-DFF, an energy-efficient NVP is designed, as shown in Fig. 13(b), in comparison with a conventional NVP in Fig. 13(a). As both logic and memory are intrinsically nonvolatile, there is no need for backup and restore controls for the NCFET storage. The baseline design uses conventional CMOS transistors for logic, a clustered FeRAM array as data and instruction memory, and NV-DFF using ferroelectric capacitor for state backup under external control [56].

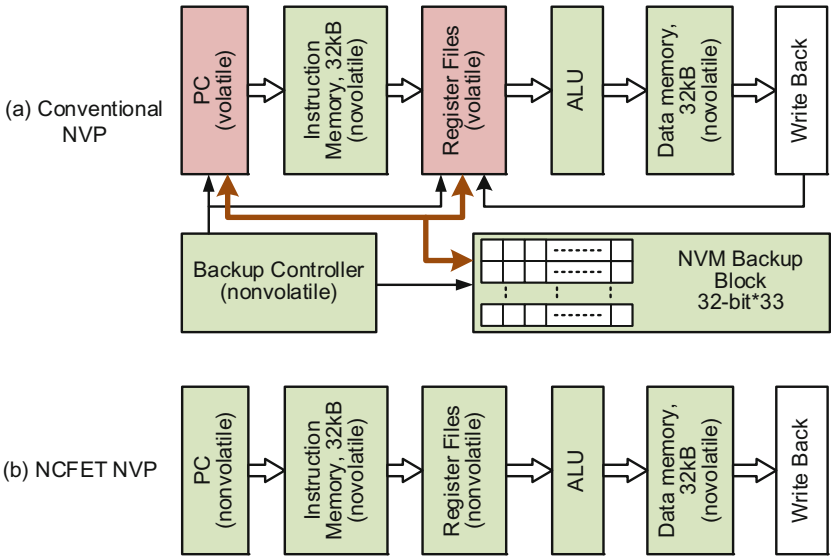


Fig. 13. NVP design using NCFET logic and memory

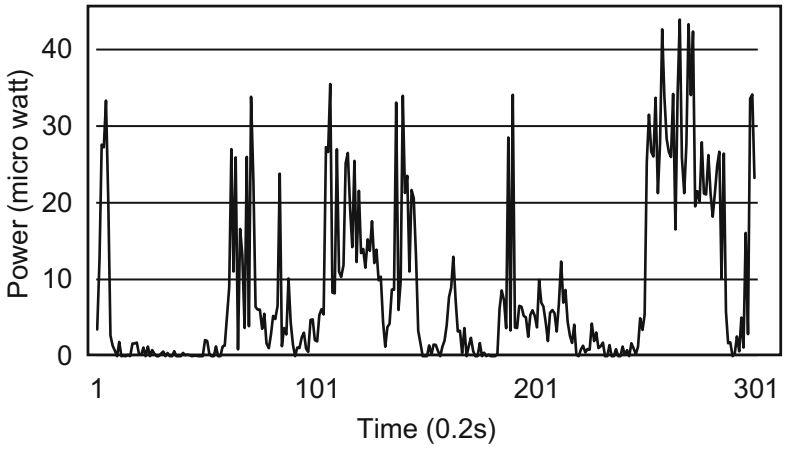


Fig. 14. Input power profile in the test

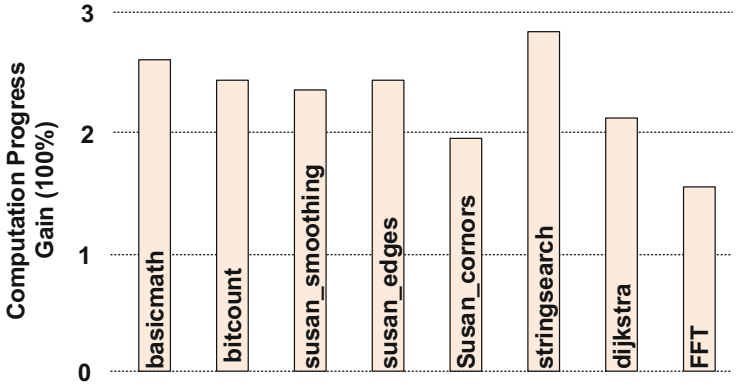


Fig. 15. Computation progress gain for various test benches

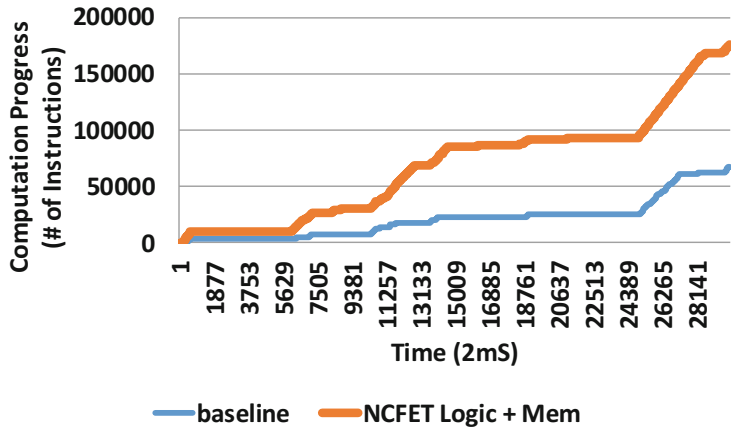


Fig. 16. Comparison between NCFET NVP and the baseline NVP of computation progress versus time (Tesebench: basic math).

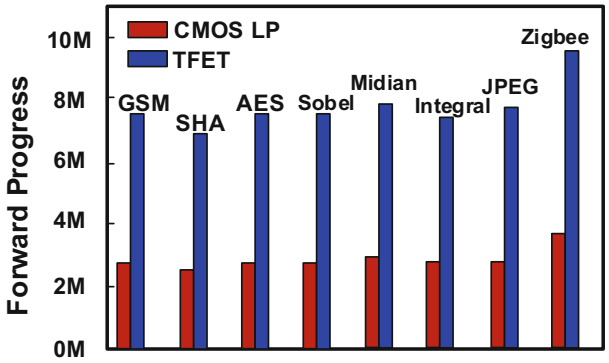


Fig. 17. TFET NVP with more forward progress for various tasks (source: [57]).

Giving the harvested RF power from TV stations, as shown in the power profile sampled per 0.2S in Fig. 14, the average input power is 8.7 μW, with the peak up to ~45 μW and frequent power failures with power lower than nW. The simulations are carried out in various test cases in MiBench [55]. Figure 15 shows the simulation results for these testbenches. Figure 16 shows the comparison of computation progress versus time. The computation progress gain ranges between 1.5× to 2.8×, which confirms the benefit of using NCFET for NVP design.

When only TFET is used to replace CMOS in NVP design, improvement of computation progress is also observed, as shown in Fig. 17 [57]. The improvement comes from the energy savings by low-power digital logic and less number of backup/restore operations.

4.5 Nonvolatile Computing Architectures

In this new NVP design regime, most existing guidance of low-power design techniques are still useful, but there has also been significant difference in the design and optimization methodology.

The first different rule is that "design for low power does not guarantee more computing progress" [13]. This is because, in a battery-less energy-harvesting IoT system, there is no ideal temporary energy storage, and the harvested energy will be wasted in the form of overflowing or leaking if it is not used efficiently in time. In other words, the computing forward progress (CFP) indicated by the number of executed instructions (NI) could be expressed as a function of computation energy (CE) and energy-per-instruction (EPI):

$$\mathrm{CFP} = \mathrm{CE}/\mathrm{EPI}, \tag{2}$$

where CE is a fraction of the total harvested energy, considering the energy loss from backup/restore operations, leakage and overflow in the energy storage capacitor, and leakage in the circuits.

And the fact is that the lowest EPI does not guarantee the highest CE because of the abovementioned energy harvesting and storage non-idealities. For example, an out-of-order processor may contribute to more forward progress than a non-pipelined processor in scenarios when the harvested power is higher. As a result, the power of the processor should adapt to the harvester and energy storage status to find the best trade-off between the lowest EPI and the most CE.

There are various approaches to configuring the processor so as to fit the input power trace. One approach is to dynamically switch between different processing cores which are all embedded on the same chip based on level of harvested power and the store energy [49]. The second approach is to dynamically scale the operating frequency and voltage (DVFS) accordingly [53, 54]. The third approach is to dynamically re-allocate computing and storage resources for the processor which turns out to be a different degree of parallelism [54].

In addition to the trade-off between CE and EPI, there are other optimizations that have significant impact on the overall forward progress.

The first consideration is "what" and "when" to back up the computing states. There are various reasons that make this consideration important. First of all, less amount of backup states needs less backup energy but needs more energy to recover and re-compute; Secondly, a backup operation which is carried out too early may be a complete waste of energy and time as it may not be necessary at all, while a too-late backup may lead to backup failure and progress rolling-backup. Also, there can be a risk to take, on how much energy that could be harvested in the future – which could also be counted into a certain amount of usable energy.

The second consideration is how to understand the feature of the harvested energy, and how to predict its trend. An accurate prediction of the input power will certainly help system configuration for more forward progress. For some energy sources such as ambient RF energy, the harvested power varies radically and is challenging to predict. Meanwhile, for some other energy sources, such as motion and solar energy sources, the harvested power has a certain pattern and could be predicted. Machine learning techniques have been proposed to predict the future energy to assist dynamic system configuration for more forward progress [49, 54].

The third consideration is on-chip NVM optimization. There are a few key factors that must be considered. One factor is what types of on-chip memory to use for backup operations. Different NVM devices, such as ReRAM, FeRAM, STT-RAM, and the emerging NCFET NVM, etc., have different energy-delay performance for read and write operations. Another factor is to use centered (aka clustered) or distributed memory. Distributed memory uses a local nearby NVM bit storage close to each DFF with a copy of access interface circuit. Clustered memory is implemented with arrays of memory and could be dense in area due to shared elements such as sense amplifiers but may consume more energy and delay in access due to longer interconnection lines and limited degree of access parallelism.

5 Future Work for IoT Using Emerging Devices

While emerging devices have shown great potential for future energy-efficient IoT applications, there is still a large gap between what has been experimentally demonstrated and a complete system implementation and application mapping. Significant efforts from all the levels of device, circuits, system and applications are required to speed up the progress [58].

Device understanding, characterization, and integrated fabrication: Continuous optimization of material and process is required for large-scale integration. It is a key to build accurate models of emerging devices that support more aspects of devices features, such as matching, noise, endurance, parasitics, etc., for circuit and higher level simulations;

Circuit and architecture optimizations: It is unlikely for emerging device features to be used as a drop-in replacement for all conventional CMOS techniques. Innovative circuit topology re-design and optimization are sometimes a must to obtain the desired circuit functionality and performance, which also brings additional trade-offs to carry out. Circuit and architecture optimizations to make the most use of pros and mitigate

cons of emerging devices are necessary [13–15, 46]. Meanwhile, device features deviating from conventional CMOS behavior may actually be very useful in some applications, highlighting the necessity of device-circuit-application co-design.

Higher-level considerations: Quality-of-service (QoS) and task scheduling optimization, with support from software design are also an area of key interest [47, 51, 52]. Security, privacy, and communication protocols are core concerns in any IoT deployment. The study of the interaction of quality and security metrics with design and power efficiency optimizations requires further investigations from device to architecture to software ecosystem.

The exploration of emerging devices, circuits, and architectures should be a joint effort. It is impossible to dig into all emerging devices for all different types of applications. Efforts spent for emerging device modeling and benchmarking may not be meaningful if the device finally turns out to be far from satisfactory. Moreover, research on modeling and higher-level design needs strong support from device developing groups and continuous interactions with them are crucial to ensure that each is aware of the newest findings and phenomena understandings in the other's domain.

6 Conclusion

This chapter has discussed new opportunities in Internet-of-Things enabled by emerging devices, circuits and architectures through enhanced and new features to the implementations. The future work for IoT based on emerging technologies is also discussed.

Acknowledgements. This work was supported in part by the Center for Low Energy Systems Technology (LEAST), one of the six SRC STARnet centers, sponsored by MARCO and DARPA, by NSF awards 1160483 (ASSIST), and NSF Expeditions in Computing Award-1317560.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Networks **54**, 2787–2805 (2010)
2. Li, X., Heo, U.D., Ma, K., Narayanan, V., Liu, H., Datta, S.: Rf-powered systems using steep-slope devices. In: 2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS) (2014)
3. Kim, S., Vyas, R., Bito, J., Niotaki, K., Collado, A., Georgiadis, A., Tentzeris, M.M.: Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms. Proc. IEEE **102**, 1649–1666 (2014)
4. Nikonov, D.E., Young, I.A.: Overview of beyond-CMOS devices and a uniform methodology for their benchmarking. Proc. IEEE **101**, 2498–2533 (2013)
5. Lu, L., Li, X., Narayanan, V., Datta, S.: A reconfigurable low-power BDD logic architecture using ferroelectric single-electron transistors. IEEE Trans. Electron Devices **62**(3), 1052–1057 (2015). doi:10.1109/ted.2015.2395252

6. Roy, K., Sharad, M., Fan, D., Yogendra, K.: Computing with spin-transfer-torque devices: prospects and perspectives. In: 2014 IEEE Computer Society Annual Symposium on VLSI (2014)
7. Seabaugh, A.C., Zhang, Q.: Low-voltage tunnel transistors for beyond CMOS logic. Proc. IEEE **98**, 2095–2110 (2010)
8. Khan, A.I., Yeung, C.W., Hu, C., Salahuddin, S.: Ferroelectric negative capacitance MOSFET: capacitance tuning & antiferroelectric operation. In: 2011 International Electron Devices Meeting (2011)
9. Swaminathan, K., Liu, H., Li, X., Kim, M.S., Sampson, J., Narayanan, V.: Steep slope devices: enabling new architectural paradigms. In: Proceedings of the 51st Annual Design Automation Conference on Design Automation Conference - DAC 2014 (2014)
10. Brito, M.A.G.D., Galotto, L., Sampaio, L.P., Melo, G.D.A.E., Canesin, C.A.: Evaluation of the main MPPT techniques for photovoltaic applications. IEEE Trans. Ind. Electron. **60**, 1156–1167 (2013)
11. Liu, H., Li, X., Vaddi, R., Ma, K., Datta, S., Narayanan, V.: Tunnel FET RF rectifier design for energy harvesting applications. IEEE J. Emerg. Sel. Top. Circuits Syst. **4**, 400–411 (2014)
12. Heo, U., Li, X., Liu, H., Gupta, S., Datta, S., Narayanan, V.: A high-efficiency switched-capacitance HTFET charge pump for low-input-voltage applications. In: 2015 28th International Conference on VLSI Design (2015)
13. Ma, K., Zheng, Y., Li, S., Swaminathan, K., Li, X., Liu, Y., Sampson, J., Xie, Y., Narayanan, V.: Architecture exploration for ambient energy harvesting nonvolatile processors. In: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA) (2015)
14. Ma, K., Li, X., Li, S., Liu, Y., Sampson, J.J., Xie, Y., Narayanan, V.: Nonvolatile processor architecture exploration for energy-harvesting applications. IEEE Micro. **35**, 32–40 (2015)
15. Ma, K., Li, X., Swaminathan, K., Zheng, Y., Li, S., Liu, Y., Xie, Y., Sampson, J.J., Narayanan, V.: Nonvolatile processor architectures: efficient, reliable progress with unstable power. IEEE Micro. **36**, 72–83 (2016)
16. Liu, V., Parks, A., Talla, V., Gollakota, S., Wetherall, D., Smith, J.R.: Ambient backscatter. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM - SIGCOMM 2013 (2013)
17. Ueki, M., Takeuchi, K., Yamamoto, T., Tanabe, A., Ikarashi, N., Saitoh, M., Nagumo, T., Sunamura, H., Narihiro, M., Uejima, K., Masuzaki, K., Furutake, N., Saito, S., Yabe, Y., Mitsuiki, A., Takeda, K., Hase, T., Hayashi, Y.: Low-power embedded ReRAM technology for IoT applications. In: 2015 Symposium on VLSI Technology (VLSI Technology) (2015)
18. George, S., Gupta, S., Narayanan, V., Ma, K., Aziz, A., Li, X., Khan, A., Salahuddin, S., Chang, M.-F., Datta, S., Sampson, J.: Nonvolatile memory design based on ferroelectric FETs. In: Proceedings of the 53rd Annual Design Automation Conference on - DAC 2016 (2016)
19. Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y., Xie, Y.: PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. In: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA) (2016)
20. Roman, R., Najera, P., Lopez, J.: Securing the internet of things. Computer **44**, 51–58 (2011)
21. Kim, M.S., Liu, H., Swaminathan, K., Li, X., Datta, S., Narayanan, V.: Enabling power-efficient designs with III-V tunnel FETs. In: 2014 IEEE Compound Semiconductor Integrated Circuit Symposium (CSICS) (2014)

22. Liu, H., Shoaran, M., Li, X., Datta, S., Schmid, A., Narayanan, V.: Tunnel FET-based ultra-low power, low-noise amplifier design for bio-signal acquisition. In: Proceedings of the 2014 International Symposium on Low Power Electronics and Design - ISLPED 2014 (2014)
23. Tsai, W.-Y., Liu, H., Li, X., Narayanan, V.: Low-power high-speed current mode logic using Tunnel-FETs. In: 2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC) (2014)
24. Kim, M.S., Liu, H., Li, X., Datta, S., Narayanan, V.: A steep-slope tunnel FET based SAR analog-to-digital converter. IEEE Trans. Electron Devices **61**, 3661–3667 (2014)
25. Kim, M.S., Li, X., Liu, H., Sampson, J., Datta, S., Narayanan, V.: Exploration of low-power High-SFDR current-steering D/A converter design using steep-slope heterojunction tunnel FETs. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **24**(6), 2299–2309 (2016)
26. Salahuddin, S., Datta, S.: Use of negative capacitance to provide voltage amplification for low power nanoscale devices. Nano Lett. **8**, 405–410 (2008)
27. Hu, C., Salahuddin, S., Lin, C.-I., Khan, A.: 0.2 V adiabatic NC-FinFET with 0.6 mA/μm ION and 0.1nA/μm IOFF. In: 2015 73rd Annual Device Research Conference (DRC) (2015)
28. Lee, M.H., Wei, Y.-T., Chu, K.-Y., Huang, J.-J., Chen, C.-W., Cheng, C.-C., Chen, M.-J., Lee, H.-Y., Chen, Y.-S., Lee, L.-H., Tsai, M.-J.: Steep slope and near non-hysteresis of FETs With antiferroelectric-like HfZrO for low-power electronics. IEEE Electron Device Lett. **36**, 294–296 (2015)
29. Jo, J., Choi, W.Y., Park, J.-D., Shim, J.W., Yu, H.-Y., Shin, C.: Negative capacitance in organic/ferroelectric capacitor to implement steep switching MOS devices. Nano Lett. **15**, 4553–4556 (2015)
30. Khan, A.I., Chatterjee, K., Duarte, J.P., Lu, Z., Sachid, A., Khandelwal, S., Ramesh, R., Hu, C., Salahuddin, S.: Negative capacitance in short-channel FinFETs externally connected to an epitaxial ferroelectric capacitor. IEEE Electron Device Lett. **37**, 111–114 (2016)
31. Jo, J., Shin, C.: Negative capacitance field effect transistor with hysteresis-free Sub-60-mV/Decade switching. IEEE Electron Device Lett. **37**, 245–248 (2016)
32. Li, K.-S., Chen, P.-G., Lai, T.-Y., Lin, C.-H., Cheng, C.-C., Chen, C.-C., Wei, Y.-J., Hou, Y.-F., Liao, M.-H., Lee, M.-H., Chen, M.-C., Sheih, J.-M., Yeh, W.-K., Yang, F.-L., Salahuddin, S., Hu, C.: Sub-60 mV-swing negative-capacitance FinFET without hysteresis. In: 2015 IEEE International Electron Devices Meeting (IEDM) (2015)
33. Lee, M.H., Chen, P.-G., Liu, C., Chu, K.-Y., Cheng, C.-C., Xie, M.-J., Liu, S.-N., Lee, J.-W., Huang, S.-J., Liao, M.-H., Tang, M., Li, K.-S., Chen, M.-C.: Prospects for ferroelectric HfZrOx FETs with experimentally CET = 0.98 nm, SSfor = 42 mV/dec, SSrev = 28 mV/dec, switch-off 0.2 V, and hysteresis-free strategies. In: 2015 IEEE International Electron Devices Meeting (IEDM) (2015)
34. George, S., Aziz, A., Li, X., Datta, S., Sampson, J., Gupta, S., Narayanan, V.: NCFET based logic for energy harvesting systems. In: SRC TECHCON 2015 (2015)
35. George, S., Aziz, A., Li, X., Kim, M.S., Datta, S., Sampson, J., Gupta, S., Narayanan, V.: Device circuit co design of FEFET based logic for low voltage processors. In: 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) (2016)
36. Aziz, A., Ghosh, S., Datta, S., Gupta, S.: Physics-based circuit-compatible SPICE model for ferroelectric transistors. IEEE Electron Device Lett. **37**, 1 (2016)
37. Asbeck, P.M., Lee, K., Min, J.: Projected performance of heterostructure tunneling FETs in low power microwave and mm-wave applications. IEEE J. Electron Devices Soc. **3**, 122–134 (2015)
38. Datta, S., Bijesh, R., Liu, H., Mohata, D., Narayanan, V.: Tunnel transistors for energy efficient computing. In: 2013 IEEE International Reliability Physics Symposium (IRPS) (2013)

39. Saripalli, V., Datta, S., Narayanan, V., Kulkarni, J.P.: Variation-tolerant ultra low-power heterojunction tunnel FET SRAM design. In: 2011 IEEE/ACM International Symposium on Nanoscale Architectures (2011)
40. Kim, M.S., Cane-Wissing, W., Li, X., Sampson, J., Datta, S., Gupta, S.K., Narayanan, V.: Comparative area and parasitics analysis in FinFET and heterojunction vertical TFET standard cells. ACM J. Emerg. Technol. Comput. Syst. **12**, 1–23 (2016)
41. Swaminathan, K., Liu, H., Sampson, J., Narayanan, V.: An examination of the architecture and system-level tradeoffs of employing steep slope devices in 3D CMPs. ACM SIGARCH Comput. Archit. News. **42**, 241–252 (2014)
42. Wang, Y., Liu, Y., Li, S., Zhang, D., Zhao, B., Chiang, M.-F., Yan, Y., Sai, B., Yang, H.: A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In: 2012 Proceedings of the ESSCIRC (ESSCIRC) (2012)
43. Pandey, R., Madan, H., Liu, H., Chobpattana, V., Barth, M., Rajamohanan, B., Hollander, M.J., Clark, T., Wang, K., Kim, J.- H., Gundlach, D., Cheung, K.P., Suehle, J., Engel-Herbert, R., Stemmer, S., Datta, S.: Demonstration of p-type In0.7Ga0.3As/-GaAs0.35Sb0.65 and n-type GaAs0.4Sb0.6/In0.65Ga0.35As complimentary Heterojunction Vertical Tunnel FETs for ultra-low power logic. In: 2015 Symposium on VLSI Technology (VLSI Technology) (2015)
44. Rajamohanan, B., Pandey, R., Chobpattana, V., Vaz, C., Gundlach, D., Cheung, K.P., Suehle, J., Stemmer, S., Datta, S.: 0.5 V supply voltage operation of In0.65Ga0.35As/GaAs0.4Sb0.6Tunnel FET. IEEE Electron Device Lett. **36**, 20–22 (2015)
45. Morita, Y., Mori, T., Fukuda, K., Mizubayashi, W., Migita, S., Matsukawa, T., Endo, K., O'uchi, S., Liu, Y., Masahara, M., Ota, H.: Experimental realization of complementary p- and n- tunnel FinFETs with subthreshold slopes of less than 60 mV/decade and very low (pA/μm) off-current on a Si CMOS platform. In: 2014 IEEE International Electron Devices Meeting (2014)
46. Liu, Y., Yang, H., Wang, Y., Wang, C., Sheng, X., Li, S., Zhang, D., Sun, Y.: Power system design and task scheduling for photovoltaic energy harvesting based nonvolatile sensor nodes. In: Lin, Y.-L., et al. (eds.) Smart Sensors and Systems, pp. 243–277. Springer, Cham (2015)
47. Zhang, D., Liu, Y., Li, J., Xue, C.J., Li, X., Wang, Y., Yang, H.: Solar power prediction assisted intra-task scheduling for nonvolatile sensor nodes. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **35**, 724–737 (2016)
48. Liu, Y., Wang, Z., Lee, A., Su, F., Lo, C.-P., Yuan, Z., Lin, C.-C., Wei, Q., Wang, Y., King, Y.-C., Lin, C.-J., Khalili, P., Wang, K.-L., Chang, M.-F., Yang, H.: 4.7 A 65 nm ReRAM-enabled nonvolatile processor with 6 × reduction in restore time and 4 × higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic. In: 2016 IEEE International Solid-State Circuits Conference (ISSCC) (2016)
49. Ma, K., Li, X., Liu, Y., Sampson, J., Xie, Y., Narayanan, V.: Dynamic machine learning based matching of nonvolatile processor microarchitecture to harvested energy profile. In: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) (2015)
50. Li, Q., Zhao, M., Hu, J., Liu, Y., He, Y., Xue, C.J.: Compiler directed automatic stack trimming for efficient non-volatile processors. In: Proceedings of the 52nd Annual Design Automation Conference on - DAC 2015 (2015)
51. Xie, M., Pan, C., Hu, J., Yang, C., Chen, Y.: Checkpoint-aware instruction scheduling for nonvolatile processor with multiple functional units. In: The 20th Asia and South Pacific Design Automation Conference (2015)

52. Wang, Y., Liu, Y., Wang, C., Li, Z., Sheng, X., Lee, H.G., Chang, N., Yang, H.: Storage-Less and Converter-Less Photovoltaic Energy Harvesting with Maximum Power Point Tracking for Internet of Things. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **35**, 173–186 (2016)
53. Ma, K., Li, X., et al.: Dynamic power and energy management for energy harvesting nonvolatile processor systems. ACM Trans. Embed. Comput. Syst. (TECS) **16**(4), 107:1–107:23 (2017)
54. Ma, K., Li, X., et al.: Spendthrift: machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors. In: ASP-DAC (2017)
55. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., Brown, R.B.: MiBench: a free, commercially representative embedded benchmark suite. In: Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538), pp. 3–14 (2001)
56. Kimura, H., Fuchikami, T., Marumoto, K., Fujimori, Y., Izumi, S., Kawaguchi, H., Yoshimoto, M.: A 2.4 pJ ferroelectric-based non-volatile flip-flop with 10-year data retention capability. In: 2014 IEEE Asian Solid-State Circuits Conference (A-SSCC), KaoHsiung, pp. 21–24 (2014)
57. Ma, K., Li, X., Sampson, J., Xie, Y., Liu, Y., Narayanan, V.: Nonvolatile processor optimization for ambient energy harvesting scenarios. In: The 15th Non-volatile Memory Technology Symposium (NVMTS), pp. 1–3 (2015)
58. Li, X., Ma, K., George, S., Sampson, J., Narayanan, V.: Enabling Internet-of-Things: Opportunities brought by emerging devices, circuits, and architectures. In: 2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Tallinn, pp. 1–6 (2016)

Logic with Unipolar Memristors – Circuits and Design Methodology

Nimrod Wald(✉), Elad Amrani, Avishay Drori, and Shahar Kvatinsky(✉)

Faculty of Electrical Engineering, Technion – Israel Institute of Technology, 3200003 Haifa, Israel
nimrodw@tx.technion.ac.il, shahar@ee.technion.ac.il

Abstract. Memristors are a general name for a set of emerging resistive switching technologies. These two terminal devices are characterized by a varying resistance, which is controlled by the voltage or current applied to them. The resistance state of a memristor is nonvolatile, and as such makes memristors attractive candidates for use as novel memory elements. Apart from their use for memory applications, the use of memristors in logic circuits is widely researched. A class of logic circuits named 'stateful logic', where the logic state of the inputs and outputs is stored in the form of resistance, is a promising approach for carrying out logic computations within memory. This chapter discusses the use of non-polar memristors, a type of memristors whose resistance depends only on the magnitude of the voltage across its terminals, for performing stateful logic operations. A design methodology is presented to allow structured development of stateful logic gates, and backed by a demonstration of the design process of OR and XOR gates using non-polar memristors.

Keywords: Memristor · Unipolar memristors · Resistive switch · Logic design · Design methodology · Stateful logic · In-memory computing · mMPU

1 Introduction

Memristor is a general term for a family of emerging technologies [1, 2], including metal oxide thin film resistive switches (RRAM or ReRAM) [3], spin torque transfer magneto-resistive RAM (STT-MRAM) [4] and phase change memory (PCM) [5]. The electrical properties of memristors were formulated in 1971 by Leon Chua [6] in an effort to achieve a symmetric relation between the known electric quantities of voltage, current, electric charge and magnetic flux. The research of memristors has been dormant from that time, until in 2008 researchers at Hewlett Packard (HP) laboratories have linked the known phenomenon of resistive switching to memristors [7]. Since then, research of memristors is being performed in the fields of memory, neuromorphic circuits [8], hardware security [9, 10] and logic [11]. Memristors are characterized by an intrinsic state variable, which determines the device resistance (sometimes called memristance), varying from a low resistance state (*LRS*, R_{ON}) to a high resistance state (*HRS*, R_{OFF}). The state variable represents the physical switching mechanism

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 24–40, 2017.
DOI: 10.1007/978-3-319-67104-8_2

(*e.g.* filament forming state in RRAM devices), and changes its value according to the current or voltage applied to the device.

Increasing power dissipation due to leakage in transistors as they are being shrunk is motivation for use of novel non-volatile devices for performing logic operations. Furthermore, the fact that processor performance increase greatly outpaces that of memories, causes a bottleneck named '*the memory wall*', meaning that most energy and latency of computations is spent on moving data between the CPU and memory [12]. Using memristors, natural candidates for replacing conventional memory technologies, as logic elements could solve this problem by performing the logic operations within the memory, eliminating much of the need for fetching data. The combination of data storage and processing in a single element enables the design of memristive memory processing unit (mMPU) [13, 14]. Many methods for performing logic operations using memristors have been previously proposed, including memristor ratioed logic (MRL) [15], Akers logic arrays [16], complementary resistive switching (CRS) [17], implication logic (IMPLY) [18], and memristor-aided logic (MAGIC) [19]. The latter two utilize the state of memristors as the logic value of both inputs and output. This method is known as '*stateful logic*' and is especially suited for performing logic within memory arrays [20, 21].

This chapter discusses the implementation of logic circuits using a more uncommon type of memristors, namely unipolar (or non-polar) memristors. The characteristics of these memristors are covered in Sect. 2, and an example for the use of such devices for logic design is presented in Sect. 3. A design methodology for developing stateful memristive logic gates with any type of memristors is described in Sect. 4, followed by another example of a unipolar memristive logic gate design in Sect. 5, pursuing the proposed methodology. All simulations are conducted using an internally developed VerilogA model for unipolar memristors, based on [22]. The chapter is concluded in Sect. 6.

2 Unipolar Memristors

The majority of research in the field of memristive logic concentrates on the use of bipolar memristors. These devices have a state variable that changes its value according to both the magnitude and polarity of the voltage. Thus, applying a positive voltage higher than a certain threshold V_{RESET} increases the resistance of the device up to HRS, and applying a negative voltage exceeding a negative threshold voltage V_{SET} lowers the resistance down to LRS. This work deals with the use of a different memristor, the unipolar memristor, which differs from bipolar memristors in the fact that only the magnitude of the voltage across the device determines the change in the resistance. Thus, applying a voltage higher than $|V_{RST}|$ across the device in any direction increases the resistance. Applying a voltage higher than a different threshold (*e.g.*, $|V_{SET}| > |V_{RESET}|$) causes the resistance to drop. Once a device is switched to LRS, a compliance current limitation is usually necessary to avoid excess current that damages the device. Resistive switching technologies that result in unipolar switching behavior include PCM and some of RRAM technologies with thermochemical mechanism [23–27]. Examples for I-V curves of both bipolar and unipolar memristors are shown in Fig. 1.

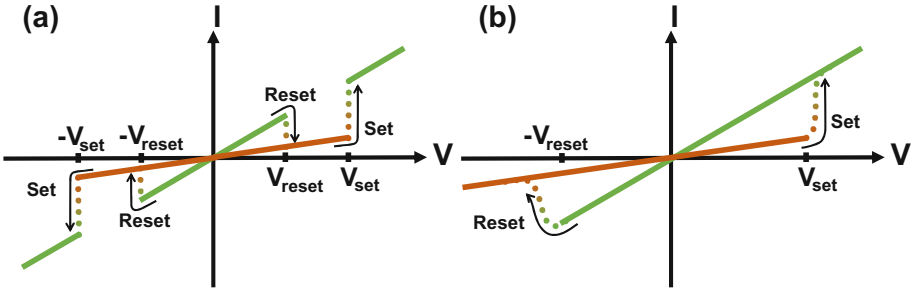


Fig. 1. I-V curves for unipolar (a) and bipolar (b) memristors. The regions in which the device is in LRS are in green, the ones in HRS are in orange, and the dotted lines are transitions between the two. (Color figure online)

We define the logic values stored in a memristor in the following manner, HRS is denoted as logical '0' and LRS as logical '1'. The use of unipolar memristors for logic gates opens the possibility of performing computation within memristive arrays of types previously not considered for use as logic. Furthermore, the use of unipolar memristors allows designing simpler controllers and voltage sources due to the fact that only a single voltage polarity is required for switching back and forth.

3 A Unipolar Memristive Logic Gate Example

In this section, a concept to design logic gates with unipolar memristors is presented [28]. The operation mechanism is first presented, followed by examples of OR and NOT gates.

3.1 Operation Principle

The basic mechanism of the proposed logic technique is a voltage divider between two resistive elements: a memristor and a resistor for a NOT gate or two memristors for an OR gate. The proposed circuits are based on connecting two resistive elements in series and applying a voltage bias. The ratio of voltages on the two elements complies with the ratio of their resistance, *i.e.*, the states are distinguished using a bias voltage. The first step of operation is translating resistance to resistive states. The applied voltage for distinction is called the *preset voltage*.

After state distinction has been achieved, a higher voltage is applied to the circuit, adding higher applied voltage across both elements, regardless of their states. The voltage in this step is predetermined to a value that promotes switching if necessary for proper execution, thus this voltage is called the *evaluation voltage*. The operation is therefore comprised of two execution steps: *preset* and *switching*.

One obstacle to operate properly arises from the fact that every change in resistance immediately changes the voltages, hence, possibly changing the distinction between states. This phenomenon may lead to an incorrect result. Therefore, maintaining the initial voltage distinction for a sufficient time is required to reach the desired resistance

(HRS or LRS). One possible solution is to incorporate capacitors in the circuit in parallel with each resistive device. The capacitors add delay to the system due to the need to charge/discharge them during operation. Thus, we call them *suspension capacitors*. In addition to prolonging the validity of voltage values in the switching stage, suspension capacitors also delay the preset stage and in the case of the NOT gate, are actually mandatory for proper operation. Furthermore, the transition from preset to switching stages cannot be instantaneous. Hence, the intermediate evaluation stage is abstractly depicted as a transitive state and three stages are used to execute the operation as illustrated in Fig. 2.

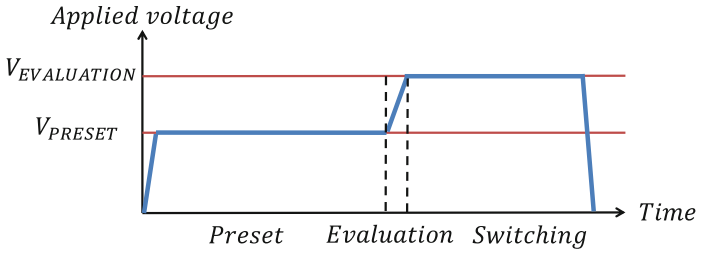


Fig. 2. The sequence of the applied voltage for the three stages of a general logic operation. The preset voltage distinguishes between logical states and charges the suspension capacitors. The evaluation stage converts the preceding voltages to the required voltages for switching.

(a) Preset Stage

In the preset stage, a voltage V_{PRESET} is applied to the circuit to charge the capacitors and initialize the voltage division between the resistive devices. The applied voltage is sufficiently high to distinguish between resistive states, but lower than the threshold voltage, thus does not change the state of the memristors. After sufficient time, approximately no current passes through the capacitors and their voltages are consistent with the voltage divider.

(b) Evaluation Stage

The evaluation stage starts immediately after the preset stage. A voltage pulse $V_{EVALUATION}$ is applied to the circuit. The purpose of this stage is to increase the voltage on both resistive elements abruptly. The final voltage in this stage depends on the final voltage of the preset stage, hence correlates with the resistance of the circuit elements. However, the voltage increase $V_{EVALUATION} - V_{PRESET}$ is fixed for all scenarios. The exact increase in voltage after the voltage jump is determined by the capacitance ratio (charge sharing).

(c) Switching Stage

In the switching stage, $V_{EVALUATION}$ is still applied for sufficient time to allow switching of the memristors. The key is to choose proper pulse length and voltage magnitude to switch the memristors according to the desired logical functionality.

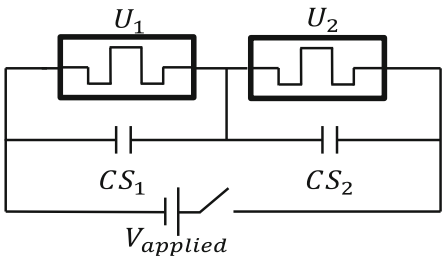


Fig. 3. Schematic of an OR gate. The input memristors U_1, U_2 are overwritten with the output.

3.2 OR Gate

A two-input OR gate consists of two unipolar memristors U_1 and U_2 connected in series. A suspension capacitor is connected in parallel to each memristor, as shown in Fig. 3. The initial logical state of the memristors is the input of the gate and after execution both memristors have the same logical state, which serves as the output of the gate.

Assume $V_{SET} > V_{RESET}$, for proper behavior of the gate certain conditions need to be fulfilled. First, when both inputs are identical (*i.e.*, both are logical '1' or '0') there is no memristor switching. Second, when the inputs are different, the HRS memristor (in logical '0') has to switch to LRS since the desired output is logical '1'. Assuming that the voltage on the HRS memristor equals V_{PRESET} in the preset stage and $V_{PRESET} + \frac{1}{2}(V_{EVALUATION} - V_{PRESET})$ in the evaluation stage; the constraints on the voltages are therefore

$$V_{PRESET} < 2V_{RESET}, \tag{1a}$$

$$2V_{SET} - V_{PRESET} < V_{EVALUATION} < 2V_{RESET}. \tag{1b}$$

Figure 4 shows simulation results of an OR gate for the case where the inputs are different and U_2 switches for proper result. Note that when U_1 is logical '0' and U_2 is logical '1', the operation is destructive, *i.e.*, the value of the inputs is overwritten.

3.3 NOT Gate

The NOT gate consists of a single unipolar memristor connected in series with a reference resistor. The memristor acts as both input and output of the NOT gate. For proper operation both the memristor and the resistor have a suspension capacitor connected to them in parallel as shown in Fig. 5. Without the suspension capacitors, $V_{EVALUATION}$ must be absurdly high to allow switching the memristor in the case of RESET operations. The resistance of the reference resistor is between LRS and HRS. This value ensures that the voltage at the end of the preset stage across a HRS (LRS) memristor is high (low), as illustrated in Fig. 6. A reasonable choice is $R_{REF} = \sqrt{R_{OFF}R_{ON}}$. For proper operation, the conditions on the applied voltage are

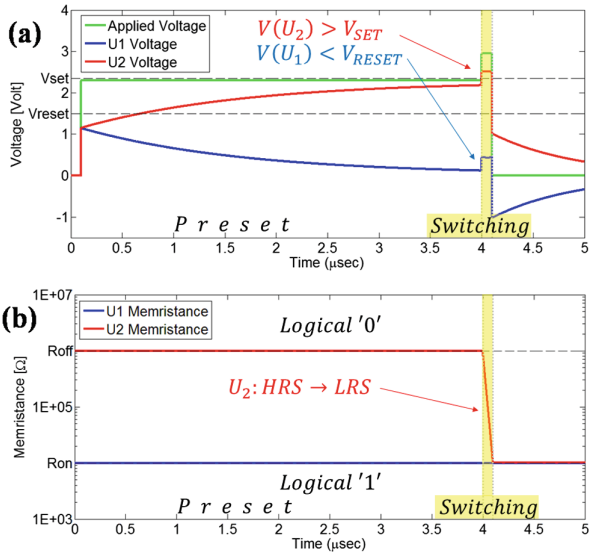


Fig. 4. OR gate simulation results. U_1 and U_2 are initialized to, respectively, LRS (logical '1') and HRS (logical '0'). (a) Voltages across the memristors during the operation, and (b) their resistance. In the first 4 μs the system is in the preset stage, and the capacitors are charged/discharged to distinctive voltages. In the switching stage, U_2 voltage is higher than V_{set} for sufficient time and its logical value is switched to logical '1' as desired.

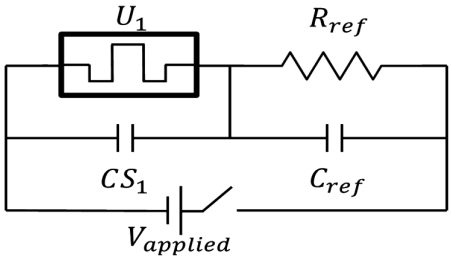


Fig. 5. Schematic of a NOT gate. A resistor is used as a reference to determine the state of the memristor

$$V_{PRESET} < min\left\{\sqrt{\frac{R_{OFF}}{R_{ON}}}V_{RESET}, V_{SET}\right\}, \tag{2a}$$

$$\frac{1}{\gamma} max\{V_{SET}, V_{RESET} + V_{PRESET}\} < V_{EVALUATION}, \tag{2b}$$

$$V_{EVALUATION} < \frac{1}{\gamma}(V_{SET} + V_{PRESET}), \tag{2c}$$

where $\gamma \triangleq \frac{C_{REF}}{C_{REF} + C_{S_1}}$.

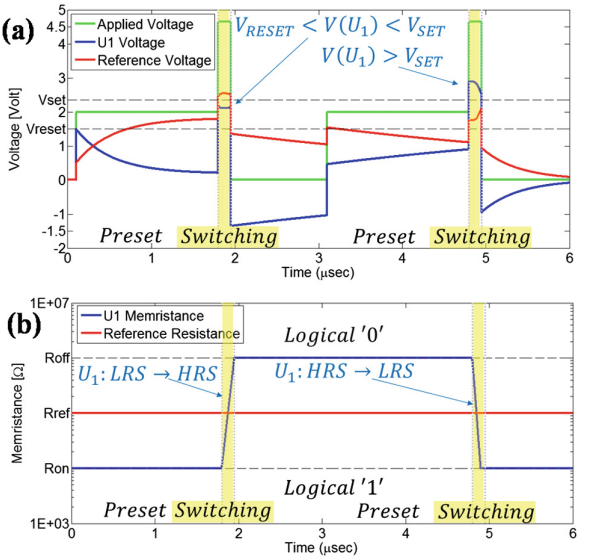


Fig. 6. NOT gate simulation results. (a) Voltages and (b) resistance during two consecutive memristor switching. In the first 3 μs, U_1 switches from $LRS \rightarrow HRS$. In the second NOT operation U_1 switches back to *LRS*.

3.4 Timing Considerations

One of the critical points for proper behavior of the proposed logic technique is to apply the right voltage for a sufficient time during the switching stage. In this section, the timing constraints in the switching stage are explored. Assume $\tau_{SET}(\tau_{RESET})$ is a minimal transition time from HRS (LRS) to LRS (HRS) [29]. For successful switching, the duration of the switching stage must be greater than the minimal required switching time. The minimum condition on the length of the stage is therefore

$$T_{pulse} > max\{\tau_{set}, \tau_{reset}\} = T_{pulse,min}. \tag{3}$$

At the beginning of the switching stage, each memristor is biased with a voltage which promotes switching (if necessary). The validity of the specified voltage level is maintained for a short period of time, due to the use of suspension capacitors, but will eventually become invalid. If the switching stage is not terminated in time, a memristor might reach a voltage range which promotes the opposite transition, *i.e.*, reverse switching. The maximal length of the switching stage is determined according to the

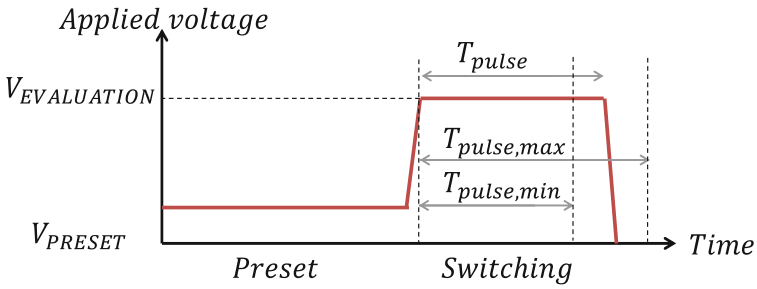


Fig. 7. Applied voltage duration in the switching stage. T_{pulse} satisfies (3) to reach the desired resistance and also meets (4) to avoid reverse switching.

transient analysis of voltages in the circuit, and might be different for SET and RESET operations. For this purpose it is possible to define $T_{SET}(T_{RESET})$ as the approximate period of time in which the conditions for a SET (RESET) operation are met. It is important to understand that while τ_{set} and τ_{reset} are properties of the memristor, T_{SET} and T_{RESET} are determined by the selection of the different circuit parameters, namely $V_{PRESET}, V_{EVALUATION}, R_{REF}, C_{REF}, C_S$, and T_{PRESET}. Hence, the maximum condition on the length of the switching step is

$$T_{pulse} < min\{T_{SET}, T_{RESET}\} = T_{pulse,max}. \tag{4}$$

To comply with both minimum and maximum conditions, both (3) and (4) must apply, as illustrated in Fig. 7. The parameters $V_{PRESET}, V_{EVALUATION}, R_{ref}$, and the switching capacitors can be chosen to support (3) and (4). Different circuit parameters, however, may lead to a reduction in performance. For example, larger capacitors ease the maximum condition, but slow the preset stage and increase power consumption.

3.5 Evaluation and Comparison

We evaluate the proposed circuits in terms of speed, power, and area, and compare them to previously proposed memristive logic families that are suitable for bipolar memristors. Evaluation is conducted using the model mentioned in Sect. 1 based on [22]. All simulations are conducted in Cadence Virtuoso environment, and using device parameter values of R_{ON} = 10 kΩ, R_{OFF} = 1 MΩ, V_{SET} = 2.5 V and V_{RESET} = 1.5 V. In terms of speed, the need for a long preset stage is a disadvantage of the proposed mechanism. To accelerate the preset stage, higher voltages can be used in the cost of higher power consumption. Our simulations show that for a memristor with switching time τ, the delay time of the presented basic logic gates (OR\NOT) is approximately $10 \cdot \tau$.

The basic cell that would be incorporated into a crossbar array consists of a memristor and a capacitor. Suspension capacitors increase the area of the memory cell; the exact area of the capacitor depends on the switching time of the memristor. For example, memristors with switching time of 1 ns require suspension capacitors with

Table 1. Latency and area of different functions using OR, NOT and COPY

Operation	# *OR*	# *NOT*	# *COPY*	# *Backup*	*Latency*
NOT	0	1	0	0	1
OR	1	0	0	0	1
NAND	1	2	0	0	3
NOR	1	1	0	0	2
AND	1	3	0	0	4
XOR	3	4	2	2	9
ADD 1 bit	4	7	2	2	13
ADD N bit	$11N-7$	$14N-7$	$9N-7$	5	$34N-21$

capacitance of approximately 0.8 pF. The usage of suspension capacitors clearly impacts power consumption. Furthermore, the use of several computing phases (preset-switching) requires a clock that contributes to the power consumption and needs to be considered as well.

Some bipolar logic techniques for computation within memory are IMPLY [18] and MAGIC [19]. IMPLY and MAGIC are stateful logic techniques, similar in nature to the proposed technique. In both techniques, logical state is represented by resistance and the computation consists of multi-stage voltage application. Similarly to our proposed unipolar technique, in IMPLY the input data is overwritten with the output result. For devices with switching time of τ, the switching times of IMPLY and MAGIC are 3.15 τ and 1.3 τ respectively. To compare the fundamentals of the performance and area of the different techniques, we have evaluated a test case of an N-bit adder. Recent unipolar and bipolar memristor technology exhibit switching times in the order of 1 ns–10 ns and device area of $4F^2$ [30], making IMPLY and LOGIC comparable to each other and to the proposed logic.

Assume the operation is incorporated in a crossbar that is optimized for area, *e.g.*, only a single operation can be performed at a clock cycle and backup devices can be discarded after usage. The latency and number of backup memristors needed for different logical operations are listed in Table 1. A single bit addition can be performed in 13 cycles. An N bit addition can be performed in 34 $N-21$ cycles. A comparison of this result with existing bipolar logic families is presented in Table 2. Note that due to the requirement of a long preset stage, logic execution for the proposed logic is slower. Given the capacitance and memristor resistance used in simulations, the preset stage is in the order of 100 ns. Thus, the operating frequency of the proposed method is probably lower than the bipolar methods, possibly reducing performance.

Table 2. Latency and area of *N*-bit adder with different memristor-based logic methods, optimized for minimum area

Method of Execution	Latency (# Cycles)	Latency (τ)	Area (# Memristors)
IMPLY (Serial) [10]	29N	91N· τ	2
MAGIC [17]	15N	19N· τ	5
Unipolar (This work)	34N-21	(340N-210)τ	5

4 Methodology for Stateful Memristive Logic Design

One of the most important things when designing novel stateful memristive logic gates is proper selection of the circuit parameters, *i.e.* voltages, currents, resistances, etc. The space of possibilities for choosing these is usually too large to explore, forcing the designer to rely on heuristics. Recently, we have proposed a set of steps to form a structured methodology for the design of stateful memristor-based logic gates [31]. This methodology improves efficiency when inventing new stateful memristive logic gates, and allows a systematic choice of circuit parameters. The design process consists of seven steps, as detailed next. The methodology treats voltage across a memristor as the value that determines its dynamic behavior (switching). While this methodology assume voltage-controlled memristors [32], the same methodology can be adapted with small adjustments for current-controlled memristors. The steps of the design methodology are:

1. *Definition of gate topology* – Decide what are the elements being used (memristors, resistors, capacitors, *etc.*), and how are they connected to each other and to the ports of the logic gate (*e.g.*, connecting the gate to external voltage/current sources).
2. *Definition of gate inputs/outputs* – Decide which memristor values are used as input variables and which as output. All the inputs must have their updated values prior to execution. The output values should be written to the output memristor before execution finishes. An output may run over an input value if needed, as in the OR gate in the previous section and in [18]. When several options exist, this step may be postponed until after step 6 to make a decision relying on a better understanding of the circuit dynamics.
3. *Naming of relevant circuit parameters* that may change their value during execution (*e.g.*, voltage, current, memristance).
4. *Developing an expression for the momentary voltage/current* on each of the memristors in the circuit.
5. *Constructing a truth table of initial voltages* - For each combination of input values, determine what are the voltages across each circuit element at time $t = 0$ (*i.e.*, before any change is observed).
6. *Exploring constraints for choosing the operating voltage/current and the initialization of output memristors* (if they exist). For example, when using bipolar

memristors, the initial state of the output memristor and the applied voltage must be carefully chosen to allow a change of the state.
7. *Examining the unconstrained values and understanding the circuit dynamics* - To allow the proper ranges for each unconstrained value that may produce different behaviors. Once the behavior of the memristors for all parameter ranges is known, select the options that yield the desired logic functionality.

Clauses 1 through 4 are basic groundwork for the gate analysis. Clauses 5 and 6 put restrictions on the chosen parameters so they do not infringe on constraints set by the circuit topology and device properties. Clause 7 requires the most in-depth analysis and should result in parameter selection leading to a new logic gate with useful properties. We demonstrate this design methodology in the next section for unipolar memristors.

5 Design Procedure for a Novel Unipolar Memristor Based Logic Gate

The methodology presented in the previous section is demonstrated for developing another logic gate using unipolar memristors. The steps followed in the development of the gate are presented next.

1. The gate comprises of two unipolar memristors connected in series. The structure, shown in Fig. 8, is compatible for use within a crossbar array.
2. The inputs of the gate are represented by the resistances of the two memristors before the logic function is executed. The output is not selected at this point and will be dealt after step 6. Note that either memristor can be selected as an output after execution since the circuit is symmetrical, and that the lack of a dedicated output memristor makes the gate undoubtedly destructive to at least one of the inputs.
3. The memristors are named *M1* and *M2*, and their resistance, voltage drops and applied voltages to connected terminals are respectively denoted R_1, V_{M1}, $V1$ and R_2, V_{M2}, $V2$. These notations are shown in Fig. 8.
4. The expressions of the momentary voltages as functions of the applied voltages to the gate terminals are given by

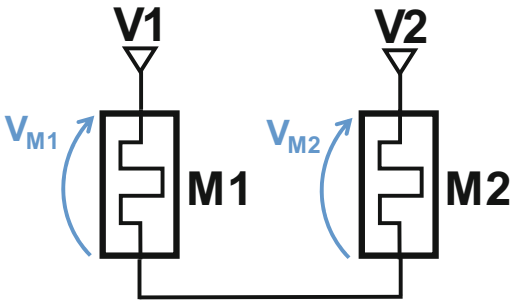


Fig. 8. Gate topology of the analyzed two unipolar logic gate.

$$V_{M1} = (V1 - V2) \cdot \frac{R_1}{R_1 + R_2}, \tag{5a}$$

$$V_{M2} = (V2 - V1) \cdot \frac{R_2}{R_1 + R_2}. \tag{5b}$$

To simplify (5a) and (5b), *V1* is set as ground and *V2* is named V_{OP}. Thus, the simplified expressions are

$$V_{M1} = -V_{OP} \cdot \frac{R_1}{R_1 + R_2}, \tag{6a}$$

$$V_{M2} = V_{OP} \cdot \frac{R_2}{R_1 + R_2}. \tag{6b}$$

5. A truth table for the applied voltage on each device prior to logic execution is presented in Table 3.
6. The chosen topology involves only a single parameter (V_{OP}), whose value will be set in the next clause. Due to the fact that the memristors are unipolar and connected in a symmetric manner, there are no constraints on the polarity of the voltage.
7. Examining the I-V curve shown in Fig. 1, we see that $0 < |Vreset| < |Vset|$. The initial truth table demonstrates that any single memristor within the gate has either 0, $V_{OP}/2$, or V_{OP} applied across it. Considering all of the above, three meaningful options for selecting the value of V_{OP} are present:

(a) 0V → No change, $2V_{OP}/2$ → Reset, V_{OP} → Set.
(b) 0V → No change, $V_{OP}/2$ → No change, V_{OP} → Reset.
(c) 0V → No change, $V_{OP}/2$ → No change, V_{OP} → Set.

Option (b) does not lead to switching of any of the memristors. As is apparent in Table 4, both remaining options lead to an identical state in both memristors at the end of the computation. Hence, we are free to choose the output of the gate to be either of the memristors, affirming the conclusion of step 2.

Table 3. Truth table for memristor voltages before any change in device state

Input Values		Input Resistance		Applied Voltage	
$M1_{init}$	$M2_{init}$	$R_{1,init}$	$R_{2,init}$	V_{M1}	V_{M2}
0	0	R_{OFF}	R_{OFF}	$V_{OP}/2$	$V_{OP}/2$
0	1	R_{OFF}	R_{ON}	$\sim V_{OP}$	~ 0
1	0	R_{ON}	R_{OFF}	~ 0	$\sim V_{OP}$
1	1	R_{ON}	R_{ON}	$V_{OP}/2$	$V_{OP}/2$

Table 4. Analysis of gate operation for the two relevant operating voltage selections

Input Resistance		Initial Applied Voltage		Option (a) Final state		Option (c) Final state	
$R_{1,init}$	$R_{2,init}$	V_{M1}	V_{M2}	$R_{1,final}$	$R_{2,final}$	$R_{1,final}$	$R_{2,final}$
R_{OFF}	R_{OFF}	$V_{OP}/2$	$V_{OP}/2$	R_{OFF} (0)		R_{OFF} (0)	
R_{OFF}	R_{ON}	$\sim V_{OP}$	~ 0	R_{ON} (1)		R_{ON} (1)	
R_{ON}	R_{OFF}	~ 0	$\sim V_{OP}$	R_{ON} (1)		R_{ON} (1)	
R_{ON}	R_{ON}	$V_{OP}/2$	$V_{OP}/2$	R_{OFF} (0)		R_{ON} (1)	

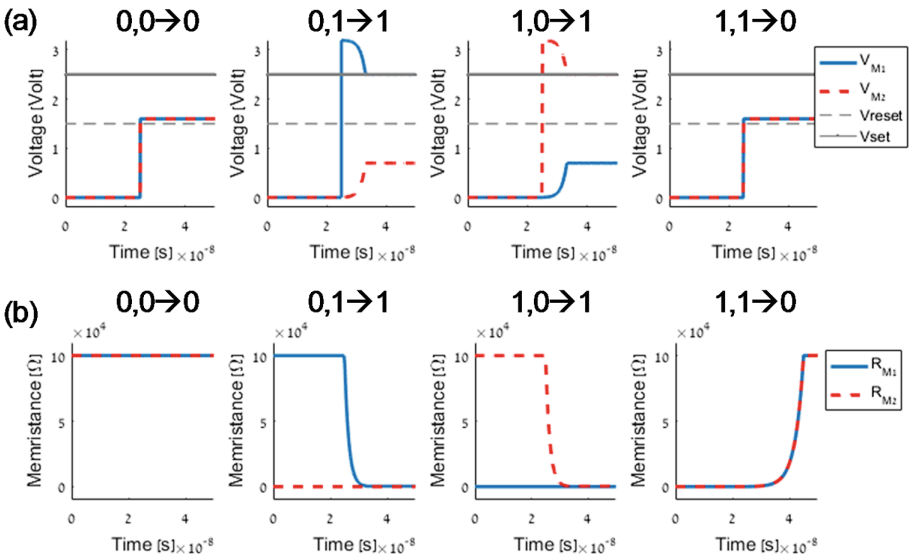


Fig. 9. (a) Applied voltage and (b) memristance for an XOR gate. The memristor is characterized by R_{ON} = 100 Ω and R_{OFF} = 100 kΩ. The circuit parameters are V_{RESET} = 1.5 V, V_{SET} = 2.5 V, V_{OP} = 3.2 V.

Option (a) results in an XOR gate. However, this gate is unstable since the output values for an XOR function are, theoretically, initial values for another round of computation, resulting in a constant output equal to R_{OFF}. Using the model discussed in Sect. 1, our results show convergence of the output at a resistance of approximately R_{ON}. The exact value depends on R_{OFF}/R_{ON} and *Vset*/*Vreset*, as shown in Fig. 9. Thus, executing an XOR operation is possible if we allow partial switching, although the noise margin of the gate is relatively low (asymptotically reaching a full switching with a proper selection of parameters, improving the noise margin).

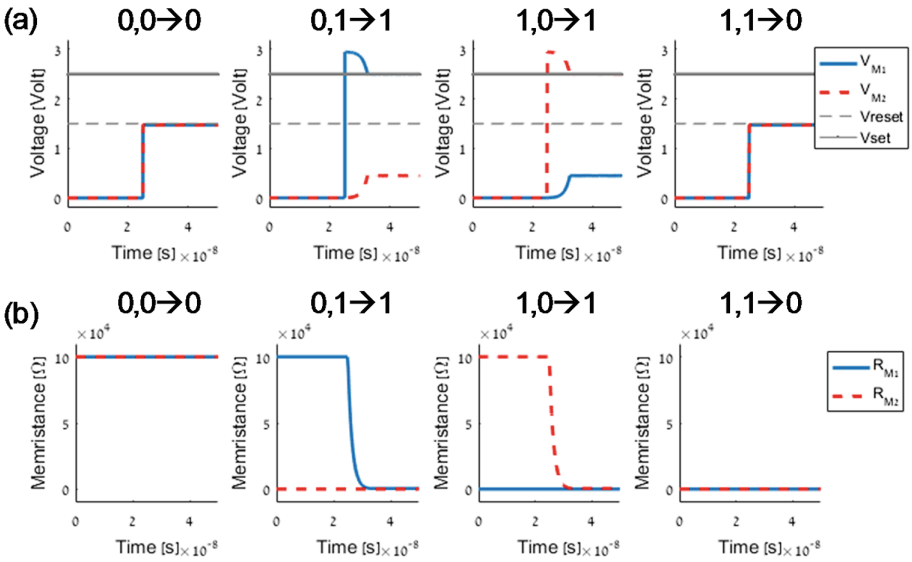


Fig. 10. (a) Applied voltage and (b) memristance of an OR gate. Memristor and circuit parameters are identical to an XOR gate, except V_{OP} = 2.95 V

Option (c) results in an OR gate. This gate is stable and, with a wide range of parameter values, correctly converges to the desired output with no noise margin issues. Simulation results of this gate are shown in Fig. 10. For proper operation of this OR gate the threshold voltages of the memristors are required to uphold

$$1 < \left|\frac{V_{SET}}{V_{RST}}\right| < 2. \tag{7}$$

Some physical unipolar devices exhibit (7) [33, 34], while other devices exhibit a higher ratio ($2V_{RESET} < V_{SET}$) [35–37], enabling only XOR operations, or do not fulfill any of these conditions (*i.e.*, uphold $V_{SET} < V_{RESET}$) [34] and therefore are not suitable for use with the proposed topology.

Contrary to the gate described in Sect. 3, these gates do not contain any capacitors, nor do they rely on retaining previous voltage divider values. For these reasons, there are no timing constraints on gate operation, apart from the obvious necessity to apply V_{OP} for a time sufficient for achieving full swing in the device states (τ). This time depends on properties of the used device and may vary substantially between different types of devices.

The gate described in this section outperforms the gates from Sect. 3 in several aspects. First, the topology does not include the use of capacitors or resistors, which is area efficient, allows implementing gates within a pure memristive crossbar, and eliminates the need to use two different input voltages to perform the logic function. Second, the topology allows, with a proper selection of devices and parameters, to use the same gate for two different logic functions by changing only the operating voltage.

6 Conclusions

Combining data storage and processing is appealing since it can solve some of the critical issues in modern computing, such as limited memory bandwidth and power consumption. Both unipolar and bipolar memristors enable the execution of logic operations within memory using different methods. Since it is still unclear whether unipolar or bipolar mechanisms will become dominant for data storage, both phenomena are of interest. In this chapter, we focus on unipolar mechanism and propose logic techniques for these devices using NOT, XOR and two types of OR gates. The proposed techniques can be naturally integrated within memristive crossbar memory. The proposed technique can fit different unipolar technologies such as Phase Change Memory, 3D-Xpoint, RRAM, and Thermochemical Resistive Memory.

We present how a design methodology helps in the invention of new logic gates that can be executed within memristive memories to form memristive memory processing units (mMPU). The methodology is demonstrated by designing XOR and OR gates. This procedure is formed from a series of simple steps, and meant to facilitate a successful choice of circuit parameters and an overall efficient design process.

References

1. Chua, L.: If it's pinched it's a memristor. Semicond. Sci. Technol. **29**(10), 104001 (2014)
2. Chua, L.: Resistance switching memories are memristors. Appl. Phys. A **102**(4), 765–783 (2011)
3. Wong, H.S.P., et al.: Metal–oxide RRAM. Proc. IEEE **100**(6), 1951–1970 (2012)
4. Diao, Z., et al.: Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory. J. Phys. Condens. Matter **19**(16), 165209 (2007)
5. Wong, H.S.P., et al.: Phase change memory. Proc. IEEE **98**(12), 2201–2227 (2010)
6. Chua, L.: Memristor-the missing circuit element. IEEE Trans. Circ. Theory **18**(5), 507–519 (1971)
7. Strukov, D.B., Snider, G.S., Stewart, D.R., Williams, R.S.: The missing memristor found. Nature **453**(7191), 80–83 (2008)
8. Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., Lu, W.: Nanoscale memristor device as synapse in neuromorphic systems. Nano Lett. **10**(4), 1297–1301 (2010)
9. Rose, G.S., McDonald, N., Yan, L.-K., Wysocki, B.: A write-time based memristive PUF for hardware security applications. In: Proceedings of the International Conference on Computer-Aided Design, pp. 830–833 (2013)
10. Rajendran, J., Rose, G.S., Karri, R., Potkonjak, M.: Nano-PPUF: a memristor-based security primitive. In: Proceedings of the IEEE Computer Society Annual Symposium on VLSI, pp. 84–87 (2012)
11. Borghetti, J., Snider, G.S., Kuekes, P.J., Yang, J.J., Stewart, D.R., Williams, R.S.: 'Memristive' switches enable 'stateful' logic operations via material implication. Nature **464** (7290), 873–876 (2010)
12. McKee, S.A.: Reflections on the memory wall. In: Proceedings of the First Conference on Computing Frontiers on Computing frontiers, p. 162 (2004)
13. Ben Hur, R., Kvatinsky, S.: Memory processing unit for in-memory processing. In: Proceedings of the International Symposium on Nanoscale Architectures, p. 208 (2016)

14. Ben Hur, R., Kvatinsky, S.: Memristive memory processing unit (MPU) controller for in-memory processing. In: Proceedings of the IEEE International Conference on the Science of Electrical Engineering (ICSEE) (2016)
15. Kvatinsky, S., Wald, N., Satat, G., Kolodny, A., Weiser, U.C., Friedman, E.G.: MRL — memristor ratioed logic. In: Proceedings of the International Workshop on Cellular Nanoscale Networks and Their Applications, pp. 1–6 (2012)
16. Levy, Y., et al.: Logic operations in memory using a memristive Akers array. Microelectron. J. **45**, 1429–1437 (2014)
17. Rosezin, R., Linn, E., Kugeler, C., Bruchhaus, R., Waser, R.: Crossbar logic using bipolar and complementary resistive switches. IEEE Electron Device Lett. **32**(6), 710–712 (2011)
18. Kvatinsky, S., Satat, G., Wald, N., Friedman, E.G., Kolodny, A., Weiser, U.C.: Memristor-based material implication (IMPLY) logic: design principles and methodologies. IEEE Trans. Very Large Scale Integr. VLSI Syst. **22**(10), 2054–2066 (2014)
19. Kvatinsky, S., et al.: MAGIC - memristor-aided logic. IEEE Trans. Circuits Syst. II Express Briefs **61**(11), 895–899 (2014)
20. Ben Hur, R., Talati, N., Kvatinsky, S.: Algorithmic considerations in memristive memory processing units (MPU). In: Proceedings of the International Workshop on Cellular Nanoscale Networks and their Applications (2016)
21. Talati, N., Gupta, S., Mane, P., Kvatinsky, S.: Logic design within memristive memories using memristor-aided logic (MAGIC). IEEE Trans. Nanotechnol. **15**(4), 635–650 (2016)
22. Su Kim, Y., Min, K.-S.: Behavioral Current-voltage model with intermediate states for unipolar resistive memories. J. Semiconductor Technol. Sci. **13**(6), 539–545 (2013)
23. Ielmini, D., Bruchhaus, R., Waser, R.: Thermochemical resistive switching: materials, mechanisms, and Scaling projections. Phase Transit. **84**(7), 570–602 (2011)
24. Long, S., Cagli, C., Ielmini, D., Liu, M., Sune, J.: Cell-based models for the switching statistics of RRAM. In: Proceedings of the Annual Non-Volatile Memory Technology Symposium, pp. 1–5 (2011)
25. Tran, X.A., et al.: High performance unipolar AlO y/HfO x/Ni based RRAM compatible with Si diodes for 3D application. In: Symposium on VLSI Technology - Digest of Technical Papers, pp. 44–45 (2011)
26. Pirovano, A., Lacaita, A.L., Benvenuti, A., Pellizzer, F., Bez, R.: Electronic switching in phase-change memories. IEEE Trans. Electron. Devices **51**(3), 452–459 (2004)
27. Redaelli, A., Pirovano, A., Pellizzer, F., Lacaita, A.L., Ielmini, D., Bez, R.: Electronic switching effect and phase-change transition in chalcogenide materials. IEEE Electron Device Lett. **25**(10), 684–686 (2004)
28. Amrani, E., Drori, A., Kvatinsky, S.: Logic design with unipolar memristors. In: Proceedings of the International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1–5 (2016)
29. Waser, R., Dittmann, R., Staikov, G., Szot, K.: Redox-based resistive switching memories - nanoionic mechanisms, prospects, and challenges. Adv. Mater. **21**(25–26), 2632–2663 (2009)
30. Yang, J.J., Strukov, D.B., Stewart, D.R.: Memristive devices for computing. Nat. Nanotechnol. **8**(1), 13–24 (2012)
31. Wald, N., Kvatinsky, S.: Design methodology for stateful memristive logic gates. In: Proceedings of the ICSEE International Conference on the Science of Electrical Engineering (ICSEE) (2016)
32. Kvatinsky, S., Ramadan, M., Friedman, E.G., Kolodny, A.: VTEAM: a general model for voltage-controlled memristors. IEEE Trans. Circ. Syst. II Express Briefs **62**(8), 786–790 (2015)
33. Guan, W., Liu, M., Long, S., Liu, Q., Wang, W.: On the resistive switching mechanisms of Cu/ZrO2:Cu/Pt. Appl. Phys. Lett. **93**(22), 223506 (2008)

34. Huang, Y., Luo, Y., Shen, Z., Yuan, G., Zeng, H.: Unipolar resistive switching of ZnO-single-wire memristors. Nanoscale Res. Lett. **9**(1), 381 (2014)
35. Park, W.I., et al.: Self-assembly-induced formation of high-density silicon oxide memristor nanostructures on graphene and metal electrodes. Nano Lett. **12**(3), 1235–1240 (2012)
36. Huang, H.H., Shih, W.C., Lai, C.H.: Nonpolar resistive switching in the Pt/MgO/Pt nonvolatile memory device. Appl. Phys. Lett. **96**(19), 193505 (2010)
37. Guan, W., Long, S., Liu, Q., Liu, M., Wang, W.: Nonpolar nonvolatile resistive switching in Cu doped ZrO2. IEEE Electron Device Lett. **29**(5), 434–437 (2008)

Robust Hybrid TFET-MOSFET Circuits in Presence of Process Variations and Soft Errors

Maedeh Hemmat[1], Mehdi Kamal[1(✉)], Ali Afzali-Kusha[1], and Massoud Pedram[2]

[1] School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
{m.hemmat,mehdikamal,afzali}@ut.ac.ir
[2] Department of Electrical Engineering, University of Southern California, Los Angeles, USA
pedram@usc.edu

Abstract. In this work, to improve the timing yield of Tunnel Field Effect Transistor (TFET) circuits in the presence of process variations as well as their soft-error resiliency, we propose replacing some of TFET-based gates by MOSFET-based ones. The effectiveness of the proposed TFET-MOSFET hybrid implementation of the circuits are investigated by first studying the impacts of the process variation on the performances (*I-V* characteristics) of both homojunction InAs TFETs and MOSFETs. Next, to analyze the soft error rate of the circuits, the particle hit-induced transient current profiles of these devices are extracted. Based on these studies, a hybrid TFET-MOSFET circuit design approach which improves the reliability and soft-error resiliency compared to those of pure TFET-based circuits is suggested. Finally, the efficacy of the design approach is investigated by applying it to some circuits of ISCAS'89 benchmark package.

Keywords: Tunnel FET · Reliability issues · Process variation · Low power design · Hybrid TFET-MOSFET designs · Soft error

1 Introduction

Todays, almost all of the digital circuits are based on the MOSFET transistors. However, owing to the increase in the usage of portable devices, reducing the power consumption of digital circuits has become a critical target for designers. A fundamental method for reducing the power consumption is voltage scaling which leads to a quadratic reduction in dynamic power. On the other hand, to maintain the performance of design, the threshold voltage of transistors is decreased leading to increase in the leakage current of the circuits. Also, short channel effects cause considerable increases in the leakage current of the nano-scaled MOSFET transistor. One may reduce the leakage power by reducing the subthreshold swing which may not be decreased below 60 mV/decade for conventional MOSFET device structures [1]. Hence, for low-leakage power application one may replace conventional MOSFET structures with devices having smaller swings.

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 41–59, 2017.
DOI: 10.1007/978-3-319-67104-8_3

One of these alternative devices is Tunnel Field Effect Transistor (TFET) which is known for its steep sub-thermal subthreshold swing [1]. TFETs are P-i-N gated diodes, operating under reverse bias condition, with a gate over the intrinsic region. They are good candidates to operate at low supply voltages ($V_{dd} < 0.3$ V) while having ultra-low leakage power. The current generation mechanism in TFETs is band to band tunneling of carriers across a reversed biased PN junction [1]. Nowadays, III-V TFETs with small and direct band gap have acceptable ON-current compared to those of MOSFETs [2]. Therefore, unlike MOSFETs, reducing the (leakage) power consumption is possible for TFETs without any degradation in the performance. In [3], a mixed TFET-MOSFET 8T SRAM was proposed providing a significant improvement in the performance as well as the minimum operating voltage.

While enjoying TFET advantages in terms of leakage and power consumption, their reliability issues should also be considered and investigated. The reliability of these circuits may significantly be affected by the process variation which causes uncertainties in the (design) parameters of fabricated devices. Several studies have compared the impacts of the process variation on the performances of TFET and MOSFET devices [4–6]. For instance, in [4], by comparing the changes of the ON-current for a Hetero-junction TFET and Silicon MOSFET, it was concluded that TFETs were more sensitive to the process variation. In [7], the influence of the process variation on the electrical parameters of III-V TFET were studied and their statistical distributions were extracted and compared to those of MOSFET devices. To increase the reliability of TFET-based circuits, in [8], a heuristic algorithm for generating hybrid TFET-MOSFET based circuits was introduced. In this algorithm, some of the TFET-based gates whose variation degraded the performance (speed) of the circuit were replaced by their MOSFET-based gates.

Radiation hardness is another important issue in designing reliable circuits. Radiation-induced single-event upset (SEU) leads to the generation of soft errors and radiation unreliability issues [9]. Furthermore, increasing the number of processing elements (as is the case for data centers) on one side and the technology scaling on the other side, make satisfying soft error immunity requirements increasingly challenging. In every generation, technology scaling roughly leads to a 30% decrease in node capacitance, 30% decrease in the supply voltage, and doubling clock frequencies. As a byproduct of these changes, the soft error rate (SER) per logic state bit increases by 8% [10]. Also, from a totally different perspective, compared to silicon, low band gap materials such as InAs, generally have low ionization energy making them more susceptible to neutron radiation which generates soft errors [11].

The investigation of the soft error phenomenon in TFETs in [11] showed that TFETs had a different soft error generation behavior than that of MOSFETs. In [12], we investigated and compared the behavior of TFETs and MOSFETs in terms of generation and propagation of the soft error. The results indicated that TFETs possessed better performance in terms of the generation of soft errors while MOSFETs were able to mask the generated current better. Inspired by this difference in behaviors, then we proposed a hybridization algorithm to generate soft error resilient MOSFET-TFET based circuits.

In this work, we propose a design approach to increase the reliability of TFET-based circuits considering the process variation and soft error phenomena. In the first step, the

characteristics of III-V TFET and MOSFET devices in the presence of the process variation and particle hits are compared. In the second step, for ultra-low (static and dynamic) power applications, we suggest a hybrid TFET-MOSFET circuit design approach which minimizes the impacts of the process variation and improves the soft-error resiliency compared to the cases where only of these device types are used. To optimize the efficacy of the approach for each application, the designer may set one of these phenomena as the one with the higher priority in the design process. Finally, the efficacy of the proposed approach is investigated by applying the method to some circuits from the ISCAS'89 benchmark package. It should be noted that the overall fabrication processes for TFETs are compatible with CMOS process [3, 13], and hence, it is possible to have hybrid circuits as stated in [3, 13, 14]. Of course, the fabrication of TFET circuits costs more and is more complicated [14]. Finally, it should be mentioned that the proposed hybridization approach is efficient for the range of supply voltages where the performances of both TFET and MOSFET devices are similar.

The rest of paper is organized as follows. Section 2 compares the characteristics of the InAs TFET device and with those of the InAs MOSFET device. The hybridization approach realized by a proposed heuristic algorithm is discussed in Sect. 3. The efficacy of the proposed algorithm in increasing the reliability in the presence of soft error and process variation is evaluated in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 TFET Device and Its Characteristics

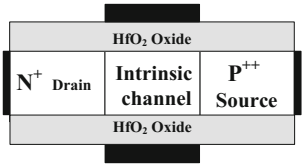
2.1 TFET Device Model

In this work, for obtaining an ultra-thin 22 nm double-gate InAs homojunction TFET, the TCAD simulation was used. The nominal parameters of the device model which were adapted from [8, 12, 15] are summarized in Table 1. Also, the structure and band diagram of the device are shown in Fig. 1. All the device level simulations have been performed using Silvaco ATLAS version 5.18.3.R. The activated physical models included non-local band to band tunneling, band gap narrowing, Shockley-Read-Hall model, and mobility model. The output characteristic of the Homojunction TFET ($I_d - V_{ds}$) is shown in Fig. 2. It should be noted that, unlike MOSFETs, TFETs are uni-directional devices because of their structural asymmetry between the source and drain [1]. The ON-current of 123μA at $V_{ds} = V_{gs} = 0.5$ V, leakage current of 32 pA, and $I_{on}/I_{off} = 3.8 \times 10^6$ were obtained for this TFET device.

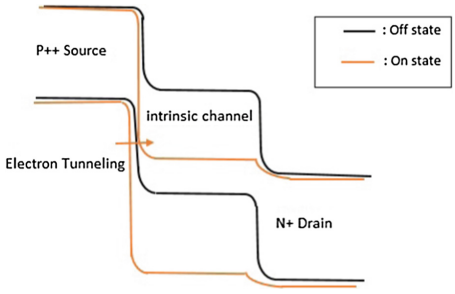
For the circuit-level simulations, Verilog-A look-up tables, which used the small signal model of Fig. 3(a), was used. Furthermore, the validity of the model was verified by the transient output characteristics of InAs TFET inverter (see Fig. 3(b)). The considered structure of InAs MOSFET was also an ultra-thin 22 nm double gate. To have a meaningful comparison (and better hybridization during the design), the nominal parameters of the MOSFET were selected similar to those of TFETs except for the high drain doping level (the same as that of the source) which was necessary for this structure.

Table 1. Nominal parameters of TFET device considered in this work.

Parameter	Nominal value
Relative Gate Dielectric Permittivity	21
Body Thickness	5 nm
Gate Oxide Thickness	2.5 nm
Source Doping	$4 \times 10^{19}/cm^3$
Drain Doping	$4 \times 10^{17}/cm^3$
Gate Work Function	4.8 eV
Channel length	22 nm
Channel width	22 nm



(a)



(b)

Fig. 1. TFET (a) device structure and (b) energy band diagram.

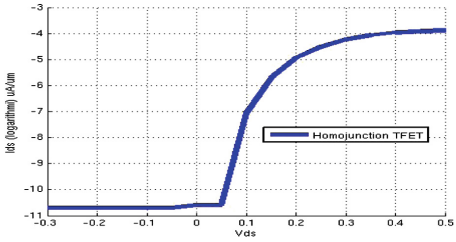


Fig. 2. Output characteristics of InAs TFET.

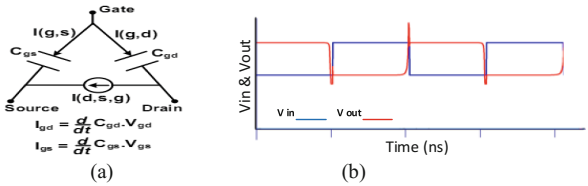


Fig. 3. (a) Verilog-A small signal model and (b) transient response of a TFET inverter based on the model.

2.2 Comparison of TFET and MOSFET Operation

In this subsection, a comparative study on the leakage current, drive current, dynamic power consumption, sensitivity to process variation and soft error generation and propagation of both III-V MOSFET and TFET devices based on our prior works of [8, 12] are performed.

Current and capacitance comparison

The ON-current of the III-V TFET and MOSFET device under two operating voltage levels are compared in Fig. 4 which indicate that the TFETs has lower leakage current and higher ON-current at low voltages. On the other hand, as is demonstrated in Fig. 4 (b), in spite of the low leakage current of the TFET at high supply voltages, the MOSFET device has a higher ON-current which would lead to a higher performance at high supply voltages for the circuits realized by this device. Also, Fig. 5 compares TFET and MOSFET total capacitances. The comparison indicates that the gate capacitance (C_g) of the TFET is smaller than that of the MOSFET where the difference enlarges as the gate voltage is increased. The lower capacitance, which is due to the lighter drain doping of the TFETs [16], leads to a smaller gate capacitance (C_{gg}) (and hence, lower dynamic power consumption) for the TFET device.

In MOSFETs, the dominant capacitance is the gate-source capacitance (C_{gs}) while in TFETs, the gate-drain capacitance (C_{gd}) is the dominant one [16]. This would imply a larger ON-state Miller capacitance for TFETs [3], and consequently, larger induced voltage spike during the switching giving rise to increased total dynamic power consumption. Considering the capacitance and ON-current characteristics of both the

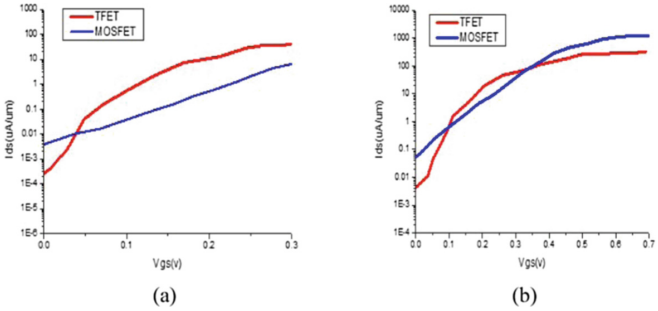


Fig. 4. Comparison of leakage current and ON-current at (a) V_{dd} = 0.3 V and (b) V_{dd} = 0.7 V.

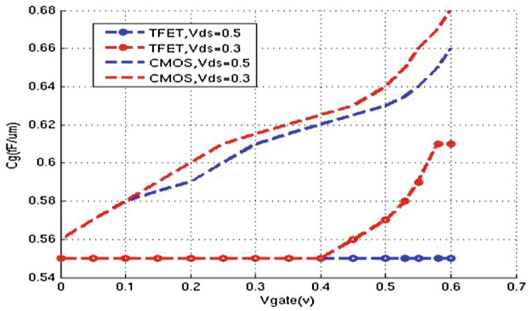


Fig. 5. Comparison of capacitance $C_g - V$ of the TFETs and MOSFET devices at different supply voltages.

TFET and the MOSFET, the supply voltage range in which the overall performance of TFET and MOSFET is close to each other, is about 0.45 V < V_{dd} < 0.55 V.

Process variation

As discussed earlier, compared to MOSFETs, TFETs are more prone to the process variation. In [7], we investigated the impact of the process variation on the physical parameters of InAs TFET in the presence of the process variation. The investigation was performed by utilizing the Monte-Carlo simulations where the distributions of threshold voltage and ON-current were extracted for 1000 samples. The results presented in the work showed a threshold voltage variation of about 75 mV (45 mV) for the 22 nm InAs TFET (MOSFET) device. This implies more sensitivity to the process variation for the TFET device. It is attributed to the fact that the ON-current and subsequently the threshold voltage of the TFET device have exponential dependences on the electric field making them more susceptible to the sources of variations.

Table 2 summarizes the statistical parameters considered for the distributions of variation sources while Table 3 reports the means and standard deviations for the electrical parameters of the InAs TFET [7].

Soft error generation and propagation

Radiation induced single-event upset (the soft error) is a key challenge due to large number of computation nodes in circuits. Using low bandgap materials, scaling the supply voltage, and the reduction of capacitance of the internal nodes have made designing soft-error resilient circuits more challenging. The energetic particles, such as cosmic ray neutrons and alpha particles, are the sources of soft errors [11]. The energetic particles strike the sensitive nodes and travel through the bulk of the transistor. The creation of the minority carriers during the travel of the particles and the collection of them by the source/drain diffusion, can change the voltage value of the victim node. In other words, soft error occurs when the collected charge at a specific node is greater than the critical charge (Q_{crit}) of the node [17].

The particle hit and the change in the value of the node can be modeled by a transient current pulse [18]. In [12], we have investigated the transient current

Table 2. Statistical parameters considered for the distributions of variation sources in the homojunction TFET.

Physical parameter	Mean	Standard deviation
Gate oxide thickness	2.5 nm	0.5 nm
Body thickness	5 nm	1 nm
Gate alignment	0	3 nm
Gate work function	4.8 eV	0.08 eV
Source doping	4e19	0.8e19
Drain doping	4e17	0.8e17
Channel length	22 nm	2 nm

Table 3. Means and standard deviations of the electrical parameters for the TFET.

Parameter	Nominal	Mean	Standard deviation
Threshold voltage	123 mV	137 mV	75 mV
ON-current	123 μA	121 μA	26.9 μA

generation and soft error propagation in III-V TFETs and III-V MOSFETs, using TCAD device models and HSPICE simulations. As discussed in detail in [12], to analyze the behavior of the transistor after the ion strike, the radiation-induced transient current evaluation was performed. The generated charges, due to the particle hit, result in a transient current when the device is in the off state. ($V_{gd} = V_{dd}$, $V_{gs} = 0$). The transient current generation and charge collection in fully depleted channel devices such as MOSFETs, FinFETs, and TFETs are significantly influenced by the bipolar gain effect [11]. Figure 6 compares the generated transient current profile for the InAs TFET and the MOSFET. As the figure demonstrates, the duration and amplitude of the transient current of the TFET are about 80% and 70% smaller than those of the MOSFET. It suggests that TFET devices are more immune to the soft error generation. Here, as an example, Fig. 7 shows the output voltage spike of a MOSFET- and TFET-based FO4 inverter when a particle with a charge density of 50 fC/μm strikes the input of the inverter. The output voltage spike of the MOSFET-based inverter is up to 310 mV while the voltage change for the TFET-based inverter is about 140 mV. Therefore, the generated voltage spikes in the case of the TFET-based gates are smaller compared to those of the MOSFET-based gates.

The generated transient current due to the particle strike induces a voltage pulse, known as a glitch, which may propagate through logic paths in the circuit. The voltage pulse may be electrically masked due to the delay of logic gates. If it is not masked, an error would happen when the generated voltage pulse either is latched by a flip flop or reaches to a primary output node of the circuit. Therefore, to conduct a complete investigation of the soft error rates in TFET- and MOSFET-based circuits, the propagation of glitches (due to the particle hits) in each type of the circuits was also considered and studied in [12]. In this work, it was concluded that TFETs propagate glitches more easily compared to MOSFETs indicating lower electrical masking for TFETs. The observed behavior is attributed to the fact that the smaller overall capacitance of TFETs compared to that of MOSFETs (see Subsect. 2.1). In addition, as

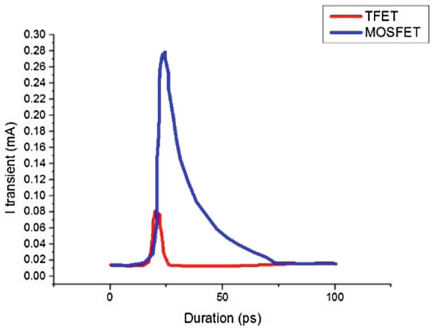


Fig. 6. Profiles of the generated transient current for the InAs TFET and MOSFET.

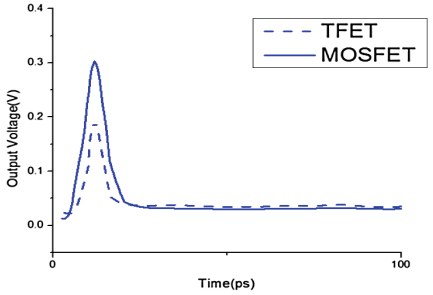


Fig. 7. Output voltage of MOSFET and TFET-based FO4 inverters after particle hit.

mentioned before, at low supply voltage levels ($V_{dd} < 0.3$ V), the TFET has higher ON-current compared to that of the MOSFET. This yields a higher probability for the error propagations (less masking) by the TFET device (the TFET-based gate). Figure 8 shows the voltage spikes at the output nodes of two inverter chains when a particle with a charge density of 50 fC/μm hits the input of the chain. As the figures demonstrate, the amplitude of the voltage spike at the output of the chain in the case of MOSFET-based implementation is smaller than that of the TFET-based implementation showing a better masking for the MOSFET one.

As the conclusion of this part, we showed that the TFET device exhibited a better characteristic in terms of the generation of the transient current due to the particle hit while the MOSFET device revealed a better characteristic for the electrical masking of the transient current due to the particle hit [12].

3 Hybrid TFET-MOSFET Circuits

3.1 Why Hybridization?

As mentioned previously, while the TFET leakage power and (dynamic) energy consumption are smaller than those of the MOSFET, the relation between the speeds (and

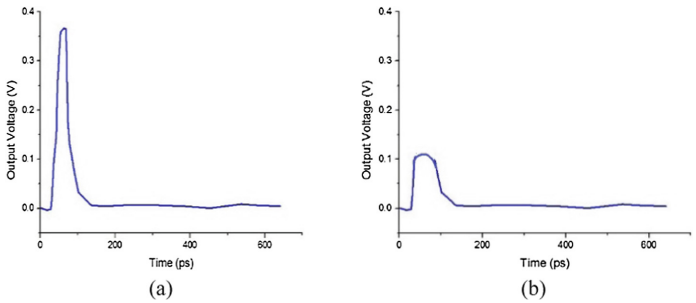


Fig. 8. Output voltage of the chains of three (a) TFET-based and (b) MOSFET-based inverters after the particle hit the input of chain.

delays) of the TFET and MOSFET circuits depend on the operating voltage. Also, the process variation impacts the electrical parameters of the TFET more compared to the MOSFET. Hence, the probability of timing violation in TFET-based circuits is higher. Furthermore, the generated transient current is weaker in TFETs despite the fact that MOSFETs have better performance in masking these errors.

As mentioned before, to take advantage of both TFETs and MOSFETs superior features, robust ultra-low power circuits were proposed in [8, 12]. To increase the reliability of TFET-based circuits in the presence of the process variation, a heuristic algorithm was proposed in [8] which replaced the TFET-based gates in the potential critical paths with their corresponding MOSFET-based gates. In the proposed approach, the hybridization process (gate replacement) started from the initial stages of the circuit. The results showed about 50% increase, on average, in the reliability of hybrid TFET-MOSFET based gates compared to the pure TFET-based gates when applied to ISCAS'85 and ISCAS'89 benchmarks.

Another hybridization algorithm, which was proposed in [12], focused on decreasing the soft error rate of TFET-based circuits, again, by replacing some of the TFET gates by their corresponding MOSFET ones. The TFET-based gates are superior to MOSFET-based ones in terms of the error generation, while the latter mask the error more efficiently. Thus, to have a soft-error resilient design, one should have the TFET-based gates in the generation path of the transient (error) current and the MOSFET-based ones in the propagation path of the transient (error) voltage. Based on these features, in the proposed algorithm of [12], first, the sensitive internal nodes were chosen. The sensitive nodes considered to be the nodes with small capacitance and high probability of generating the voltage spike. Then, for each chosen sensitive node, the first gate of each path starting from the sensitive node, was considered to be implemented by TFETs. This resulted in generating smaller voltage pulses due to particle hits. However, the gates in the second and third levels were considered to be implemented by MOSFETs to electrically mask the generated voltage pulses. Furthermore, a hybrid soft-error resilient flip-flop was proposed in [12]. Applying the proposed algorithms to the circuits from the ISCAS'89 benchmark package as well as utilizing the hybrid flip-flop in these circuits led to, on average, 80% decrease in the soft-error rate compared to those of the pure TFET-based designs.

These prior works showed that the hybridization resulted in improvements in the timing yield as well as the soft error immunity. While each of these hybridization approaches improved the performance of the circuit against the impact of one of these undesired phenomena, in this work, we focus on enhancing the circuit operation in the simultaneous presences of the process variation and soft error. In the next subsection, the proposed hybridization algorithm is described in detail.

3.2 Proposed Heuristic Hybridization Algorithm

The flow of the proposed algorithm may be divided into three major parts. In the first part of the algorithm, which is shown in Fig. 9, the operating supply voltage level, potential critical paths, and sensitive nodes to the soft error are determined. In the first step of this part (❶), in order to avoid the degradation of the speed of the circuits after the hybridization, the supply voltage level which leads to almost the same delays for both the TFET- and MOSFET-based implementations of the input design is determined. The delays of both pure TFET and MOSFET circuits should be smaller than a predefined delay (*i.e.*, D_{const}). This is an iterative process which is performed by using HSPICE simulations under different supply voltage levels. It starts by the parameter V_{start} as the initial operating voltage level and continues by increasing the voltage by V_{step} at the end of each iteration. In each iteration, the delays of both TFET- and MOSFET-based circuits are compared. If the delays have an acceptable delay difference with both delays smaller than D_{const}, this process is terminated. Otherwise, the process is repeated by increasing the supply voltage level. In this work, we considered the delay difference of smaller or equal than 10% of the delay of MOSFET-based circuit as the acceptable delay difference. The reason for considering the 10% delay difference as the acceptable value was to lower the effort for finding the supply voltage for the similar performances for all the implementations.

Based on the ON-current values which were given in Sect. 2, in this work, we consider 0.4 V as V_{start}. Also, our study showed that the selected supply voltage levels for different circuits did not exceed 0.55 V. It is worth mentioning that if D_{const} is too small, the process would not converge to a supply voltage level, and hence, D_{const} should be increased in these cases. Obviously, the lower supply voltage is translated to lower power consumption and, hence, the search process starts from the lowest supply voltage.

After determining the supply voltage level, the set of potential critical paths (*i.e.*, S_{PCP}) of the TFET-based circuit is determined by using static timing analysis (STA) (❷). In this work, the paths which have delays larger than 80% of the longest path, are considered as the potential critical paths. For performing the hybridization, we propose to start from TFET-based implementation of the design while all the flip-flops are implemented by the proposed hybrid approach of [12]. Hence, after determining the supply voltage level and critical paths, the capacitances of the internal nodes are extracted and a collection of sensitive internal nodes are chosen (*i.e.*, S_{SIN}) (❸). As mentioned previously, the sensitive nodes are the nodes with small capacitances where the probability of generating the voltage spike due to the particles hit on these nodes is high. Here, a node whose capacitance is less than or equal to 1.2 times of the smallest internal node capacitance of the circuit is specified as the sensitive node. In the

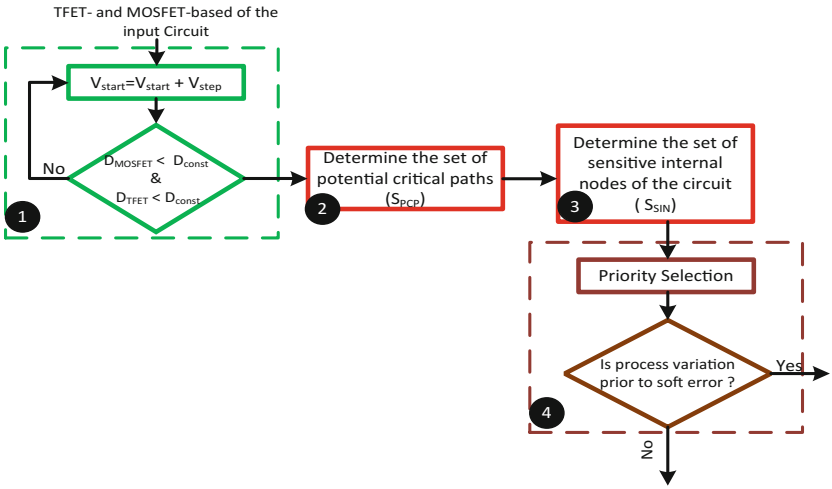


Fig. 9. First part of the proposed algorithm which is common between the two hybridization paths.

proposed flow, the objective is both to increase the timing yield and decrease the soft error rate. The designer, in this step, based on the priority specified by the designer, one the two hybridization paths is selected (❹). If the priority is given to the soft error-resiliency (process variation mitigation), the next steps of the algorithm put more emphasis on reducing the impact of the soft error (process variation) while attempting to decrease the impact of the process variation (soft error) with a lower priority effort. Next, the details of the algorithm in the cases of process variation mitigation and soft error resiliency priorities are explained.

Priority on process variation mitigation

Figure 10 shows the rest of the proposed algorithm for the case where the process variation mitigation is selected as the main priority of the hybridization. For this case, similar to [8], the hybridization process for each path starts from the first gate of the path owing to the fact that the delay variation at the beginning stages of TFET-based designs is larger. By starting the replacement process from the beginning gates, one mitigate the delay variation of the path more by replacing smaller numbers of gates [8].

In the first step of the proposed flow (❶), for each potential critical path, the delay distribution of the MOSFET-based implementation is extracted (*i.e.,* $(\sigma/\mu)_{MOSFET}$). The delay distribution of the MOSFET-based implementation is used as a reference point for the next phase of the algorithm. The process of extracting the delay distribution may be performed by exploiting either statistical static timing analysis (SSTA) or Monte-Carlo simulation. The former approach is fast and inaccurate while the latter one is accurate and slow. In this work, we have used Monte-Carlo simulations to extract the delay distributions of the potential critical paths.

For each potential critical path, after determining the delay distribution of the MOSFET-based implementation, the hybridization process for this path in the TFET-based implantation circuit is performed (❷). This process is an iterative process,

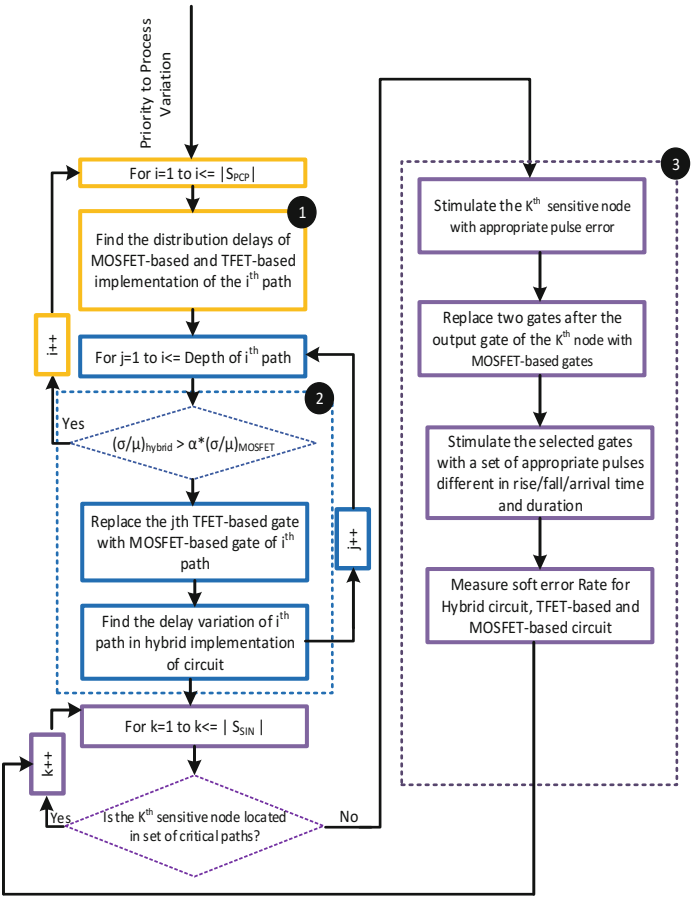


Fig. 10. Second part (priority on process variation mitigation) of the proposed algorithm.

which starts from the first gate of the path and continues to the last gate of the path. In each iteration, first, the delay variation of the path is extracted, and if the delay variation of the path ($(\sigma/\mu)_{hybrid}$) becomes smaller than the $\alpha \times (\sigma/\mu)_{MOSFET}$, the process for this path will be terminated. Otherwise, the TFET-based gate is replaced by the MOSFET-based gate, and the process is repeated for the next gate. The parameter α, which is a predefined coefficient specified by the designer, determines the expected delay variation of the hybrid TFET-MOSFET-based design compared to that of the MOSFET-based design. It should be noted that, for each new path, the hybridization process does not reconsider the gates which have been replaced by MOSFET-based ones in the previous iterations. This case may occur when some potential critical paths have a shared gate.

After applying the proposed method for increasing the timing yield of the circuit, we take an approach to decrease the soft error impact on the reliability of circuits (❸). In this step, the sensitive internal nodes are chosen one by one where for each chosen

sensitive node, all the paths which start from this node are extracted. Due to the better behavior of TFETs in generating transient currents, this type of gates are preferred. If, however, these gates are located in the critical paths, they are not replaced by TFETs if they are MOSFET-based gates as dictated by the previous hybridization part. If the gates are not in the critical paths, the third step of hybridization starts. For each extracted path, the first gate of the path is considered to be implemented by TFETs. This leads to generating a smaller voltage pulse due to the particle hit. The gates in the second and third levels are considered to be implemented by MOSFETs to electrically mask the generated voltage pulse. Our results show that one MOSFET-based gate is not able to fully mask the generated voltage pulse, while with a high probability, two consecutive MOSFET-based gates mask more effectively the generated pulse [12].

Next, we compare the soft error rates of the path under three different implementations of TFET-based, MOSFET-based, and hybrid TFET-MOSFET-based. If the error rate of the hybrid TFET-MOSFET implementation is smaller than the other two implementations, the hybridization for this path is terminated while, if the error rate of the hybrid TFET-MOSFET path is larger than that of the other implementations, the first gate after the last MOSFET-based gate (which is initially TFET-based) is considered to be implemented by a MOSFET one. This process is carried out till either the soft error rate of hybrid path reaches to a value smaller than the soft error rates of the two other implementations or there are no more gates in the path for the replacement. When the process of the hybridization for all the extracted paths of a sensitive node is terminated, the algorithm chooses another sensitive node and applies the above replacement procedure to the gates of its paths.

Note that replacing a TFET-gate by a MOSFET-gate may result in a capacitance increase of the internal nodes of the circuit connected to the inputs of this gate. Hence, after finalizing the hybridization of all the paths of a sensitive node, all the capacitances of the internal nodes of these paths are extracted. Now, if the capacitance of a node which belongs to the sensitive list increases to a value higher than the considered threshold value for the sensitive nodes, this node is removed from the sensitive nodes list.

Priority on decreasing the soft error rate

Third part of the algorithm deals with the case that the priority is on decreasing the soft error rate. The flow of the this part of the algorithm is depicted in Fig. 11. The overall flow for the hybridization technique in this part is almost similar to the one described in the previous subsection. For this case, first, the sensitive nodes are extracted where for any of the sensitive nodes, all the paths starting from the sensitive nodes are chosen. To have a soft-error resilient design, the first stage of each path is implemented in TFETs while two latter stages are implemented in MOSFETs due to the better error masking of MOSFETs (❶). Next, for each of the potential critical paths, the delay distribution of MOSFET-based and TFET-based designs are determine (❷). Finally, the hybridization technique is performed to increase the reliability of design (❸). The overall approach is as the one presented in the second step of the algorithm. In this case, however, if the gates that are located in the critical paths are the gates which are set to be implemented in TFETs due to transient current generation, they will remain unchanged and the hybridization is performed for the next step of the design.

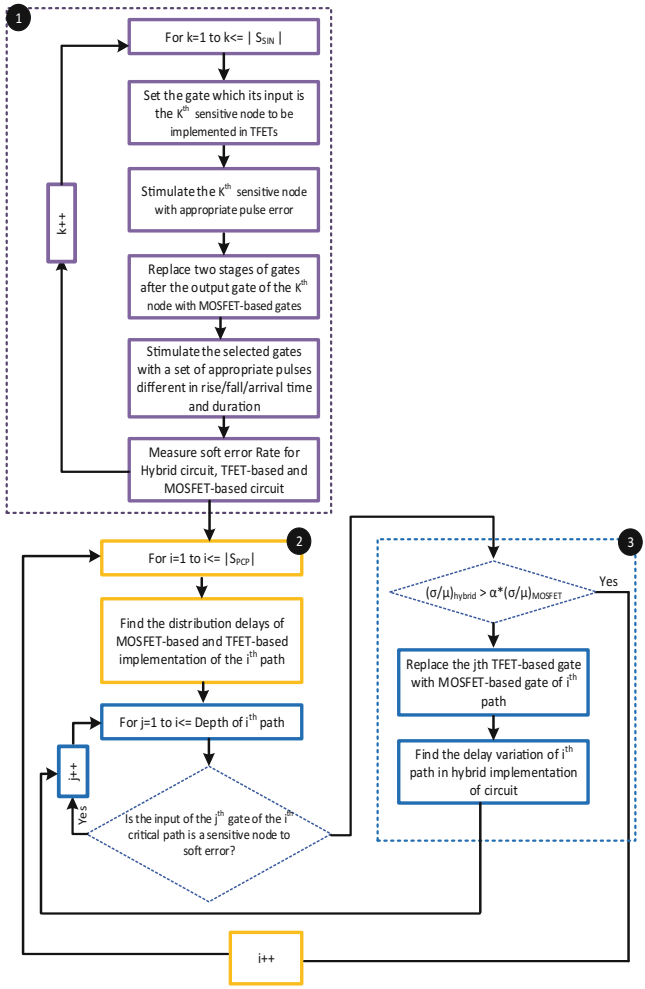


Fig. 11. Third part (priority on decreasing the soft error) of the proposed algorithm.

Now, in the following section, the efficiency of the proposed hybridization algorithm on increasing the reliability is evaluated for the two cases of priority with process variation mitigation and priority with soft error rate reduction. Also, the power consumptions for these circuit implementations is discussed.

4 Results and Discussion

4.1 Simulation Framework

In this study, the results of applying the proposed design approach to some sequential circuits from ISCAS'89 benchmark package are discussed. All the simulations were performed by utilizing the HSPICE tool. As mentioned before, we have used Verilog-A

models of the 22 nm double gate InAs TFET and 22 nm InAs MOSFETs during the HSPICE simulations. To perform the simulations, first, all of the considered circuits were synthesized to a gate-level netlists using Synopsys Design Compiler. In this work, without loss of generality, only Inverter, and 2-input NAND, NOR, AND, OR, XOR and XNOR gates were considered as the cells of the technology library. After extracting the gate-level netlist, the HSPICE netlist of the circuits were generated by using an in-house tool. In addition, the potential critical paths of the benchmarks were obtained using an in-house STA tool. It should be stated that the master-slave flip-flops of the selected benchmarks were replaced by the proposed hybrid TFET-MOSFET flip-flop of [12], to provide a more soft-error resilient design. Furthermore, the proposed heuristic hybridization algorithm was implemented using Python language. Finally, each of the studied circuits was implemented in three forms of MOSFET-based circuit, TFET-based circuit, and Hybrid TFET-MOSFET-based circuit. As mentioned before, to keep the overall performance of the circuit almost constant after the hybridization, the operation voltage was determined by the algorithm such that both TFET-based gates and MOSFET-based gates had about the same delays.

The particle hit in each critical node was modeled by injecting voltage pulses. The injected voltage pulses, which were totally 60, were different in rise time, fall time, arrival time compared to the edge of the clock, duration, and amplitude. The ranges of these parameters utilized in this work are given in Table 4. In the table, V_{error} refers to the nominal amplitude of the generated error for TFET and MOSFET transistors. During the simulations, error pulses with larger amplitudes were applied to MOSFET.

Table 4. The ranges of the values used for generating error injection voltage pulses

Parameter	Range of values
Rise time	10 ps to 100 ps
Fall time	10 ps to 100 ps
Arrival time compared to edge of the clock	−100 ps to +100 ps
Duration	10 ps to 300 ps
Amplitude	$V_{error} \pm 0.1$V

4.2 Comparison of Different Implementation Efficiencies

In this section, the efficacy of the proposed hybridization algorithm is evaluated. The normalized leakage power, energy consumption, delay variation, and soft error rate of the hybrid benchmarks are measured and compared to MOSFET-based and TFET-based designs. The normalized values of these parameters for each implementation of each benchmark circuit are shown in Fig. 12. In this figure, the values of the leakage power and energy consumption are normalized to the leakage power and energy consumption of the MOSFET-based design while the values of the delay variation and soft error rate are normalized to those of the TFET-based design.

As was mentioned in the previous section, the algorithm allows the designers to specify either the soft error or process variation mitigation as the higher priority. Here, we present the results for the application of the algorithm to each benchmark, considering both cases. As the results indicate, for both cases, the delay variation

mitigation and soft error rate reduction of the hybrid design are improved compared to those of the pure TFET-based design. Also, the delay variation (soft error rate) of circuits decrease more in the case that the process variation mitigation (soft error rate reduction) has the higher priority. In addition, the proposed design approach decreases the leakage power and energy consumption compared to those of the MOSFET-based design for both cases. It should be emphasized that the pure TFET-based design offers lower leakage power and energy consumption while pure MOSFET-based design provides smaller delay variation. The hybrid design has lower delay variation compared to that of the pure TFET design and smaller leakage power and energy consumption compared to those of the MOSFET design.

Now, we discuss these results in more detail. In the case of S838 (S713) circuit, the leakage power (energy consumption) decreases about 75% (67%) compared to the MOSFET-based design while, compared to the TFET-based circuits, the leakage power (energy consumption) only increases about 14% (15%). For the delay variation, the highest reduction belongs to the S838 benchmark with the delay variation of about 66% when process variation mitigation has the higher priority. When the priority is given to soft error resiliency of the design, the maximum reduction in the delay variation is about 58% belonging to S344 circuit. Considering the results for the soft error rates, the maximum reduction of 87% is for the S344 circuit in the case of the soft error reduction priority and the maximum reduction of 73% for the S344 circuit in the case where the priority is for the process variation mitigation. The study shows that, the proposed hybrid approach leads to, on average, 64% (48%) leakage power (energy consumption) reduction compared to the case of the MOSFET-based design when the process variation has the higher priority. Also, compared to the TFET-based design, it provides, on average, 52% delay variation reduction and 71% soft error rate reduction. On the other hand, when the priority is given to the soft error issue, the approach results in, on average, 67% (50%) leakage power (energy consumption) reduction compared to that of the MOSFET-based design as well as 42% (80%) decrease in the delay variation (soft error rate) decrease compared to that of the TFET-based circuit.

The results for the delay variation and soft error rate of the S27 and S344 circuits are the same in both priority cases of the proposed algorithm. It originates from the fact that, in these circuits, none of the sensitive nodes is located in the critical path. Therefore, the critical paths may be completely implemented using MOSFET-based gates wherever necessary. Similarly, we can keep TFET-based gates whenever shorter and smaller transient current are preferred. Hence, a complete hybridization is performed to reduce both delay variation and soft error rate.

Finally, to illustrate the significance of each of the defined priority cases in the algorithm, we define a parameter which is the ratio of the number of sensitive nodes located in the set of potential critical paths of the circuit to the total number of sensitive nodes. The ratio is called Sensitive Nodes in Critical Paths (SNCP). The larger the SNCP is, the more sensitive nodes are located in the critical paths. Hence, when the priority is given to the soft error rate reduction, the delay variation of the circuit becomes larger. On the other hand, if the priority is given to the process variation mitigation, the circuit with larger SNCP, will have greater soft error rate. Figure 13 depicts the SNCPs for different benchmarks. Interestingly, the S27 and S344 circuits have SNCP values of zero, making leading to the same results for both priority cases.

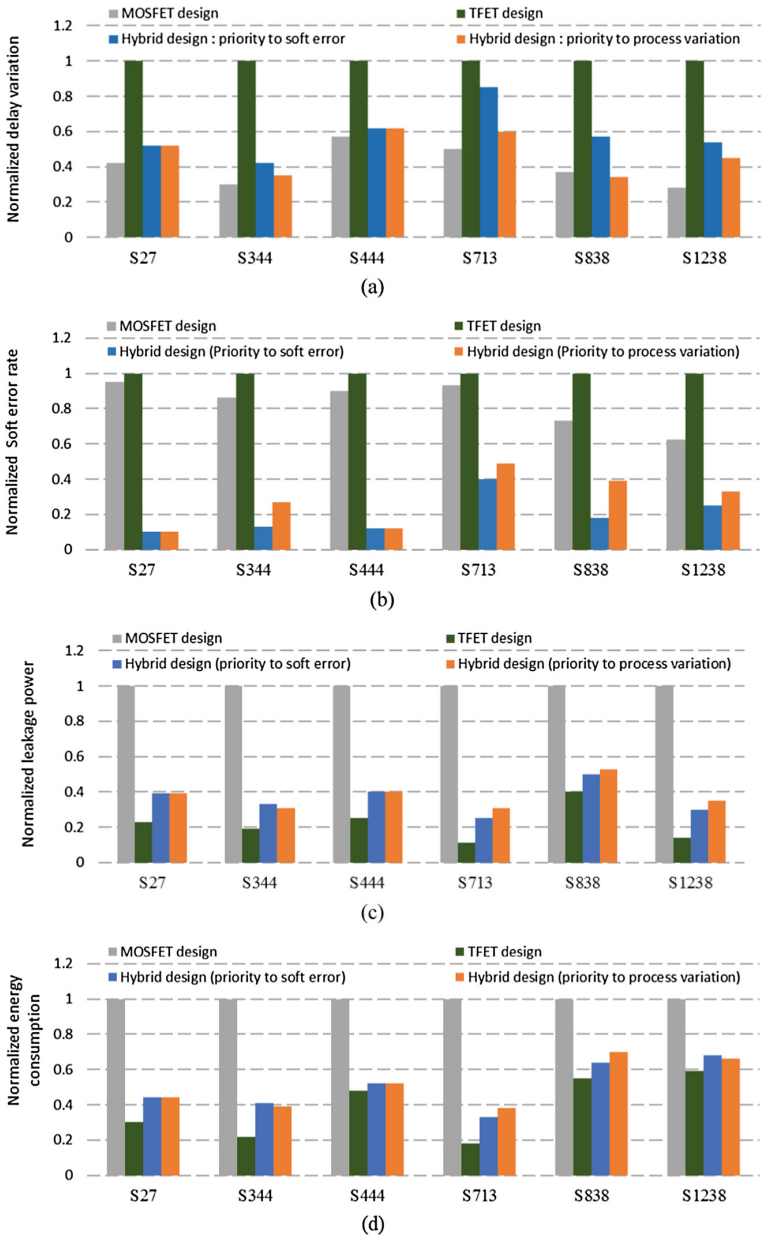


Fig. 12. Normalized (a) delay variation, (b) soft error rate, (c) leakage, and (d) energy, for some circuits from ISCAS'89 benchmark package.

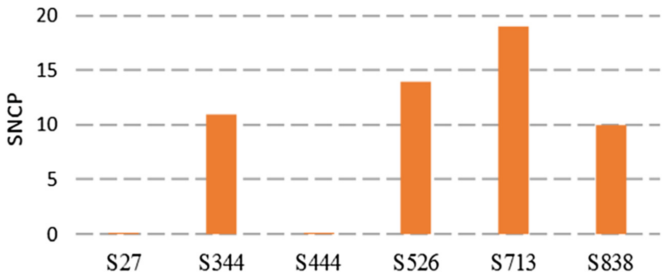


Fig. 13. Parameter SNCPs (Sensitive Nodes in Critical Paths) for different benchmarks.

5 Conclusion

In this paper, a hybrid TFET-MOSFET design was proposed to decrease the probability of timing violation and soft error rate in TFET-based designs. The hybridization method was inspired by the features of TFET and MOSFET devices. First, compared to MOSFET, the process variation impact was more on the TFET characteristics. Second, while the transient current, generated by a particle hit, was shorter and smaller for TFETs, MOSFETs had better electrical masking of these pulses. In this work, first, a III-V TFET model was selected and the impact of the process variation on its output electrical characteristic as well as soft error generation and propagation were investigated. Next, considering the operation and characteristics of both TFET and MOSFET devices, a heuristic algorithm was proposed for the TFET-MOSFET hybridization design. Finally, the proposed algorithm was applied to some sequential circuits of the ISCAS'89 benchmark package. The results showed that the delay variation of the TFET-MOSFET-based circuits, on average, would decrease about 52%, compared to that of the TFET-based circuits, if the priority were given to the process variation mitigation. The decrease in the variation was about 42%, on average, for the case with the soft error rate reduction priority. On the other hand, the soft error rate of the TFET-MOSFET-based circuits decreased about 80%, on average, when the priority was given to the soft error issue. The reduction of soft error rate was about 72%, on average, if the priority was given to the process variation mitigation. These results suggested that one TFET-MOSFET hybridization technique may be employed effective to improve the yield and soft error immunity characteristics of ultra-lower power circuits based on pure TFET-based design circuits.

References

1. Mukundrajan, R., Cotter, M., Saripalli, V., Irwin, M., Datta, S., Narayanan, V.: Ultra low power circuit design using Tunnel FETs. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 153–158. IEEE Press, MA (2012)
2. Kim, M.S., Liu, H., Swaminathan, K., Li, X., Datta, S., Narayanan, V.: Enabling power-efficient designs with III-V tunnel FETs. In: IEEE Compound Semiconductor Integrated Circuit Symposium (CSICs), pp. 1–4. IEEE Press, California (2014)

3. Chen, Y.-N., Fan, M.-L., Hu, V.-H., Su, P., Chuang, C.-T.: Evaluation of stability, performance of ultra-low voltage MOSFET, TFET, and mixed TFET-MOSFET SRAM cell with write-assist circuits. J. Emerg. Sel. Top. Circuits Syst. **4**, 389–399 (2014)
4. Migita, S., Matsukawa, T., Mori, T., Fukuda, K., Morita, Y., Mizubayashi, W.: Variation behavior of tunnel-FETs originated from dopant concentration at source region and channel edge configuration. In: 44th European Solid State Device Research Conference (ESSDERC), pp. 278–281. IEEE Press, Venice (2014)
5. Zhang, L., Chan, M., He, F.: The impact of device parameter variation on double gate tunneling FET and double gate MOSFET. In: IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC), pp. 1–4. IEEE Press, Hong Kong (2010)
6. Damrongplasit, N., Shin, C., Kim, S.H., Vega, R.A., Liu, T.-J.K.: Study of random dopant fluctuation effects in germanium-source tunnel FETs. J. Electron Devices. **58**, 3541–3548 (2011)
7. Hemmat, M., Kamal, M., Afzali-Kusha, A., Pedram, M.: Study on the impact of device parameter variations on performance of III-V homojunction and heterojunction tunnel FETs. J. Solid State Electron. **124**, 46–53 (2016)
8. Hemmat, M., Kamal, M., Afzali-Kusha, A., Pedram, M.: Hybrid TFET-MOSFET circuits: an approach to design reliable ultra-low power circuits in the presence of process variation. In: IEEE International conference on Very Large Scale Integration (VLSI-SoC), pp. 1–6. IEEE Press, Tallinn (2016)
9. Dhillon, Y.S., Diril, A.U., Chatterjee, A.: Soft-error tolerance analysis and optimization of nanometer circuits. In: IEEE International conference on Design, Automation, and Test in Europe, pp. 389–400. Springer, GA (2008)
10. Liu, H., Cotter, M., Datta, S., Narayanan, V. : Technology assessment of Si and III-V FinFETs and III-V tunnel FETs from soft error rate perspective. In: IEEE International Electron Devices Meeting (IEDM), pp. 25.5.1–25.5.4. IEEE Press, California (2012)
11. Datta, S., Liu, H., Narayanan, V.: Tunnel FET technology: a reliability perspective. J. Microelectron. Reliab. **54**, 861–874 (2014)
12. Hemmat, M., Kamal, M., Afzali-Kusha, A., Pedram, M.: Hybrid TFET-MOSFET circuit: a solution to design soft-error resilient ultra-low power digital circuit. J. Integr. VLSI J. **57**, 11–19 (2017)
13. Tura, A.: Novel Vertical Tunnel Transistors for Continued Voltage Scaling. Ph.D. dissertation, Univ. of California, Los Angeles (2010)
14. Mishra, A., Jha, K.K., Pattanaik, M.: Parameter variation aware hybrid TFET-CMOS based power gating technique with a temperature variation tolerant sleep mode. J. Microelectron. **45**, 1515–1521 (2014)
15. Liu, H., Datta, S.: III-V Tunnel FET model manual. The Pennsylvania state university (2015)
16. Cotter, M., Liu, H., Datta, S., Narayanan, V.: Evaluation of tunnel FET-based flip-flop designs for low power, high performance applications. In: 14th International Symposium on Quality Electronic Design (ISQED), pp. 430–437. IEEE Press, California (2013)
17. Miskov-Zivanov, N., Marculescu, D.: Modeling and optimization for soft-error reliability of sequential circuits. J. Comput. Aided Design Integr. Circuits Syst. **27**, 803–816 (2008)
18. Lin, S., Kim, Y.-B., Lombardi, F.: Soft-error hardening designs of nanoscale CMOS latches. In: 27th IEEE Symposium on VLSI Test, pp. 41–46. IEEE Press, Washington (2009)

Logic Synthesis for Silicon and Beyond-Silicon Multi-gate Pass-Logic Circuits

Valerio Tenace, Andrea Calimera(✉), Enrico Macii, and Massimo Poncino

Dipartimento di Automatica e Informatica, Politecnico di Torino,
Corso Duca Degli Abruzzi 24, Torino, Italy
{valerio.tenace,andrea.calimera,enrico.macii,massimo.poncino}@polito.it

Abstract. In the last decade several new technologies have been proposed as possible replacement for MOSFETs; Silicon Nanowires, Magnetic Tunnel Junctions, Graphene p-n Junctions are just some of the most representative examples. Although their intrinsic differences, they all share a common key characteristic, i.e., enable the implementation of logic gates with an expressive power much higher than that of state-of-art silicon CMOS gates. This may translate into more complex and faster switching functions that count less devices. The view of new materials that can serve as technological vehicles for energy efficient circuits and systems attracted the interested of the whole electronics research community. Apart from the many technological aspects, the path towards large-scale integration of emerging devices crosses the need of *(i)* new integration strategies that better fit the characteristics of the new technologies and *(ii)* new computer-aided design (CAD) methodologies able to cope with the complexity of today's design specs. The availability of this two elements may open the way for fast design space exploration and better assessment of new technologies against standard CMOS.

This work focuses on logic synthesis and optimization tools for ultra-low power pass-gate circuits mapped into emerging technologies, Graphene and silicon nano-wires. More specifically, we describe a novel multi-function decomposition engine that *(i)* efficiently performs abstract circuit modeling through a highly-compact data structure called Multi-Function Pass Diagram (MFPD), *(ii)* provides an effective multi-gate synthesis & optimization flow, *(iii)* allows accurate power/delay estimations. The contents reported in the following sections represent one of the first examples of how dedicated algorithms and data-structures can substantially improve the quality-of-design when moving from CMOS to emerging technologies.

Simulation run conducted on different benchmarks demonstrate that pass-gate circuits synthesized with the proposed tool are smaller and shallower, hence less power hungry and faster than circuits obtained through conventional synthesis methodologies based on standard design flows. As an additional contribution, the results prove that our solution is not only applicable to beyond-silicon technologies but also to standard MOSFETs.

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 60–82, 2017.
DOI: 10.1007/978-3-319-67104-8_4

Keywords: Emerging technologies · Graphene · Silicon nano-wires · Pass-gate logic · CAD · Logic synthesis · Low-power · Adiabatic computing

1 Introduction

1.1 CMOS at the End of the Line

The introduction of Metal-Oxide-Semiconductor Field-Effect-Transistors (MOSFETs), officially set in 1947 at Bells Labs as a replacement to vacuum tubes, represents a milestone in the industry of semiconductors. Since then, and after 60-year long research efforts, Complementary MOS (CMOS) electronic circuits have become the dominant technology for the entire ICT segment.

The continuous demand for smaller, faster and more power-efficient Integrated Circuits (ICs) have pushed CMOS technology close to its boundary. At the time being, technology trends are clearly highlighting that a radical shift in thinking digital hardware design might come soon. Among the many aspects, we highlight three well known issues that sustain this claim.

- **Non-ideality of the Silicon scaling process**: below the 45 nm node the technology scaling process faced several limitations due to (*i*) the increased difficulty in discretizing transistors on a physical die, a problem related to the gate-oxide thickness that is slowly approaching a few-atom width [1]; (*ii*) the miniaturized gate length of MOS transistors and the upsurge of short-channel effects (SCEs), leakage current in particular, which represent a serious reliability issue [2]; (*iii*) as transistors size decreases, power dissipation and process-variation induced reliability issues become critical [3]. These issues impact the fabrication yield of reliable ICs.
- **Nanometric CMOS styles are no longer the most energy-efficient integration strategy**: static CMOS has been taken as a reference style for mainstream VLSI circuits due to high noise immunity, resilience to supply-voltage scaling and low leakage currents. However, as the technology scaling process went below the 90 nm mark, some of these characteristics faded out due to SCEs. This suggests that other logic families that were discarded in the past, e.g., Dynamic-Logic, Pass-Transistor-Logic [4], may represent a new way out for low-power ICs.
- **Hitting the power-wall and the dark-silicon problem**: achieving ultra-low power consumptions is becoming a vital feature for consumer electronics, especially in the context of the Internet-of-Things (IoT) [5] where always-on, always-connected devices running sensoring applications and data-intensive computing represent the new mainstream paradigm. Even if many low-power techniques for CMOS circuits and systems are available today [6], e.g., Dynamic-Voltage-Frequency Scaling, Power-Gating, Multi-Threshold-Voltage and Reverse-Body-Biasing, the power consumed by CMOS based Systems-on-Chips (SoCs) architectures may exceed the power budget. This implies that an ever larger portion of the silicon die must be kept off (Dark-Silicon).

To address this issue, standard SoC architectures will make space to less power hungry solutions that implies the use of dedicated accelerators with embedded memory resources and more energy efficient circuitry.

For such reasons, soon or late, Silicon, CMOS and standard Von-Neumann architectures will drop the scepter in favor of emerging technologies, alternative integration strategies and new architectures. From a technological point of view, recent works proposed several options such as Ambipolar Silicon-NanoWires [7], Graphene p-n Junctions [8], Graphene Nanoribbons [9], Magnetic Tunnel Junctions [10] and Domain-Wall Nanowires [11]. Apart from their improved electrical characteristics, those technologies could enhance switching primitives with new fascinating properties able to accommodate the specifications of alternative computing paradigms.

1.2 Candidates to Replace the CMOS Technology

The above qualitative analysis suggests that the Silicon/CMOS pair could be soon replaced by some new material and a more energy efficient integration strategy. Among the many options we believe ambipolar technologies, such as Graphene or Silicon-Nanowires, integrated à la *pass-gate logic* style, a.k.a. *pass-transistor logic* (PTL), represent an interesting option. In particular, as it will be shown later in the text, Graphene-based devices are particularly suited for pass-gate logic.

The choice of PTL is justified by its high intrinsic efficiency, already proven for silicon technologies. PTL circuits can implement logic functions with a lower transistor count, smaller parasitic capacitance and hence better performance [12]. Even today's CMOS libraries make use of PTL for some logic gates, e.g., flip-flops and multiplexers, because of their efficient implementation. Moreover, PTL circuits offer an opportunity to work "adiabatically", namely, mimicking the adiabatic (i.e., without energy exchange) charging process [13]. Adiabatic PTL may find space with the implementation of dedicated hardware accelerators in charge of processing "slow" physical-data (e.g., biometric signals) with a very limited energy budget [5].

The use of PTL and, more precisely, adiabatic PTL, has been already proven for emerging technologies, such as nanoelectromechanical switches (NEMs), carbon nanotubes (CNTs), and graphene p-n junctions. For such devices the PTL style enables the design of logic circuits with improved energy efficiency if compared to CMOS [14–16].

1.3 Lack of Logic Synthesis Tools for PTL

It is clear that new hardware schemes, such as PTL, will inevitably ask for new CAD tools for the logic synthesis of digital blocks. Algorithms and data structures for the logic synthesis evolved following the growth of semi-custom CMOS libraries, while synthesis for PTL has been improved only marginally. This is why, even today, PTL remains underutilized [12]. It's not a coincidence

that most of the previous works do focus on circuits for very specific arithmetic functions [17,18] or handcrafted basic Boolean logic gates [4,19]. Indeed, when the target design turns into random logic, standard multi-level synthesis engines can't exploit the structural properties of PTL. That brings to sub-optimal implementations that typically require *ad-hoc* actions at the post-synthesis stage.

This problem is not new to the research community and several solutions have been introduced in the last years. Most of them, if not all, are closely related to the concept of Binary Decision Diagrams (BDDs) or some of its variants [12, 20,21]. There are two main reasons behind the use of BDDs. First, there exists a one-to-one matching between the BDD representation of the logic function and the final PTL circuit implementation; this enables the concept of *one-pass logic synthesis* [22] where logic optimization and technology mapping are carried out concurrently on the same data structure thereby saving CPU and memory usage. Second, BDDs [23] are a very mature data-structure with lots of available optimization algorithms for redundancy removal and circuit optimization.

Despite the efficiency of BDDs as data-structure is unquestionable, BDD-based synthesis tools show many limitations. First, the tree-like structure of BDDs reflects into a deep circuit topology with large depth, and hence large propagation delays. Second, state-of-the-art decomposition methods for BDDs construction all operate using a pre-fixed variable-order (VO), namely, the order used for variable expansion is fixed during the entire decomposition procedure, no matter what the logic function is. Since VO affects the vertex-set cardinality of BDDs, a wrong VO might result into dramatic area increase of the resulting circuit. Third, decomposition methods are constrained to a "single-function" decomposition. Such a function, here referred as $g(\mathbf{X})$, differs depending on the type of BDD variant in use, e.g., MUX for standard BDDs [23], XOR for Biconditional-BDDs [21]. Logic circuits dominated by $g(\mathbf{X})$, e.g., XOR-rich arithmetic circuits, take advantage of this characteristic, others, like random logic circuits, may suffer from sub-optimal minimization. While the first two issues have been addressed in [24] with the introduction of the *Pass Diagram* (PD) data-structure and the non-fixed VO decomposition, this work elaborates on the third issue, i.e., how to overcome "single-function" decomposition.

1.4 Contribution of This Work

As an extension of the contents proposed in [25], this work gives a comprehensive description of efficient abstract models and data-structures that are particularly suited for the synthesis of Multi-Gate Pass Logic (MGPL) circuits mapped with emerging technologies, Graphene and Silicon-NanoWires in particular, or, alternatively, with standard silicon MOSFETs.

An MGPL circuit consists of series connections of two-input *pass-gates* that can be turned-ON (OFF) and thus open (close) an electrical path between a clocked-power (the source) and the main output (the leaf); multiple paths are connected in parallel making the final logic circuit. Hence, similar to PTL, the information is not carried in the form of charges stored in parasitic capacitance, but rather through the root-to-leaf propagation of the clock-power signal. It is

worth noticing that, differently from any other existing PTL solutions, MGPL makes use of pass-gates that embed multiple Boolean operators, like AND, OR, NAND, and not just MUX or XOR as in the previous works; the choice of which operators depends on the technology in use.

We introduce a novel abstraction model, namely, the binary *Multi-Function Pass-Diagram* (MFPD), a graph-based representation for k-ary Boolean functions. An MFPD is a polarized, acyclic directed graph made up of N root-to-leaf logical paths. Each path is composed of an arbitrary number of two-input nodes connected in series, where each node represents a binary connective between two, out of k, primary input variables and can assume either a TRUE logical value, e.g., closed switch, or a FALSE logical value, namely an open switch. Under a specific input pattern, logical paths can be activated (all nodes are closed switches) in mutual exclusion (1 path out of N) and thus create a *gateway* from the root to the leaf. In such case, the equivalent logic function represented by the MFPD is evaluated as TRUE; on the contrary, when no active paths do exist, the logic function is said to be evaluated as FALSE. This structure matches the topology of a Multi-Gate Pass-Logic circuit.

The construction of a binary MFPD encompasses two major steps: *multi-function decomposition* using a set of basic Boolean operators, e.g., AND, OR, XOR and their complement; and *redundancy removal* through iterative reduction rules. Those phases have been integrated into an automatic synthesis and optimization tool named *Kanon*. Moving from single- to multi-function decomposition can be conceptually seen as the shift from two-level to multi-level synthesis carried out for CMOS circuits.

We apply our tool *Kanon* to a sub-set of generic benchmarks mapped onto three different technologies, i.e., Silicon MOS transistors, Ambipolar Silicon Nanowires and Graphene p-n junctions. The use of generic benchmarks avoids biased results due to the presence of circuits dominated by a specific function, the use of different technologies demonstrates that the proposed solution well fits both silicon and beyond-silicon technologies. The obtained MGPL circuits are then compared against standard PTL circuits synthesized using state-of-art BDD-based tools. The collected results validate the functionality of the proposed MFPD model and the related multi-function decomposition, whilst simulations using SPICE models quantify the energy efficiency of MGPL circuits.

2 Multi-Gate Pass Logic

A first example of pass-logic circuit for emerging devices has been recently proposed in [26] in the form of Pass-XNOR Logic (PXL) network using graphene p-n junctions. A PXL circuit consists of a network of Pass-XNOR Gates (PXGs); PXGs connected in series form a logic path, while logic paths connected in parallel connect the root of the circuit (fed by a clock-power signal) to the leaf (the main output). The clock-power signal works as an evaluation signal that eventually reaches the output when at least one parallel logic path is ON; in this case the logic function is evaluated as TRUE, i.e., 1-logic. When none of the parallel

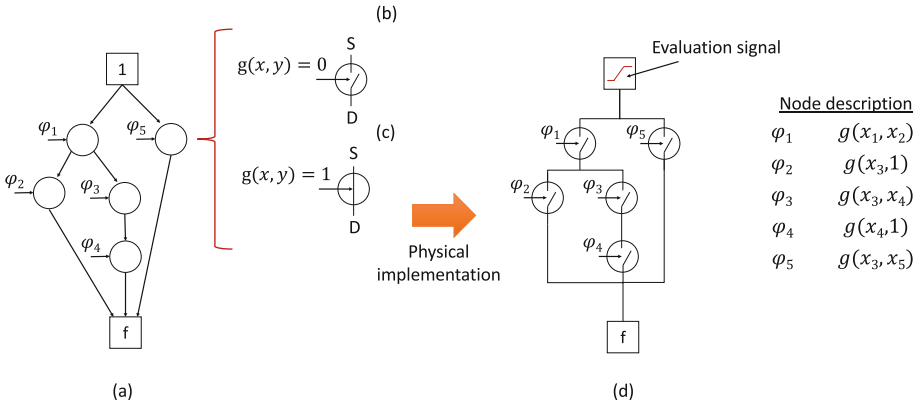


Fig. 1. MGLP circuit example, where $f = (x_1 \neg\vee x_2) \wedge [(x_3 \neg\vee 1) \vee ((x_3 \neg\vee x_4) \wedge (x_4 \neg\vee 1))] \vee (x_3 \neg\vee x_5)$

logic paths is ON, the propagation of the clocked-power signal is inhibited and the logic function is evaluated as 0-logic.

A logic path is ON *iff* all its series connected PXGs are ON simultaneously. The PXGs can be seen as logic primitives whom electrical behavior resembles a CMOS transmission gate with an embedded XNOR Boolean functionality. More specifically, each PXG is electrostatically controlled through primary input logic signals that tune the equivalent resistance of the PXG itself; a PXG fed by logic signals having same polarity shows a low-resistance, the ON state, whereas logic signals with opposite polarity lead the PXG to high-impedance, the OFF state.

As a further step to achieve a higher level generalization of the Pass-XNOR Logic (PXL) style, [25] introduces the concept of *Multi-Gate Pass Logic* (MGPL). The physical primitives of an MGPL network are generic *pass-gates* (PGs), that, from a functional point of view, can be seen as *function*-controlled switches. Similar to PXGs, they consist of two *logic-terminals* fed by the input logic signals (x and y in Fig. 1-(b)), and two *transmission terminals*, one playing as the source of an evaluation signal and the other as the collector (S and D in Fig. 1-(b)). The control function is a two-input Boolean operator $g(x, y)$ between the x and y logic inputs; when $g(x, y) = 1$ the PG is ON (low-resistance), Fig. 1-(b), when $g(x, y) = 0$ the PG is turned OFF (high-impedance), Fig. 1-(c). PGs with different control functions can be designed depending on the technology in use; the example in Fig. 1 is for a NAND-PG, i.e., $g(x, y) = x \neg\vee y$. Notice that an MGPL circuit can contain PGs with different embedded functions. Similar to PXL, an MGPL (Fig. 1-(d)) consists of logic paths connected in parallel between a clocked-power supply (the root) and the main output (the leaf). Each path consists of a cascade of independent PGs driven by primary inputs. When activated (all PGs turned-ON), a logic path creates a low-resistive gateway through which the clocked-power signal can flow from the root to the leaf. Under this condition the circuit's output is evaluated as 1-logic. Logic paths are in mutual

exclusion by construction, that is, for a given input pattern one and only one path can be eventually activated. When there are no activated paths the circuit's output is evaluated as 0-logic. An MGPL circuit can be modeled using a new dedicated abstract model, the Multi-Function Pass-Diagram (MFPD), Fig. 1-(a), described in the next section.

As for other dynamic logic families, the logical computation of MGPL circuits consists of two distinct phases: the *configuration* phase and the *evaluation* phase. During the former, primary logic inputs, i.e., the literals composing the logic function, are fed to the logic inputs of the pass-gates. At the end of this phase the doping profile of each and every device is fixed and the resistive paths of the network are set up. In the evaluation phase the clocked power signal is precharged and propagated through the network. A pulse detected on an output leaf evaluates the implemented function as TRUE; in this regard, a Sense Amplifier can be used for each output cone in order to quickly identify the 1-logic and reshape the clock-supply signal [27].

It is worth emphasizing that although the MGPL resembles the PTL structure, the difference is substantial. In PTL circuits, transistors are used as switches that deviate the current flow to different paths; on the contrary PGs are used as switches to open/close a logic path. This is reflected by the model used to represent the circuit. Indeed, BDDs are not the most intuitive representation as PG gates do not implement any deviation of the signal. Second, while in PTL an output is always connected to a static power supply terminal, V_{dd} if '1' or *Gnd* if '0', output evaluation in MGPL logic is dynamic: current is flowing if '1', not flowing if '0'. Alternatively, one can see MGPL circuits as a half way between CMOS and PTL. As in CMOS series/parallel connections between gates are available, as in PTL, information is carried out by means of root-to-leaf current flow.

2.1 Pass-Gate Devices

New logic primitives introduced by emerging technologies represent a perfect fit to the structure of PGs. Figure 2 pictorially describes some of them. In particular, Fig. 2-(a) shows four PG embodiments using Ambipolar Silicon-NanoWires (SiNW) [7]. The first two (left) are composed of a single SiNW transistor and implement the AND and NOR logic gates. The remaining two (center and right) consist of a pair Si-NW transistors and implement the XNOR or XOR logic gates.

Figure 2-(b) shows two possible pass-gates using standard MOSFET transmission-gates. The first one (left) implements the AND, whereas the second one (right) implements the NOR. Since both configurations require four MOSFETs, silicon devices have less expressive power when compared to SiNWs.

Finally, Fig. 2-(c) shows pass-gates mapped on graphene p-n junctions [28]. A graphene p-n junction consists of two metal back-gates (blue and green triangles) driven by logic signals (x and y). Logic signals with same polarity turn the junction ON. The first PG (top left) implements the NOR gate; the outer input connections x and y are both compared to a logical-0 reference. It works as

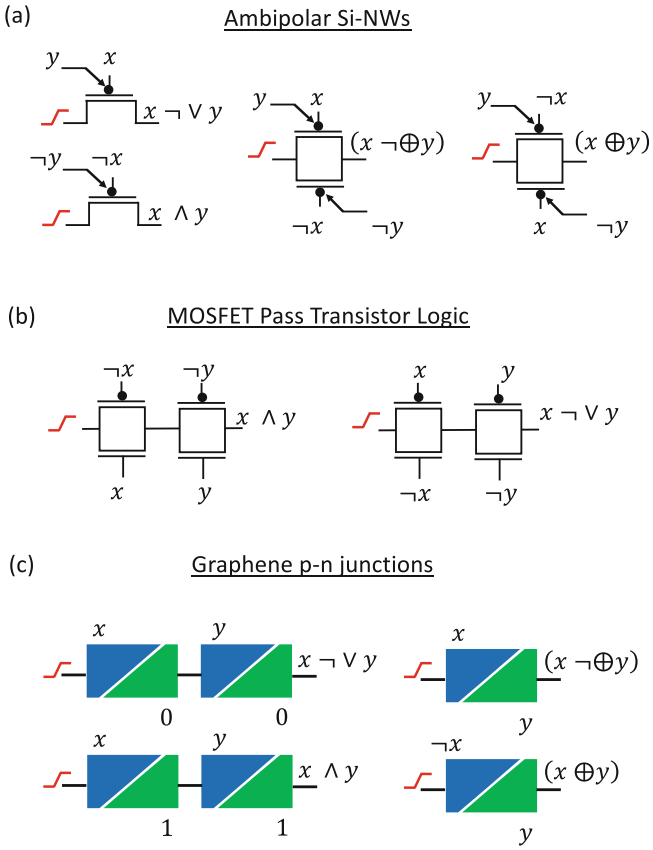


Fig. 2. Possible PGs for different logic primitives. (Color figure online)

follows: when both x and y are set to 0-logic, the input evaluation signal (red ramp) is allowed to propagate; in all the remaining cases at least one p-n junction is OFF and the evaluation signal is stopped. Similarly, the second pass-gate (bottom left) implements the AND; the evaluation signal propagates *iff* both x and y are fed wit 1-logic. Notice that for NOR and NAND SiNWs need less devices (1 vs. 2). The last two pass-gates (top and bottom right) implement the XNOR and XOR gates. In this case graphene shows higher expressive power than SiNW. It is therefore clear how different technologies can be better exploited using different logic primitives.

2.2 Delay and Power Modeling of MGPL Circuits

The total delay D_p of an MGLP logic circuit can be estimated as the sum of delays due to the configuration phase D_{conf} and the evaluation phase D_{eval}, as

described in (1), where D_{conf} is the time primary logic inputs take to charge

$$D_p = D_{conf} + D_{eval} \tag{1}$$

the parasitic capacitances at the back-gates, whereas D_{eval} is the propagation delay of the input pulse through the front resistive paths of the network.

The amount of power consumed during the configuration phase P_c is due to charging/discharging of the input gate capacitance of the PGs. For a circuit made up N gates, an approximate, yet accurate model borrowed from CMOS is reported in (2), with P_{PG_i} as the power consumed from the i-th PG, V_{dd} as the supply voltage, f the operating frequency, C_i the input capacitance of the i-th PG, and $E_{SW_{i,j}}$ representing the probability that the input signal makes a transition.

$$P_c = \sum_{i}^{N} P_{PG_i} = \sum_{i}^{N} \sum_{j=1}^{2} 0.5 V_{dd}^2 \cdot f \cdot C_{in} \cdot E_{sw_{i,j}} \tag{2}$$

During the evaluation phase, once primary inputs have settled and PGs have been turned-ON or OFF, the circuit simply reduces to an equivalent resistor R_{eq}, i.e., the sum of the ON resistances R_{ON} of the PGs belonging to the ON-path, in series with the output load capacitance C_l. The average power consumed P_e can be therefore obtained as described in (3), where t_{rf} is rise/fall output transition time, and $i_{C_l}(t)$ is the current charging C_l. Notice that P_e is consumed iff the output is TRUE, while it is almost zero otherwise.

$$P_e = \frac{1}{t_{rf}} \int_0^{t_{rf}} R_{eq} i_{C_l}^2(t) dt = \frac{R_{eq} C_l^2}{t_{rf}^2} V_{dd}^2 \tag{3}$$

Moreover, for values of T_{rf} large enough, the evaluation phase completes at zero-power, namely, *adiabatically.*

3 Building MFPDs

3.1 Multi-function Decomposition

The decomposition of a logic function through the primitives made available by the technology in use represent a fundamental step of any logic synthesis algorithm. Since most techniques leverage multi-level logic representations, in this section we illustrate an *ad-hoc* decomposition that is fine-tuned for pass-gates logic. Such decomposition, named *multi-function* decomposition, relies on the basic assumption that any Boolean equation given in the form of *sum-of-products* (SOPs), or *product-of-sums* (POSs), can be decomposed by means of a user-defined set of logic connectives $\mathcal{G} = \{g : B^2 \rightarrow B\}$. Let us assume a function $f(S)$ with support-set $S = \{x_1, x_2, x_3\}$ described with the following SOP:

$$f(S) = (x_1 \wedge \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) \tag{4}$$

By resorting to the distributive property and the identity rule, it is possible to expand the function $f(S)$ as a sequence of cubes, having cardinality of two literals:

$$\begin{aligned} f(S) = (x_1 \wedge \neg x_4) \vee (\neg x_1 \wedge \neg x_2) \\ \wedge (x_3 \wedge 1) \vee (x_1 \wedge x_2) \wedge (x_3 \wedge 1) \end{aligned} \tag{5}$$

Each product can then be rewritten using the Boolean connectives $g \in \mathcal{G}$ by means of duality. For instance, let us assume the availability of two connectives $\mathcal{G} = \{\{x \neg\vee\ y\}, \{x \neg\oplus y\}\}$, where the first one, the NOR ($\neg\vee$ symbol), has higher priority, i.e., is processed first. This means that the function $f(S)$ could be *NOR*-decomposed as shown in (6).

$$\begin{aligned} f(S) = (\neg x_1 \neg\vee x_4) \vee (x_1 \neg\vee x_2) \wedge (\neg x_3 \neg\vee \neg x_3) \\ \vee (\neg x_1 \neg\vee \neg x_2) \wedge (\neg x_3 \neg\vee \neg x_3) \end{aligned} \tag{6}$$

Such reformulation reveals that $(\neg x_3 \neg\vee \neg x_3)$ is a common term, that can be factorized as reported in (7).

$$\begin{aligned} f(S) = (\neg x_1 \neg\vee \neg x_4) \\ \vee (\neg x_3 \neg\vee \neg x_3) \wedge [(x_1 \neg\vee x_2) \vee (\neg x_1 \neg\vee \neg x_2)] \end{aligned} \tag{7}$$

At this point, the second operator in $\mathcal{G}$, the XNOR ($\neg\oplus$ symbol), could be applied on (7) in order to further reduce the number of literals in (5). Indeed, the term $(x_1 \neg\vee x_2) \vee (\neg x_1 \neg\vee \neg x_2)$ can be represented as the XNOR between x_1 and x_2. We refer to this operation as *Boolean substitution*. Eventually, the final result of the multi-function decomposition, the original function (4) is decomposed as described in (8).

$$f(S) = (\neg x_1 \neg\vee x_4) \vee (x_1 \neg\oplus x_2) \wedge (\neg x_3 \neg\vee \neg x_3) \tag{8}$$

Similarly, it is possible to assume a library of logic connectives described as $\mathcal{G} = \{\{x \neg\wedge y\}, \{x \neg\oplus y\}\}$, where the symbol $\neg\wedge$ denotes the NAND operator. In this case, the same Boolean function described in (4) is *NAND*-decomposed as described in (9), and thus optimized by means of the *XNOR* connective, as reported in (10).

$$\begin{aligned} f(S) = (\neg x_1 \neg\wedge \neg x_4) \vee \neg(\neg x_1 \neg\wedge \neg x_2) \\ \wedge (\neg x_3 \neg\wedge x_3) \vee (\neg x_1 \neg\wedge x_2) \wedge (\neg x_3 \neg\wedge x_3) \end{aligned} \tag{9}$$

$$f(S) = (\neg x_1 \neg\wedge \neg x_4) \vee (x_1 \neg\oplus x_2) \wedge (\neg x_3 \neg\wedge x_3) \tag{10}$$

It is easy to check the Boolean equivalence between (8), (10) and the original function (4); in terms of savings, both (8) and (10) show 25% literal savings.

As far as the efficiency is concerned, the proposed multi-function decomposition is closely related to (*i*) the set of Boolean operators and (*ii*) their priority ordering in $\mathcal{G}$. Although several options do exist, we resort to a technology-instructed strategy, namely, available operators in $\mathcal{G}$ are sorted, from highest to lowest, in terms of their *expressive power* (EP), which describes the ratio

between the complexity of the logic operator and the number of devices needed to implement the corresponding logic gate. This guarantees the high flexibility and orthogonality of our tool onto different technologies.

As will be shown later in the text, different primitives are used during different stages of the multi-function decomposition. For the sake of clarity we define the first operator in $\mathcal{G}$, the one with the highest EP, as the *primary* primitive, the remaining ones as the *secondary* primitives.

3.2 Multi-Function Pass Diagrams (MFPDs)

The synthesis of MGPL circuits needs an abstract model for reasoning and optimization. We introduce the MFPD, a simple, yet efficient abstract model for one-pass synthesis of MGPL circuits.

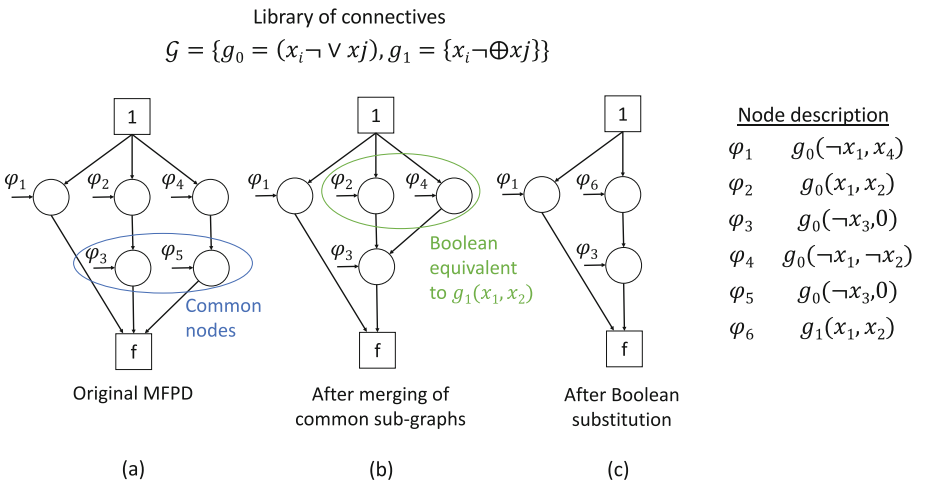


Fig. 3. MFPD of function of Eq. (6) before optimization (a), after merging of common sub-graphs (b), and after Boolean substitution (c).

Given a generic multi-input/single-output Boolean function f with support-set $S = \{x_1, ..., x_N\}$, its MFPD (Fig. 3) representation is a polarized, directed acyclic graph defined as $G = (\Phi \cup V \cup \Theta \cup A)$. The set of internal nodes $v \in V$ are labeled as $g(x, y)$, with $g \in \mathcal{G}$ a two-input primitive Boolean connective and $x, y \in S$. Each internal node v has one outgoing edge $a \in A$ representing the logical conjunction (AND) with the successor node. The terminal node with *indegree* 0 represents the root of the MFPD, where the function starts to be evaluated; the terminal node with *outdegree* 0 is the leaf of the MFPD, the output of the function f. Multiple output functions are represented by many MFPDs as the number of outputs. As an example, Fig. 3-(a) shows the MFPD structure for the function (6) with $g_0 = (x_i\neg \vee x_j)$ and $g_1 = (x_i\neg \oplus x_j)$.

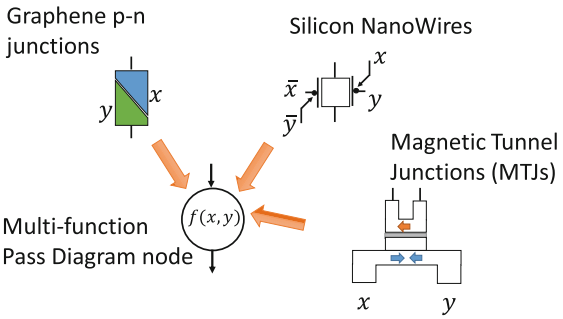


Fig. 4. Multi-Function Pass Diagram node mapping with different technologies.

The main strengths of an MFPD are three. First, they guarantee the capability of supporting multi-function decomposition. This degree of freedom comes at the cost of canonicity, that is, MFPDs do not have an unique representation of Boolean formulae. However, relaxing the canonicity constraint is a well-accepted concept in the EDA community; indeed, And-Inverter Graphs (AIGs) integrated into commercial multi-level logic synthesis tools are non-canonical representations, but nonetheless they are likely used because more compact and manageable. Second, a MFPD has a 1-to-1 mapping to the final circuit implementation, that is, each internal node is implemented by a single logic primitive in the resulting network (please refer to Fig. 1-(a)). This makes the MFPD a universal model for both the optimization stage and the technology mapping, a key aspect for *one-pass logic synthesis* [22]. Third, but not for importance, MFPDs may serve as model for MGPL circuits mapped using different emerging devices. As an example, Fig. 4 depicts a generic MFPD internal node, with $f = x \neg\oplus y$, and its implementation using Graphene p-n junctions, Silicon-NanoWires, Magnetic Tunnel Junctions and Memristors. Thus, the proposed MFPD model and its synthesis methodology can be seen as orthogonal tools for the assessment of a wide range of emerging technologies.

4 Algorithms

4.1 Building MFPD

Algorithm 1 describes the pseudo-code of the *Build* routine we implemented for multi-function decomposition and MFPD construction.

The main input parameters are (i) a tabular description T of the Boolean function and (ii) the primary connective (the first operator in the list of primitives $\mathcal{G}$). Table T can be a non-minimized *implicant* table (i.e., not prime) and can be obtained through any Verilog compiler, e.g., *ABC* [29]. We refer to T as the *PLA* table. As an example, Table 1 shows the PLA table for the Boolean function (5), where the character '-' identifies a dont'care.

Table 1. PLA table of function (5)

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	x_4	**f**
1	-	-	0	1
0	0	1	-	1
1	1	1	-	1

Algorithm 1. MFPD build

Input: PLA Table T, Primary connective $g_0 \in \mathcal{G}$
Output: Multi-Function PD $MFPD$

```
1  MFPD = ∅
2  foreach row R ∈ T do
3  |  CUBES_R = ∅
4  |  DontCareSet = DetermineDCS(R)
5  |  foreach primary input PI ∈ R do
6  |  |  if PI ∉ DontCareSet then
7  |  |  |  CUBES_R.append(PI)
8  |  |  end
9  |  end
10 |  foreach v_{i,k} ∈ CUBES_R do
11 |  |  NewNode ← SetPolarity(v_{i,k}, g)
12 |  |  PT_R.append(NewNode)
13 |  end
14 |  MFPD.append(PT_R)
15 end
```

The MFPD is generated branch-wise, that is, for each row of the PLA table, i.e., for each product term of the function, nodes are appended in series by iterating the following sequence of operations:

Cube sequence generation (line 3–9) – variables not belonging to the don't-care set are included in the cube list $CUBES_R$ in order of appearance; those belonging to the dont-care set are dropped. For odd sequences, the last variable is paired with '1' logic so as to maintain Boolean equivalence. For instance, considering Table 1, for the first row $CUBES_1 = \{(x_1, \neg x_4)\}$, for the second row $CUBES_2 = \{(\neg x_1, \neg x_2), (x_3, 1)\}$, for the third row $CUBES_3 = \{(x_1, x_2), (x_3, 1)\}$.

Node generation (line 10–14) – for each pair of cubes stored in $CUBES_R$, the polarity of the variables are fixed according to the *primary* Boolean connective g and the resulting nodes are appended on the current branch. Let us consider $CUBES_2$ which contains two cubes, $(\neg x_1, \neg x_2)$ and $(x_3, 1)$; with g the NOR operator (like the example in Sect. 3.1), variables are complemented (by De-Morgan) as (x_1, x_2) and $(\neg x_3, \neg x_3)$ respectively.

Algorithm 2. MFPD optimization algorithm

```
   Input: MFPD, Secondary connectives G = (g_1, ... g_m) ∈ 𝒢
   Output: Optimized Multi-Function PD OMFPD
1  OMFPD = ∅
2  foreach path P ∈ MFPD do
3  |  C_M ← ∅
4  |  C_E ← ∅
5  |  foreach path Q ∈ MFPD, with Q ≠ P do
6  |  |  if SameSupport(P, Q) then
7  |  |  |  if CheckBoolSub(P, Q, G) then
8  |  |  |  |  C_E.append(Q, g_k ∈ G)
9  |  |  |  end
10 |  |  else
11 |  |  |  if SharedNodes(P, Q) then
12 |  |  |  |  C_M.append(Q)
13 |  |  |  end
14 |  |  end
15 |  end
16 |  if |C_E| > 0 then
17 |  |  M ← ApplyBoolSub(P, C_E, G)
18 |  else if |C_M| > 0 then
19 |  |  M ← MergeIsomorphic(P, C_M)
20 |  OMFPD.append(M)
21 end
```

Given a table T with N implicants and M literals, the proposed *build* routine has a complexity of $O(N \cdot M)$.

4.2 Optimization

Algorithm 2 describes the pseudo-code of the optimization stage for redundancy removal. It implements two different optimization techniques: *(i)* node elimination by *Boolean substitution*; *(ii)* merging of isomorphic sub-graphs. While the latter is reminiscent of standard reduction rules from BDDs [23], the former one is an ad-hoc strategy for MFPDs. Its purpose is to find suitable equivalent logic connectives, among the list of *secondary* connectives in $\mathcal{G}$, that can be eventually substituted in order to enable node elimination and reduce the MFPD cardinality; as illustrated in the examples of Sect. 3.1. Please note that *secondary connectives* are selected with a greedy approach, that is, the first one that satisfies the Boolean equivalence is instantiated in the network.

Input parameters of Algorithm 2 are the MFPD obtained through the *MFPD Build* routine, and the list of secondary connectives $G \in \mathcal{G}$.

Candidate selection (line 3–15) – Each root-to-leaf path P of the MFPD is compared with any other path Q ($P \neq Q$). If (line 6) P and Q share the same

support set (i.e., nodes in P and Q are driven by the same literals) the algorithm checks (line 7) whether it is possible to perform a *Boolean substitution*, namely, it checks whether some of the operators associated with the nodes in Q can be substituted with some other operator $g_k \in G$ s.t. Boolean equivalence is satisfied. If so, P and Q share a common node expressed by means of g_k, that allows to merge P and Q in a single path. Therefore, Q is stored in the candidates list C_E together with the connective g_k that enables its elimination. If P and Q do not have common support set (line 10), the algorithm checks whether a path Q shares at least one node with P; if so, Q is a potential candidate for node merging and it is temporarily stored in the list of candidates C_M.

Merge and Eliminate (line 16–20) – once candidates have been selected, the algorithm first evaluates whether there exists at least one candidate for node elimination by *Boolean substitution* ($|C_E| > 0$). If so, the common node between P and C_E is replaced with the new connective g_k, and redundant paths in C_E are removed (*ApplyBoolSub* function). If not and the list C_M is not empty, then common nodes between C_M and P are evaluated for merging through the *MergeIsomorphic* function.

Figure 3-(b) and -(c) show the results of the optimization procedures described above applied on the MFPD obtained through the build function (Fig. 3-(a)).

Since Boolean elimination guarantees higher benefits in terms of node count and circuit complexity, e.g., it reduces the number of common branches due to merge operations, this characteristic is exploited first during the last phase of the optimization process with the $MergeAndRemap$ function. Otherwise, common nodes between the $CANDIDATES_M$ list and P are merged by means of the $MergeMaxSharing$ function. The final result is then appended in the optimized MFPD structure $OMFPD$. Applying this procedure to the MFPD depicted in Fig. 3-a allows us to achieve a more compact representation of the same function by means of only three logical nodes/operations, as depicted in Fig. 3-b.

Concerning complexity, since each path P is compared with any other path Q, the total number of loops is $\frac{N \cdot (N-1)}{2}$, with N the number of paths in the starting MFPD. The complexity of the optimization routine is $O(N^2)$. Notice that all other sub-routines have a $O(1)$ complexity (operations are completed in constant time) except for functions *SameSupport* and *SharedNodes* which show a complexity of $O(M)$, with M the number of nodes in the path Q.

5 Simulation Results

Experimental results reported in this section provide a fair comparison of MFPDs and the resulting MGPL circuits mapped onto different technologies against state-of-the-art data-structures and PTL circuits. The goal is to demonstrate that:

1. MFPDs optimized with the eduction rules described in Sect. 3 give substantial savings.

2. MFPDs represent the most compact solution for the representation of switching functions implemented through the MGPL style; metrics adopted are *expressive power*, namely the number of nodes and *depth*, i.e., nodes count along the longest path.
3. MFPDs are a true technology-independent abstract model, namely, the way they model a Boolean function can be orthogonally applied to any kind of technology and, most importantly, whatever the logic primitives are.
4. MGPL circuits obtained through MFPDs might become the vehicle for ultra-low power computing using emerging technologies, graphene in particular; in this regard, we show that the MGPL style allows large gains w.r.t. PTL and, most importantly, it is well suited for ultra-low power digital circuits.

We set up five different synthesis flows, the first four are for pass-gate logic circuits, the target of this work, the fifth one is for standard cells-based circuits.

- **MFPDs** (the solution proposed in this work): circuits described using the PLA format [29] are processed with our tool *Kanon* for multi-function decomposition using the connective set $\mathcal{G} = \{\{x\neg \vee y\}, \{x\neg \oplus y\}, \{x \oplus y\}\}$. It is worth to notice that even though more Boolean operators can be used, here we force our tool working in worst-case conditions where only three primitives are allowed. Resulting MFPDs are then mapped onto MGPL circuits using different technologies.
- **PDs** (introduced in [24]): circuits described using the PLA format [29] are processed leveraging *Gemini*, a single-function XNOR decomposition tool; resulting PDs are mapped onto PXL circuits using different technologies.
- **Biconditional-BDDs** (described in [21]): circuits are first synthesized using a standard multi-level synthesis tool and then translated into BBDDs using single-function XOR decomposition scheme; the resulting BBDDs are mapped onto a tree-based PTL-like structure using different technologies.
- **BDDs**: circuits are processed with a C program that leverage the CUDD package [23]; BDD structures, obtained with a single-function MUX-based decomposition, are mapped on PTL-like circuit using different technologies.
- **AIGs**: obtained with the ABC synthesis tool [29], the AIGs are mapped on a CMOS library containing only AND and INV gates; it is worth emphasizing that AIGs cannot be directly used for pass-gates logic circuit, however, they serve as a reference point to better evaluate MFPDs.

The experiments were run on a set of open-source benchmarks from the LGSynth91 suite [30], and accurate SPICE simulations were used for the characterization of the obtained netlists. Please note that the size of such benchmarks is comparable to that of those used in other synthesis-related works, e.g., [21]. Without loss of generality, only combinational logic cones have been considered for synthesis, i.e., in-to-out and register-to-register logic cones.

5.1 Efficiency of Reduction Rules During MFPD Optimization

Table 2 gives a brief description of the adopted benchmarks and an overview of the efficiency of the algorithms proposed in this work. Columns **PI**, **PO** and **I**

Table 2. MFPD reduction rule efficiency.

	PI	PO	I	Number of nodes		
				w/o opt	w/ opt	Savings %
sao2	10	4	58	229	152	33.62
o64	130	1	65	65	65	-
5xp1	7	10	75	161	111	31.06
c8	28	18	79	156	108	30.77
duke2	22	29	87	401	287	28.43
apex1	45	45	206	921	677	26.49
misex1	8	7	32	67	31	53.73
misex2	25	18	29	101	75	25.74
b12	15	9	431	1007	579	42.50
k2	45	45	936	3791	2103	44.53
bigkey	486	421	6151	19054	10771	43.47
s13207.1	700	790	10987	53868	33005	38.73
Total				**79821**	**47964**	**Avg. 39.91**

represent the total number of primary inputs, primary outputs and implicants of each benchmark. Under the label **Number of nodes**, column **w/o opt** refers to MFPDs after the build process, whereas column **w/ opt** refers to MFPDs after optimization; column **Savings** reports optimization gain;

As the table suggests, the proposed reduction rules allow about 40% of savings on average. Noticeably, large savings have been recorded for all the benchmarks, except for `o64`. For this case we observed the PLA table is a diagonal matrix of '1s' which prevents MFPD optimization.

5.2 Compactness of the MFPD Model and Execution Time

Figure 5 depicts the obtained synthesis results averaged over all the benchmarks described in Table 2. The plot suggests that MFPDs outperform BBDDs, which are 35.39x larger, BDDs (15.51x larger) and PDs (2.1x larger); only AIGs are more compact (0.63x). Indeed, AIGs leverage two key options available in multi-level optimization, namely the possibility of reusing cascades of common sub-expressions and local don't-care conditions, a technique that is not applicable to pass-gates logic styles.

However, an important aspect concerns the depth of the data-structures. In this case MFPDs are the most efficient solution since BBDDs (21.12x deeper), BDDs (20.33x deeper) and PDs (1.83x deeper) show an higher number of devices on the longest path. AIGs do the same as they return circuit topologies 3.83x

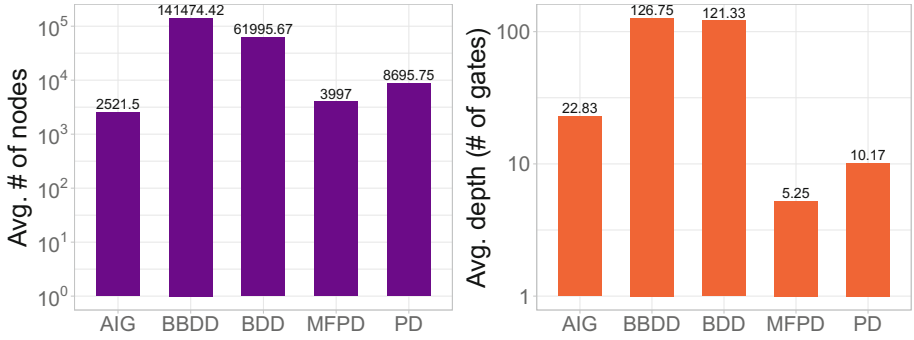


Fig. 5. Binary MFPDs efficiency w.r.t. PDs, BDDs, BBDDs and AIGs. Average number of nodes (left), average depth (right).

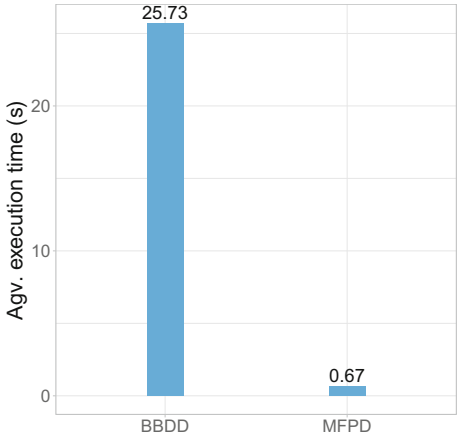


Fig. 6. Average device count after synthesis and mapping. Graphene (left), Ambipolar Si-NWs (center), MOSFET PTL (right).

deeper. Therefore, MFPDs are well suited for pass-gates logic circuits, where smaller depth translates into shorter delays and smaller voltage noise.

Such huge savings achieved are the consequence of the efficient multi-function decomposition, in particular: (*i*) the availability of more Boolean operators w.r.t. BDDs, BBDDs and PDs, (*ii*) the fact that inputs variables belonging to the dont-care set are dropped during decomposition (please refer to Algorithm 1), (*iii*) the regularity of the implication table that allows large minimization (see Algorithm 2).

From a complexity viewpoint, the barchart in Fig. 6 reports the average CPU execution time of the *Kanon* tool compared to the BBDD package. It turns out that MFPD synthesis is, on the average, 38x faster w.r.t. the procedures used for BBDDs. As a representative example, for the largest benchmark (`s13207.1`) the MFPD is built and optimized in 7.08 s, whereas the equivalent

BBDD takes 241.9 s. This is due to the lower computational workloads of MFPD algorithms which do not require the reconstruction of the network graph during optimization.

5.3 Many Technologies, One Synthesis Methodology

To demonstrate the "orthogonality" of both the MFPD model and the MGPL style over different technologies, we mapped the benchmarks under analysis using three different devices: graphene p-n junctions (*Graphene*), Ambipolar Silicon NanoWires (Si-NWs) Pass-Transistors (*Si-NW PT*), and traditional MOSFET-based Pass-Transistors (*Si-MOS PT*). As described in Sect. 2.1, each of these technologies has different "optimal" primitives, namely the ones with the highest expressive power. Hence, tools that can seamlessly use different logic primitives may represent a genuine solution for the evaluation and comparison of different emerging technologies.

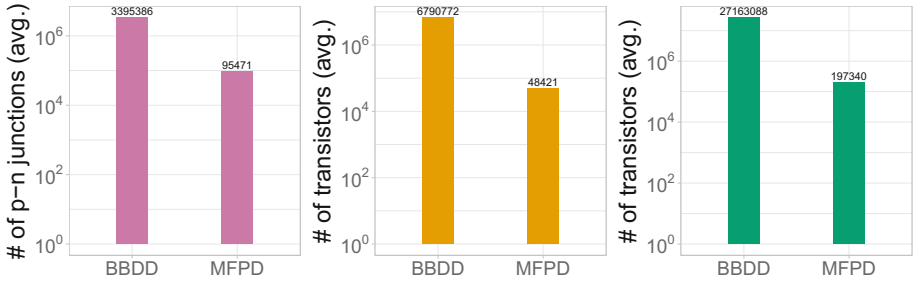


Fig. 7. Average device count after synthesis and mapping. Graphene (left), Ambipolar Si-NWs (center), MOSFET PTL (right).

Figure 7 summarizes the post-synthesis results obtained using our tool. Since MFPDs are a superclass of Pass Diagrams, we only provide comparison against BBDD-based synthesis. BBDDs represent the most recent solution proposed for emerging technologies [21] and their superiority w.r.t. other solutions have been already demonstrated. Notice that MFPDs nodes can be mapped with NOR, XOR and XNOR, while BBDDs only allow XOR mapping. Each of this pass-gates count different devices according to the adopted technology (Fig. 7). As a result of the multi-function decomposition, circuits synthesized using MFPDs are several orders of magnitude smaller in size; this translates into more area and more power efficient MGPL circuits.

5.4 Power and Performance Efficiency of MGPL Circuits

The extremely compact structure of MGPL circuits allows very high power/energy reduction. While this is an intuitive conclusion, here we underline

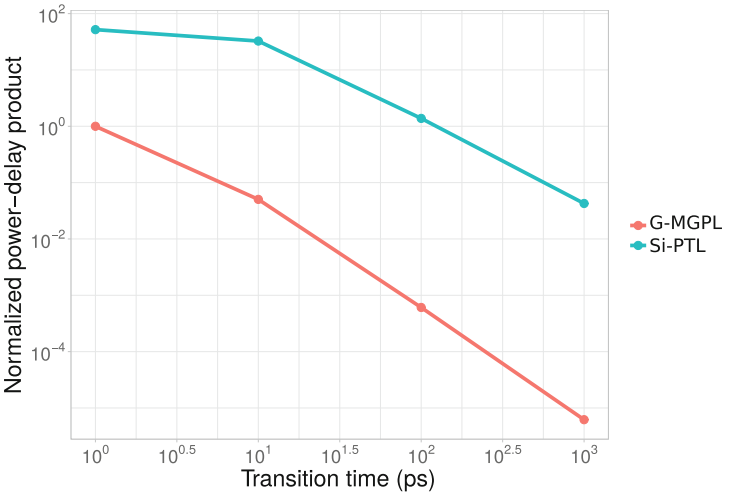


Fig. 8. Normalized PDP vs transition time.

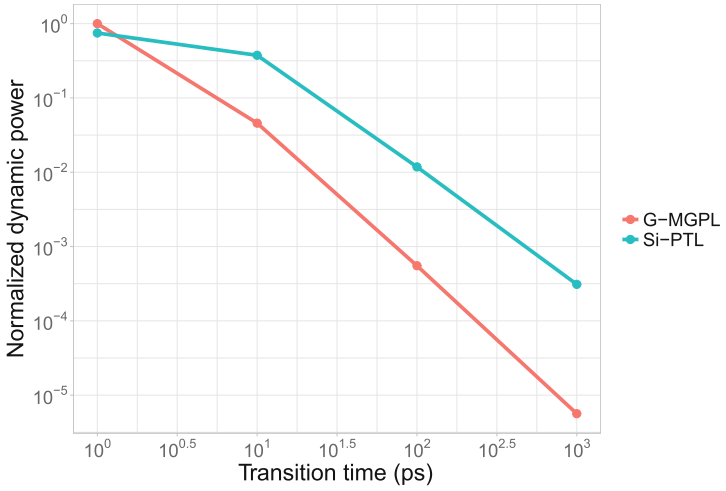


Fig. 9. Normalized dynamic power vs. transition time.

the energy efficiency of the MGPL style for emerging technologies, Graphene in particular. Figure 8 provides a technological comparison between Graphene-based MGPL circuits and Silicon-MOS PTL circuits. The plot shows the power-delay product (PDP) averaged over all the benchmarks as function of the transition time T_r of the input signals. The plot highlights the "adiabatic" nature of both implementations, i.e., PDP reduces as T_r increases. For a T_r that ranges from 1 to 1000 ps (3 orders of magnitude), the PDP of graphene reduces by more than 5 orders of magnitude, whereas that of silicon reduces only by 3 orders of

magnitude. However, and this is the most important aspect, graphene circuits are more energy efficient, not just in terms of absolute numbers, a result due to the intrinsic characteristics of the material [24], but also in terms of "scalability". This concept is further explained in Fig. 9 that correlates dynamic power consumption over transition time. As the plot suggests, for $T_r \cong 1\,\text{ps}$, namely outside the adiabatic region, PTL circuits are more power efficient, but as T_r increases, the adiabatic nature of the MGPL circuits shows reaching a power consumption that is about 2 order of magnitude lower that the PTL counterpart (best case at $T_r \cong 1\,\text{ns}$).

6 Conclusions

In this work we introduced a novel abstract representation for Boolean switching functions: the MFPD. Such a new data-structure, obtained with a multi-function logic decomposition, allows the implementation of compact MGPL circuits that well fit the characteristics of new emerging devices.

Results obtained with our tool *Kanon* demonstrate that MGPL circuits show superior characteristics w.r.t. state-of-art solutions, in particular *(i)* larger area efficiency (almost 15.51x better than PTL circuits obtained with BDDs) and *(ii)* shallower logic paths (77% w.r.t. CMOS circuits obtained with AIG-based multi-level synthesis).

These achievements demonstrate that there is a huge margin of improvement when moving to new technologies and that solutions universally recognized as a de facto standard for CMOS may fail when considering devices with different characteristics.

References

1. Schulz, M.: The end of the road for silicon? Nature **399**(6738), 729–730 (1999)
2. Thompson, S.E., Parthasarathy, S.: Moore's law: the future of SI microelectronics. Mater. Today **9**(6), 20–25 (2006)
3. Bernstein, K., Frank, D.J., Gattiker, A.E., Haensch, W., Ji, B.L., Nassif, S.R., Nowak, E.J., Pearson, D.J., Rohrer, N.J.: High-performance CMOS variability in the 65-nm regime and beyond. IBM J. Res. Dev. **50**(45), 433–449 (2006)
4. Zimmermann, R., Fichtner, W.: Low-power logic styles: CMOS versus pass-transistor logic. IEEE J. Solid-State Circ. **32**(7), 1079–1090 (1997)
5. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)
6. Rabaey, J.M., Pedram, M.: Low Power Design Methodologies, vol. 336. Springer Science & Business Media, Boston (2012). doi:10.1007/978-1-4615-2307-9
7. De Marchi, M., Sacchetto, D., Frache, S., Zhang, J., Gaillardon, P., Leblebici, Y., De Micheli, G.: Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire fets. In: IEDM 2012: International Electron Devices Meeting, pp. 8.4.1–8.4.4, December 2012

8. Chiu, H.-Y., Perebeinos, V., Lin, Y.-M., Avouris, P.: Controllable PN junction formation in monolayer graphene using electrostatic substrate engineering. Nano Lett. **10**(11), 4634–4639 (2010)
9. Han, M.Y., Özyilmaz, B., Zhang, Y., Kim, P.: Energy band-gap engineering of graphene nanoribbons. Phys. Rev. Lett. **98**(20), 206805 (2007)
10. Ikeda, S., Hayakawa, J., Lee, Y.M., Matsukura, F., Ohno, Y., Hanyu, T., Ohno, H.: Magnetic tunnel junctions for spintronic memories and beyond. IEEE Trans. Electron Devices **54**(5), 991–1002 (2007)
11. Parkin, S.S., Hayashi, M., Thomas, L.: Magnetic domain-wall racetrack memory. Science **320**(5873), 190–194 (2008)
12. Shelar, R.S., Sapatnekar, S.S.: Bdd decomposition for delay oriented pass transistor logic synthesis. IEEE Trans. VLSI Syst. **13**(8), 957–970 (2005)
13. Oklobdzija, V.G., et al.: Pass-transistor adiabatic logic using single power-clock supply. IEEE Trans. Circuits Syst. II **44**(10), 842–846 (1997)
14. Houri, S., Billiot, G., Belleville, M., Valentian, A., Fanet, H.: Limits of CMOS technology and interest of nems relays for adiabatic logic applications. IEEE Trans. Circuits Syst. I **62**(6), 1546–1554 (2015)
15. Ding, L., Zhang, Z., Liang, S., Pei, T., Wang, S., Li, Y., Zhou, W., Liu, J., Peng, L.-M.: CMOS-based carbon nanotube pass-transistor logic integrated circuits. Nat. Commun. **3**, 677 (2012)
16. Miryala, S., Calimera, A., Macii, E., Poncino, M.: Ultra low-power computation via graphene-based adiabatic logic gates. In: DSD 2014: Digital System Design Conference, pp. 365–371 (2014)
17. Suzuki, M., Ohkubo, N., Shinbo, T., Yamanaka, T., Shimizu, A., Sasaki, K., Nakagome, Y.: A 1.5-ns 32-b CMOS alu in double pass-transistor logic. IEEE J. Solid-State Circuits **28**(11), 1145–1151 (1993)
18. Lee, J.D., Yoon, Y.J., Lee, K.H., Park, B.-G.: Application of dynamic pass-transistor logic to an 8-bit multiplier. J. Korean Phys. Soc. **38**(3), 220–223 (2001)
19. Wu, T.-Y., Lu, L.-Y., Liang, C.-H.: Low-leakage and low-power implementation of high-speed 65nm logic gates. In: Electron Devices and Solid-State Circuits Conference. EDSSC 2008, pp. 1–4. IEEE (2008)
20. Bertacco, V., Minato, S., Verplaetse, P., Benini, L., De Micheli, G.: Decision diagrams and pass transistor logic synthesis. In: International Workshop on Logic Synthesis, vol. 168 (1997)
21. Amaru, L., et al.: Biconditional binary decision diagrams: a novel canonical logic representation form. IEEE J. Emerg. Sel. Top.Circuits Syst. **4**(4), 487–500 (2014)
22. Drechsler, R., Günther, W.: Towards One-Pass Synthesis. Springer Science & Business Media, Heidelberg (2013)
23. Somenzi, F.: Cudd: Cu Decision Diagram Package Release 2.3.0. University of Colorado at Boulder, Boulder (1998)
24. Tenace, V., Calimera, A., Macii, E., Poncino, M.: One-pass logic synthesis for graphene-based Pass-XNOR logic circuits. In: Design Automation Conference. DAC 2015, pp. 1–6. ACM (2015)
25. Tenace, V., Calimera, A., Macii, E., Poncino, M.: Multi-function logic synthesis of silicon and beyond-silicon ultra-low power pass-gates circuits. In: International Conference on Very Large Scale Integration, VLSI-SoC 2016, pp. 1–6. September 2016
26. Tenace, V., Calimera, A., Macii, E., Poncino, M.: Pass-XNOR logic: a new logic style for PN junction based graphene circuits. In: Design, Automation and Test in Europe. DATE 2014, pp. 1–4. IEEE (2014)

27. Tenace, V., Calimera, A., Macii, E., Poncino, M.: Quasi-adiabatic logic arrays for silicon and beyond-silicon energy-efficient ICS. IEEE Trans. Circuits Syst. II Expr. Briefs **63**(12), 1111–1115 (2016)
28. Miryala, S., Tenace, V., Calimera, A., Macii, E., Poncino, M.: Ultra-low power circuits using graphene p-n junctions and adiabatic computing. Microprocess. Microsyst. **39**(8), 962–972 (2015)
29. Synthesis, B.L., Group, V.: Abc: a system for sequential synthesis and verification (2014). http://www.eecs.berkeley.edu/alanmi/abc/
30. Collection of digital design benchmarks (2015). http://goo.gl/6fOVfN

Digital Hardware Design Based on Metamodels and Model Transformations

Johannes Schreiner[1(✉)] and Wolfgang Ecker[2]

[1] Infineon Technologies AG, Neubiberg, Germany
johannes.schreiner@infineon.com
[2] Technische Universität München, Munich, Germany
wolfgang.ecker@tum.de

Abstract. This contribution presents a Model-driven Architecture (MDA) inspired strategy for the automation of digital hardware design starting at specification level and targeting RT-level. This strategy defines a structured approach with is superior to code generation using scripts, print statements or template engines directly targeting ASCII files.

As part of this strategy, we implemented intermediate models named Models-of-Things (MoTs) for formalizing specification data which have a dependency on the design objects specified. We further implemented a Model-of-Design (MoD) for hardware design related modeling, and a Model-of-View (MoV) for target view generation. For the transformations between our intermediate models, we use a template based approach. These templates are executed and generate more concrete models utilizing information from more abstract models. The term model here describes instances of Metamodels in the terminology of MDA and must not be mixed up with simulation and synthesis models such as VHDL RTL models.

The template which guides MoD generation is called Template-of-Design. On one hand, the Template-of-Design (ToD) captures the (micro-)architecture and on the other hand, it retrieves MoT data to automate creation of the design that meets the specification. The Template-of-Design is Python code – as all our framework is implemented in Python – and uses generated APIs to simplify (micro-)architecture construction. In contrast templates for classical template engines, the Template-of-Design generates a model and not an ASCII File.

To generate the final view, a so-called Template-of-View (ToV) is used. It is also encoded as Python code, traverses the Model-of-Design and creates the MoV. This Model-of-View is in many aspects similar to Abstract Syntax Trees utilized in language parsing. It is defined by a Metamodel, which is generated from our so-called View Language Description (VLD). This language is also used to generate an un-parse method, which automates the view generation task from MoV instances. This means that provided our VLD and a Model-of-View instance, we can generate an ASCII-File containing e.g. RTL VHDL code in a completely automated way. Our VLD format also includes formatting pragmas to guide view generation tasks such as indentation and alignment.

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 83–107, 2017.
DOI: 10.1007/978-3-319-67104-8_5

Our strategy is supported by type and expression Metamodels that are used across different Models on different modeling layers (i.e. in MoTs or MoDs as well as in the Template-of-Design). This further simplifies implementation of the models as well as of the templates.

First analysis of our approach shows that we can simplify building one generator up to a factor of 10×. This factor increases further when different target languages or target language styles are generated from one MoD.

Keywords: Model-Driven Architecture · Hardware generation · Metamodeling

1 Introduction

In the 2013 McKinsey study on semiconductors [4], Collet et al. analyze how the *design gap* widened by a factor of about 5× in one decade. In this period, the number of transistors that can be manufactured increased by a factor of 100×, yet the productivity of design increased only by a factor of 20×. McKinsey concluded that closing the *design gap* is essential particularly for semiconductor companies fully or partially following the fabless approach.

The last disruptive productivity increases have been achieved by semi-custom design, RTL synthesis, and so-called "IP-reuse". Besides automating the construction of the lower level implementation, these approaches heavily rely on reusing pre-implemented pieces[1]. Although each of these approaches brought a big leap in productivity, the general benefit of reuse is often overestimated – an insight acquired in [4] as well. One reason for why TLM has not brought productivity increases may be that it does not define one agreed abstraction as RTL (synchronous design, time discretization) and Gate-Level (digital design, value discretization) do.

A measure proposed to further increase design productivity is the generation of code from more abstract descriptions. Ecker et al. reported a 20× productivity increase in special design tasks and up to 3× higher productivity in design implementation from specification freeze to tape-out through the use of Metamodeling and code generation [6]. The replacement of ad-hoc script based approaches by a more structured approach and the link to formalized specification data have been mentioned as key to this improvement.

Generation as part of a completely new approach of designing chips has been postulated by Nicolic [17] and Shacham et al. [20]. Generators should not only be used to make simpler and more efficient designs but also to generate different alternatives, thus to enable exhaustive architecture analysis. The use of generators instead of models has been claimed by Bachrach et al. [3] as well. Bachrach is developing Chisel, a so-called hardware generation language (HGL). He has

[1] Please note that beyond functional mapping and time abstraction to clock cycles, a big contribution in RTL's productivity results from mapping operators such as "+" to predefined structures such as adders.

demonstrated a code reduction by a factor of 3× in several application examples, when comparing Chisel models and Verilog code. While Chisel provides a very compact and generation centric RTL notation, micro-architecture pattern [13] go a step further and generate via an intermediate called DHDL complex pipelined micro-architecture patterns.

Interestingly, IP-XACT [10], the standard for IP integration, was prepared from the beginning to include generators and generator chains in the integration process. Unfortunately, no measure was described to formulate and build generators efficiently.

Besides controlling and configuring generation, building generators is the most time-consuming effort to enable automation. With Model-driven Architecture (MDA) [18,22], the OMG offers a vision for target code generation from specification via a set of models and a set of transformations, each deriving a model from the adjacent model. For further reading, it is important to note that we refer to *model* in terms of SW engineering and Metamodeling. A *model* is an instance of a Metamodel and a Metamodel is a class diagram-like notation[2] of the model's structure. In other words, a model specifies a set of entities – or things – their relations and their attributes.

Another challenge in making generators – or as part hereof translators – is the consideration of different Models-of-Computation (for a definition, see e.g. [7]). The generator developer does not only have to think in terms of generated design target but also in terms of simulation artifacts of the target languages that are eventually synthesized to hardware. Examples for simulation artifacts are blocking and non-blocking statements in Verilog, delta races caused by clock gates in VHDL, and proper `'X'` handling. Further examples are the type inconsistency between std_logic_vector and std_ulogic_vector in VHDL and additional code artifacts that are necessary as it is prohibited to read output ports.

In this contribution, we present an approach that follows the MDA vision postulated by the Object Management Group (OMG). MDA describes code generation via a chain of models in order to simplify and improve the construction of generators. In the next chapter, we describe the MDA idea in more detail and explain how we adopt it to digital hardware design. After that, we describe the components of our framework and the individual models and metamodels. Here, we provide simple examples for transformations and application sketches. First, we describe the Model-of-Things a specification related and design independent layer. Next, we introduce MetaRTL, a (digital hardware) Model-of-Design (MoD), acting as key model of our proposal. This MoD not only simplifies and accelerates generator construction by providing a more abstract target. It also helps to avoid thinking in terms of simulation semantics and artifacts. Here, we also show how MoDs can be efficiently built using a Templates-of-Design (ToD). To make transformation to RTL code complete in order to be able to utilize state-of-the-art design flows, we introduce as a third model the Model-of-View (MoV) an abstract syntax tree like structure. The application of the

[2] There are other notations as e.g. XML schema definitions or entity relationship diagrams. For an overview see e.g. [5].

proposed methodology for generating different flavours and styles of code for different processors and the benefits of the approach are discussed next. After introducing the general framework and our overall approach, we describe real-world application examples and use them to compare our hardware generator construction method to related generation and modeling approaches. Eventually, we describe related work in the field of hardware generation and construction approaches and evaluate how our approach competes.

2 Model-Driven Architecture

2.1 The OMG Model-Driven Architecture Vision

Model-driven Architecture (MDA, [16,18,22]) is an OMG vision of the future of software design that popped up a bit more than 15 years ago. MDA approaches a growing productivity gap that also burdens software engineering by fostering code generation. Instead of using simple model-to-code generation, MDA involves some intermediate steps before the final code is generated.

The first version of MDA introduces three kind of models, namely a Computation Independent Model (CIM), a Platform Independent Model (PIM) and a Platform Specific Model (PSM), where:

CIM is the most abstract one and closest to specification. It considers neither detailed algorithm implementation nor architecture.

PIM already defines the architecture – and therefore also the implementation but avoids details of the platform.

PSM is platform dependent and closest to the target code, also referred to as view. From this model, the view is generated.

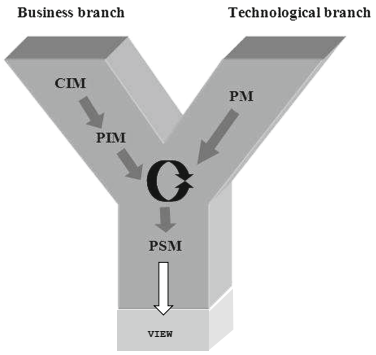


Fig. 1. MDA pictured as Y-Chart from [14]

Figure 1 shows this dependency in a Y-Chart for which MDA is well known. In addition to the aforementioned models, this figure also shows a Platform Model (PM), which contains the details of the target platform.

A remaining question is "what is a platform?". In the software world, a platform is a computing infrastructure on which generated target code can run. A platform for example defines libraries, APIs and the OS the generated view code compiles and executes on.

Before we describe our adaptation of MDA to digital hardware, we will first analyze some limitations of the initial MDA definition. First, the definitions of CIM and PIM are a bit vague, especially when it comes to the definition of functionality. Second, the agreement on 3 intermediate models is often at issue and it is questionable why there aren't 1, 2 or 4 layers of models instead, depending on the application domain and the difficulty of transformations. The third and last issue relates to the position of the Platform Model in the Y-Chart. This model is a description of the platform a PIM is mapped onto. The level of abstraction of this description is thus comparable to that of the PSM and it is depicted too "high" in the Y-Chart, since it is closer to the PSM's level of abstraction.

The problem with the modeling layers is relaxed in the second version of MDA released mid 2014 [18]. Here, a characteristic of a transformation is that it translates a PIM to a PSM. A sequence of any number of transformations is chained together to finally generate the view. At each connection point, the PSM becomes the PIM of the next transformation. A platform is therefore no longer simply related to a computation platform. Instead, for each platform an automated construction path – potentially via several intermediate steps – to the target code is provided. The target code is then compiled together with packages and libraries, etc. to a product, suitable to run on a computing platform[3].

2.2 Model-Driven Architecture for Hardware Development

Figure 2 shows our approach to adopt MDA for digital hardware generation. It conceptually follows the 3-level model of the first MDA proposal but has important enhancements to support hardware design. For instance, it introduces terms for the various hardware related models that are involved, each being more closely related to the purpose of the model they describe:

MoT. The Model-of-Things corresponds to the CIM since its intention is to formally capture data from requirements and specifications. By doing so, the MoT defines things, their attributes and the relations to the intended functionality. In this context, functionality describes what the product has to provide, without including how the product is implemented, e.g. which algorithms and architecture/structure are used. An example for a MoT is the instruction set of a CPU as discussed in Sect. 7.1.

[3] Please note that the term "platform" is used in different ways in the hardware design area: Keutzer et al. defined a hardware platform as a family of microarchitectures that allow sustainable reuse of SW in [12]. In other contexts, the term platform is used for a product family which addresses one field of application in various configurations.

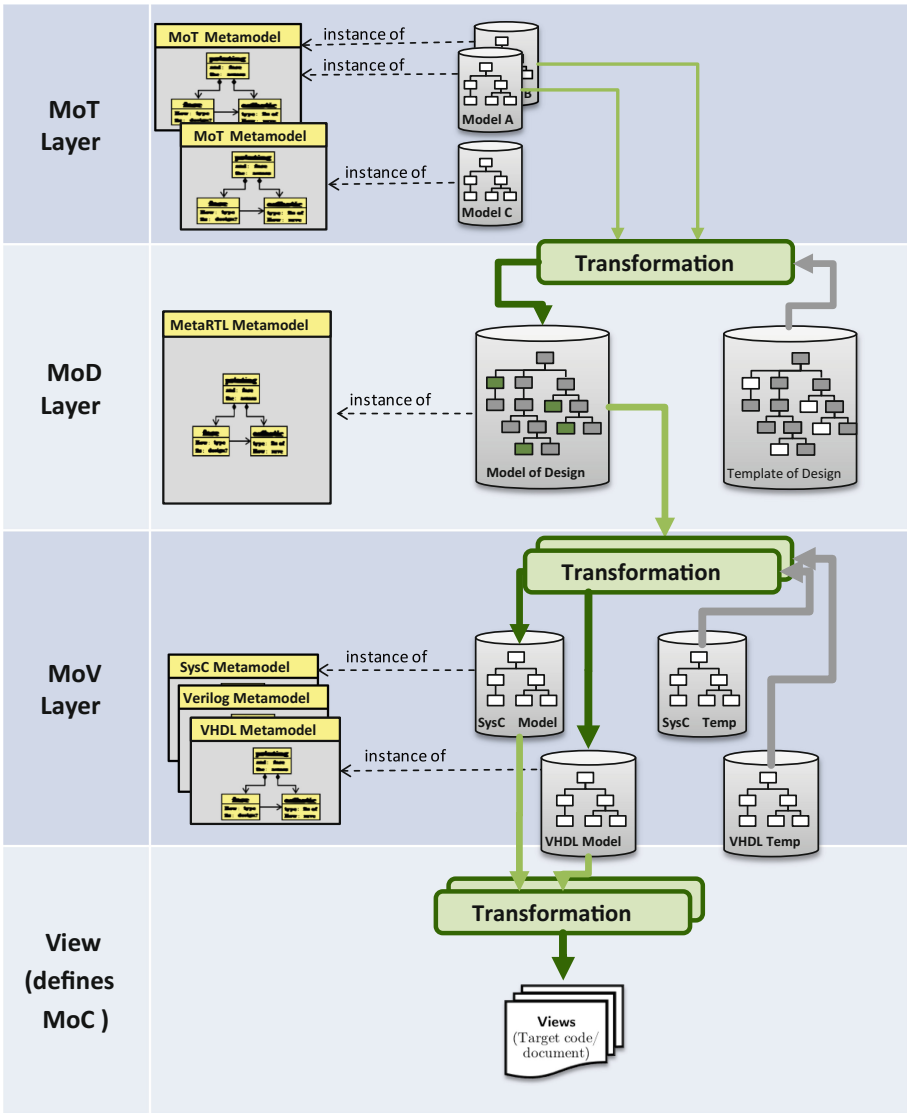


Fig. 2. MDA for hardware generation

MoD. The Model-of-Design corresponds to the PIM since its goal is to define the architecture using designer's terms. The MoD and its Metamodel are the core components of our methodology. Broadly speaking, a Model-of-Design describes the design on RT-level. It differs from modeling languages based on event-driven semantics as it does not cover any simulation or synthesis artifacts of the views we target with our approach. We detail the MoD and its Metamodel in Sect. 3.

MoV. The Model-of-View corresponds to the PSM since it is the least abstract model with a straightforward mapping to the target view. It is more or less an abstract syntax tree defined by an abstraction of the target code's grammar. In future, we might extend this model with some target architecture related information as mapping pragmas or target platform specific code snippets. The Model-of-View layer is detailed in Sect. 6.

For each of the MDA-layers, we define Metamodels. These metamodels constrain the valid models of the modeling layers. There are several Metamodels on the MoT layer and on the MoV layer respectively. For the MoD layer, there is exactly one metamodel which is related to the RTL and synchronous design abstraction. For targeting other implementations such as analog hardware or firmware, additional Models-of-Design are needed. Every specification formalized in a Model-of-Things instance is transformed onto a set of RTL components part of a Model-of-Design. On the Model-of-Things layer, several different metamodels are necessary depending on the design task. When generating several CPU cores with different RISC ISAs, the formalized description of these ISAs will use models of the same Metamodel. When a full CPU subsystem is generated, various peripherals such as timer, interrupt controller or signal processing peripherals will require different Metamodels. On the Model-of-View layer, a different Metamodel is necessary for every view language, while two different views of the same language use the same Metamodel.

In addition, Fig. 2 introduces the Templates-of-Design (ToDs, see Sect. 5.2) and Templates-of-View (ToVs, Sect. 6). They relate to the Platform Model (PM) in MDA's Y-Chart representation from Fig. 1. They are pictured right of the MoD and MoV respectively, illustrating that they are on the same level of abstraction. These templates are designed for a certain Metamodel on the respective layer of abstraction and therefore work on the API provided for that Metamodel. The Template-of-Design (ToD) describes the targeted architecture and directly instantiates Model-of-Design (MoD) items. For this instantiation, it can utilize data from several Model-of-Things (MoT) instances. It is thus possible to intuitively integrate configurable and non-configurable components. The result of the Template-of-Design execution is one static Model-of-Design instance. This instance can be transformed on the Model-of-Design layer and can further be transformed into a Model-of-View instance using the Template-of-View (ToV). Similar to the ToD, the ToV relates to the structure of the target code. It instantiates elements of the MoV, which are more-or-less nodes of the abstract syntax tree of the target view.

Similar to MDA 2.0, our adoption of MDA to hardware does not insist on exactly three modeling layers. Instead, the MoV can be omitted and the view can be directly generated with a template engine from the MoD. Further, several MoTs can be used to build the MoD and there may be transformations between MoTs before the MoD is constructed.

Our adoption of MDA realizes the two key aspects for productivity improvement already mentioned in the introduction: automation and reuse. Automation is established by transformations and reuse is enabled by re-using models, their

Metamodel-based definitions, and through reuse of existing transformations and view generators.

3 Framework

This section introduces the framework we utilize for our Model-driven Architecture approach. For this purpose, we first introduce Infineon's proprietary metamodeling framework. We then show auxiliary elements which are utilized across different Metamodels.

3.1 The Overall Framework

Our metamodeling framework uses a subset of an UML class diagrams, extended with some features from XML Schema to specify Metamodels. Substantially, objects, their attributes and various kinds of relations between them can be specified. The Metamodel in turn is defined by a Meta-metamodel, which is also used to relate and combine Metamodels as well as to generate Metamodels from other descriptions like an XML Schema or EBNF grammars. The Meta-metamodel uses the same notation as the Metamodel, i.e. is self-defined.

The Metamodel can be captured in a textural and a graphical way. The framework generates an infrastructure for handling the Metamodel and its models. This infrastructure supports the generation of extendable APIs, code generation for persistent storage of the models and tool frame generation. The generated tool frame supports joint handling of various models, creation of models, transformations between the models, and generation of views. The framework is written in Python, uses Python for manually coded and generated transformations and takes advantage of numerous Python libraries and tools including the Mako template engine for a direct Model to View generation.

Python is not only used because of the already mentioned libraries and rich features such as object-oriented programming, aspect-oriented programming utilizing introspection and functional programming. Equally important is its low adoption barrier. We observed that Metamodeling frameworks such as the "Eclipse Modeling Framework" (see [21]) are more powerful but very complex and thus very hard to learn and adopt for hardware engineers. With a Python-based approach, we managed to lower this barrier significantly.

The framework makes the development of MDA flows easy by generating the already mentioned tools. Both Template-of-Design (ToD) and Template-of-View (ToV) are developed as Python code utilizing the generated API to fill their target model and to access the more abstract source models. The models reside as data structures in the generated framework tools. These data structures can be read from and written to XML files.

The uniform structure of the modeling flow also makes it easy to understand and use the Metamodels. Thus, the different levels of detail resulting from different levels of abstraction from MoT to MoV can be easily handled. Descriptions

can be made in a uniform way, transformations are less painful and there are fewer inconsistencies.

Although the Metamodel enables the described tool building framework, Metamodeling is still mainly a modeling activity since it structures and defines the objects, attributes and relations as abstractions of the design space.

3.2 Auxiliary Metamodels

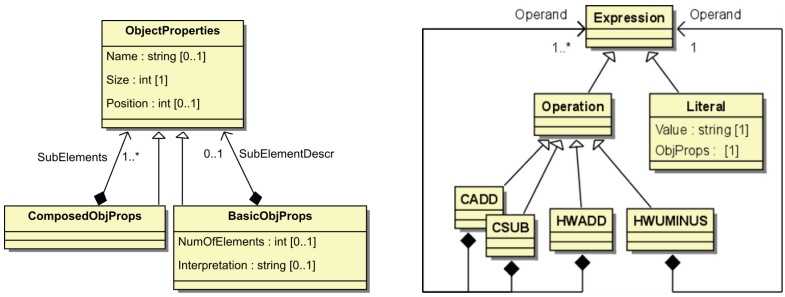
The above-mentioned Meta-metamodel-based features of the framework make it easy to structure Metamodels and to reuse predefined Metamodels. We currently support two auxiliary Metamodels which are used on different layers of our MDA approach: the Type Metamodel and the Expression Metamodel.

These two Metamodels are used on the Model-of-Things and the Model-of-Design layer. They are intended for use wherever Metamodels contain elements describing objects and behavior. It is for example necessary to describe object sizes (e.g. the sizes of registers) on both the Model-of-Things layer and the Model-of-Design layer. The utilization of the same auxiliary descriptions across different metamodels has two key advantages. First, it reduces the development effort for new metamodels and improves overall code quality and usability. Second, it simplifies the extraction of information necessary for the transformation between different modeling layers. In many cases, objects and expressions can be simply compared and copied between different MoD and MoT instances.

The Type Metamodel follows the idea describing data types and interpretation using the underlying semantics of hardware wires. Therefore, it supports the specification of any kind of bundles of wires together with some optional hardware properties such as signed interpretation and Endianness. Types can be named to ease readability of the generated targets and to simplify referencing types in specifications. Type compatibility is however not dependent on type names or other hardware properties. Instead, type consistency is only dependent on the size (i.e. the number of bits or hardware wires) of any two types. Figure 3a shows a UML representation of our type Metamodel.

The Expression Metamodel defines a tree of operators, with both classical operators from programming and specific hardware operators such as add-with-overflow or select. Figure 3b contains a UML representation of a subset of our Expression Metamodel. The expression hierarchy is defined by the location in the tree. There is no defined operator priority or counterpart to brackets for prioritizing operators. For each operator, the size of the output depends on the size of the inputs. Further, relations between the inputs are specified. There is however no size limit as a such. Leaf nodes can be binary or numeric literals in any combination or references to objects with properties defined by the type model. The Expression Model is a special case of a Dataflow Model. The Dataflow Model is a directed graph with the same operators as the Expression Model. It is different from the Expression Model as any operator output can act as input to multiple dataflow operators.

These two Metamodels are used on the Model-of-Things and Model-of-Design layer. They are intended to be used whenever Metamodels contain elements



(a) Type Metamodel (b) Expression Metamodel

Fig. 3. Auxiliary metamodels

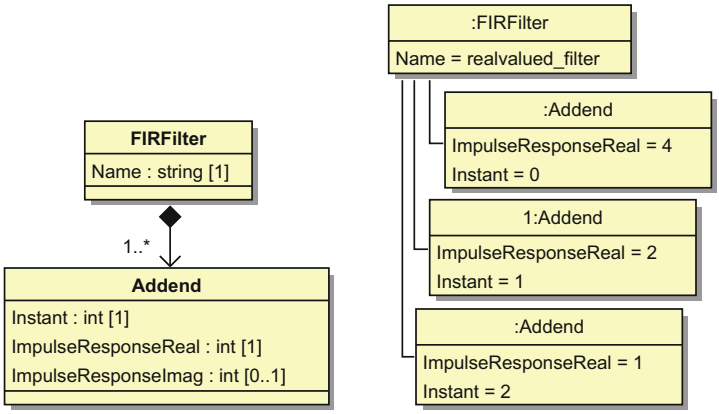
describing objects and behavior. It is for example necessary to describe object sizes (e.g. the sizes of registers) on both the Model-of-Things layer and the Model-of-Design layer. The utilization of the same auxiliary descriptions across different Metamodels has three key advantages. First, it reduces the development effort for new Metamodels and improves overall code quality and usability. Second, it simplifies the extraction of information necessary for the transformation between different modeling layers. In many cases, objects and expressions can be simply compared and copied between different MoD and MoT instances. Third, the mapping of types and expressions to the target view must be implemented only once and can be reused for all implementations. Latter holds for both possibilities for view generation, the direct code generation from MoD using a template or having a MoV in between. The first alternative produces quite rapidly results but becomes complex if some flexibility or formatting is needed in code generation. The second alternative requires thinking in terms of grammar blocks but keeps the burden of formatting to a one time effort and makes it simpler to make code generation more flexible.

4 The Model-of-Things Layer

To ease understanding of the Model-of-Things concept, this section provides a simple sample Model-of-Things along with its Metamodel. We utilize this example to illustrate how the Template-of-Design generates a Model-of-Design instance in Sect. 5.2. A more complex example is sketched in Sect. 7 of this contribution.

The Metamodel in Fig. 4a constrains all digital filters. Our Model-of-Things instances can thus capture digital filters which adhere to a recurrence relation $y[n] = \sum_{i=0}^{N} b_i \cdot x[n-i]$. Figure 4b shows a sample model that adheres to this Metamodel. The attributes `ImpulseResponseImg` does not show up in the model since it is not needed. This complies with the Metamodel since `ImpulseResponseImg` has multiplicity optional, i.e. `0..1`.

This model describes one 2nd order instance a filter that has the recurrence relation $y[n] = 4 \cdot x[n] + 2 \cdot x[n-1] + 1 \cdot x[n-2]$.



(a) Metamodel (b) Model

Fig. 4. Model-of-Things metamodel and sample model of simple FIR filter

Despite the simple structure of this example, a key characteristic of our approach is visible here: The Model-of-Things contains only the specification. It is easy to imagine a set of microarchitectures to implement such a filter. For example, a simple one sample per cycle pipeline might be used (see Fig. 7). In this case, the multipliers might be replaced by wires treating the coefficients as constants for the cost of loosing flexibility. Generally, the added chain can be replaced by an adder tree. Another microarchitecture would consist of an FSM and a multiply-accumulate unit. While these microarchitectures have different characteristics, they both equally valid microarchitectures considering a specification that only describes the transfer function of the digital filter and does not contain any performance constraints such as throughput.

As this example illustrates, there is not necessarily a one-on-one correspondence between elements in the Model-of-Things and elements in the Model-of-Design. As we utilize a flexible Python-based transformation approach to generate the Model-of-Design, it is feasible to first derive the coefficients b_i for a digital filter. It is for example possible to use a Model-of-Things that specifies intended frequency-domain characteristics and maximum deviation from these characteristics. The large set of scientific computing libraries available allow to perform the necessary computations as part of the transformation between Model-of-Things and Model-of-Design, without requiring external tools or manual work. To be more concrete, Python's `scipy` provides a package `signal` which supports the computation of filter coefficients using the functions `firwin` and `firwin2`.

5 The Model-of-Design Layer

5.1 The MetaRTL Metamodel

Figure 5 shows MetaRTL, the Metamodel of the Model-of-Design (MoD). It defines, how the design structure is described in the MoD. MetaRTL consists

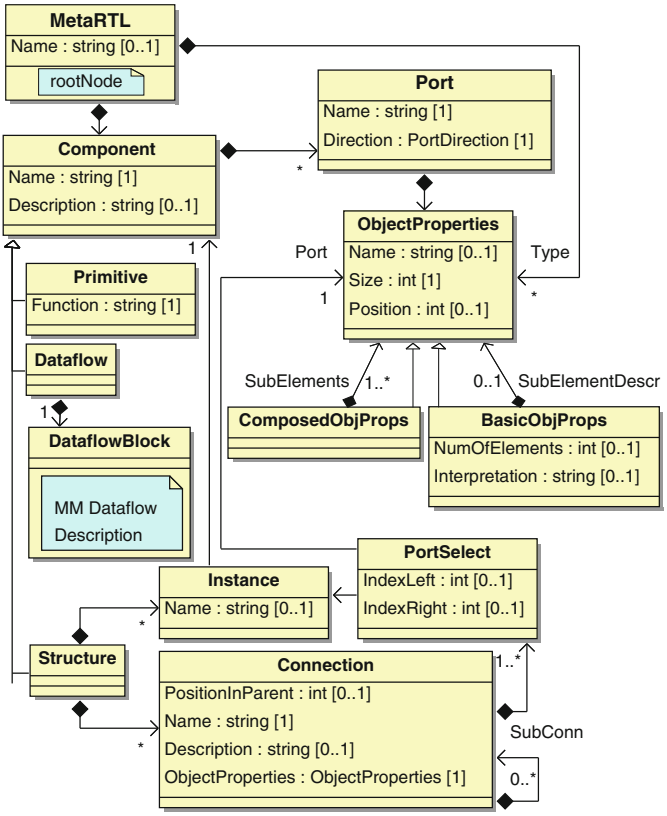


Fig. 5. MetaRTL: MoD's metamodel (simplified)

of a root node and components with realization alternatives: Structure, Primitive, Dataflow as well as – in preparation not shown in Fig. 5 – Controlflow, StateMachine, and Lookup-Table.

The last four of these are configured with other Metamodels describing their implementation. For example, the Dataflow realization of a component is configured by an instance of the Dataflow Metamodel that is linked in the DataflowBlock. The four models can be clocked, un-clocked, with or without internal delay. If they are clocked or have an internal delay, a clock and reset is associated automatically.

The Primitive and Structure realization alternatives are more flexible. They support the working mode of concept engineers and architects who often avoid the specification of port names or sizes, if they can be derived from elsewhere, e.g. from a sketch, or are common sense (as e.g. ports of a register are named q and d whereas q and d must have the same size).

There are special kind of ports like clock or reset. These ports are automatically connected to the port closest in the hierarchy, and labeled the same special

kind of ports. Of course, all connections and ports can be made explicitly and with accurately specified sizes. Then the connectivity resolution only checks the type – i.e. size – consistency of the connections.

Further, the connectivity specification supported by the Metamodel can not only connect a port with another port but can also connect a port with a component and can connect a component with a component. The connection can be associated with a set of properties that help the resolution mechanism to resolve the intended port or the size of port and connection.

Also setting of component parameters can be resolved from the connectivity. Thus, setting a connection to a reset port can determine whether a register has a reset behavior or not. Of course, all parameters can be also set explicitly in the instantiation. The resolution strategy is described in more detail in Sect. 5.4. Primitives such as registers, adders, multiplexers, decodes, etc. have methods that determine the final parameter setting and port layout.

To give some examples, in the register primitive, the size of q and d can be propagated forward and backward, i.e. the size of the data output q can be derived from data input d and vice versa. Connections to q and d can be made explicitly or distinguished by read/write or in/out property of the connection. Also a name, let's say a, can be specified. Then the register has slightly different ports, namely q_a and d_a in this case. Using this mechanism, a register may be configured to have more than one data input and output pair.

Propagation of sizes is limited when instantiating e.g. an adder. Here, the size of the output is the maximum of the size of the two inputs plus one. Thus, only a forward propagation of the sizes is possible.

Any number of connections can be applied to the output of the decoder. Then, a port is created for each of the connections. Further, each connection can have a property specifying at which select value it is enabled. This property, which can be also computed automatically, is used further on to define the behavior of the decoder and the size of the select signal.

5.2 Templates-of-Design

The Template-of-Design (ToD) acts as a blueprint or generator for instances of the Model-of-Design. It contains the instructions to build different instances of this model and thus different digital designs. It is called template since it pulls information from the more abstract MoT and since it looks almost like an HDL netlist description when a fixed architecture is described. In our implementation, the ToD is pure Python code and is not to be confused with the templates processed by template engines such as Python's Mako for target code generation.

The HDL netlist-like style (see Fig. 6) is enabled by object constructors and attribute setters and getters generated for the MetaRTL Metamodel. It is important to note that the Python code is not the hardware model. Instead, it constructs the hardware model. When Lines 2, 3 and 4 are executed, new components are instantiated in the Model. The loop in lines 7–9 creates a new connection every time it is run.

```
1 width = 32
2 alu_input_1 = Connection(ObjProps=SimpleObj(width), parent=alu)
3 alu_input_2 = Connection(ObjProps=SimpleObj(width), parent=alu)
4 alu_mux = Mux(Name='ALU output Mux', parent=alu)
5
6 for unit in alu.units:
7     alu_input_1.connect(unit.Ports['input_1'])
8     alu_input_2.connect(unit.Ports['input_2'])
9     unit.result_port.connect(target=alu_mux, ConnName=unit.name)
```

Fig. 6. Code snippet from a ToD using named method parameters

The power of the Python approach can be illustrated in following example. If a specific coding style for connections must be met, the parameter setting `ConnName=unit.name` can be modified with a small overhead e.g. to `ConnName=unit.name+"_s"`.

The code snippet is taken from a small ALU component which contains several functional units which process two input arguments (connected to `alu_input_1` and `alu_input_2`). For each functional unit (stored in `alu.units`), the code connects the output port to the `alu_mux` component (Line 9). Moreover, the input ports of the functional unit are wired to the connection objects `alu_input_1` and `alu_input_2` in Lines 7 and 8.

As the ToD is pure Python code, it provides the full flexibility of the Python environment for importing and versioning. If parts of different MoDs are similar, their ToDs can share the same code. This means that reuse is done on ToD-level and MetaRTL must not support the concept of a library of (reusable) components. This significantly reduces the complexity of MetaRTL and any transformations from its MoD-level to different MoVs.

5.3 Model-of-Things Instances for Template-of-Design Construction

In addition to describing circuits in the static HDL netlist way, the complete feature set of Python can be utilized. This provides more flexibility to define the architecture. Section 4 already named the utilization of Python-based scientific computing libraries to derive parameters for certain microarchitectures. In extreme cases, the Template-of-Design can even run instances of our complete MDA toolchain to find or evaluate solutions. Here, code is generated and analyzed in order to find out the right – or also the best – way to generate design items. We describe a scenario where we could successfully utilize this in Sect. 7.2.

To illustrate the concept without the complexity of real-world applications, this section shows a Template-of-Design which derives a digital filter from the Model-of-Things sketched in Sect. 4. The Template-of-Design is built for the Metamodel in Fig. 4b and will construct MoD instances for all models of this Metamodel. Figure 7 contains a block diagram of the circuit it creates for the sample model in Fig. 4a.

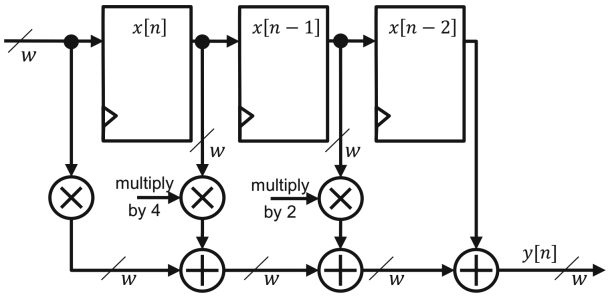


Fig. 7. Block diagram of MoD generated from the MoT in Fig. 4a by ToD in Fig. 8

```
1 class FIRFilter(Structure):
2   def __init__(self, firMoT, parent):
3     super(FIRFilter, self). \
4       __init__(Ports=[{'Name': 'data_in', 'Direction': 'IN'},
5                       {'Name': 'result', 'Direction': 'OUT'}],
6                parent=parent)
7     current_delay, sum_conn = 0, None
8
9     current_conn = Connection(self['data_in'])
10    for addend in sorted(firMoT.Addends,
11                         key=lambda x: x.Instant):
12      while current_delay < addend.Instant:
13        reg = Register()
14        current_conn.connect(reg.In)
15
16        current_conn = Connection(reg.Out)
17        current_delay += 1
18
19      mul = HWMUL(Constant=addend.ImpulseResponseReal)
20      mul.addIn().connect(current_conn)
21
22      instant_out = Connection(mul.Out)
23      if sum_conn is None:
24        sum_conn = instant_out
25      else:
26        adder = HWPLUS()
27        adder.addIn().connect(instant_out)
28        adder.addIn().connect(sum_conn)
29        sum_conn = Connection(adder.Out)
30
31    sum_conn.connect(self['result'])
32
33 fir_MoT_1 = ... # instance of a FIRFilter Model-of-Things
34 myFilter = FIRFilter(fir_MoT_1, toplevel)
```

Fig. 8. Template-of-Design example for generation of n-th order FIR filter from Model-of-Things

The Template-of-Design source code is pictured in Fig. 8. When executed, line 33 of this instance adds a `FIRFilter` instance to the Model-of-Design. The construction of this FIRFilter takes place in the constructor in Lines 3–31. First, the Ports of the filter are defined. We only define the ports `data_in` and `result` here and omit the clock and reset lines necessary for the microarchitecture. These are later inserted by a transformation on the Model-of-Design or by applying automatic connectivity resolution. In Line 9, every execution of the ToD instantiates a connection in the Model-of-Design. This connection is attached to the `data_in` port of the filter. Lines 10–29 contain the part of the filter that depends on the Model-of-Things. Here, the ToD iterates over all `Addend` instances of the Model-of-Things, sorted by their `Instant` attribute in an ascending order. Lines 12–17 contain a while loop including loop body which is executed to insert delay registers. In our sample MoT, we use values of consecutive instant n, $n-1$, $n-2$ for every output sample $y[n]$, the loop is thus executed once per iteration of the enclosing for loop. A multiplier is then inserted in line 19 and the connection referenced by `current_conn` is attached to the multiplier in line 20. In the first iteration of the for loop (line 10–29), `sum_conn` is then set to reference the connection object which is attached to the output of this multiplier. In every further iteration, an adder is inserted which sums up the connection previously referenced by `sum_conn` and the output of the multiplier. After this, `sum_conn` is redirected to point to a connection attached to the output of the adder. After all loop iterations are completed, line 31 attaches the `result` port of the FIRFilter component to the connection referenced by `sum_conn`.

It is important that the Template-of-Design is only one possible microarchitectural template. For the same Model-of-Things it would be feasible to develop a ToD that generates an adder-tree based microarchitecture or a multicycle filter using a multiply-accumulate unit.

5.4 Connectivity Resolution

The ToD can be kept simple, since our framework has a powerful resolution mechanism to do the final connection. Here, ports are created, connections of default ports, and size computation is done.

The mechanism is kept simple, since the instantiated components either utilize generic methods to compute component port information from other ports or provide special ones. The resolution algorithm calls these methods if new information about a connection has been computed and takes the new result of the method to refine information about connections. This is repeated as long as new information either about component's ports or connections can be found. At the end, either all components are fully configured – including availability of all ports – and all sizes are known, or an error message is filed summarizing the non determined items. More information must then be placed in the description of the ToD.

So far, resolution depends on component local implications at ports and connections. Potentially, the use of a formal SMT solver might resolve some more items. First analysis however showed that it is not worth the effort, since

the number of additional resolutions is small, complexity issues pop up soon and the reports aren't that easy to interpret.

6 The Model-of-View Layer

The Model-of-View layer is used to map the microarchitecture described in a Model-of-Design onto a target platform. Reasonable target platforms are HDL code for synthesis or simulation purposes. The Metamodel of the Model-of-View is tailored to the language of the target view. Two views both targeting Verilog code utilize the same Metamodel while a SystemC view would require a different Metamodel. When the same view language is used to generate code for different target platforms, the transformations are changed. A Verilog model for ASIC synthesis and a Verilog model targeting an FPGA is built from the Model-of-Design with two different transformations[4], while both MoV models utilize the same Metamodel. As those target view languages have their inherent Model-of-Computation (MoC), any Model-of-View also inherently defines the MoC.

The Metamodel of the Model-of-View layer specifies a target view similar to how Extended Backus-Naur Form (EBNF) notations describe the formal grammar of a language. The metamodel constrains the possible Model-of-View instances so that all legal instances will translate to grammatically correct views. Although there is a straightforward correspondence between the Model-of-View and the generated target views, the MoV-layer of our MDA inspired approach provides an important abstraction: it allows the developer to think about the view he wants to generate, without worrying about necessary formatting or indentation, as these are not part of any Model-of-View. Formatting and indentation are specified independent of the MoV and utilized to generate the necessary tools to generate the views from the Model-of-View. Consequently, it is also possible to alter these properties without touching the transformation process between any Model-of-Design and the Model-of-View or any transformations performed on an instance of the Model-of-View.

6.1 Automated View Generation from Model-of-View Instances

When developing the framework for our Model-of-View layer, we put special emphasis on automation. Figure 9 sketches this generation. We use a single EBNF-like description that contains information about the grammar of the target view as well as formatting and indentation of the generated code. This description is called View Language Description (VLD). It is utilized to generate the majority of the necessary MoV-layer components. First, the Metamodel

[4] Of course, the different transformations share several sub-transformations. However, we are analyzing, if two transformations or one transformation to the model of view is best, and if the one transformation uses parameter dependent code or if the one transformation is finished, and a further transformation is performed on the Model-of-View.

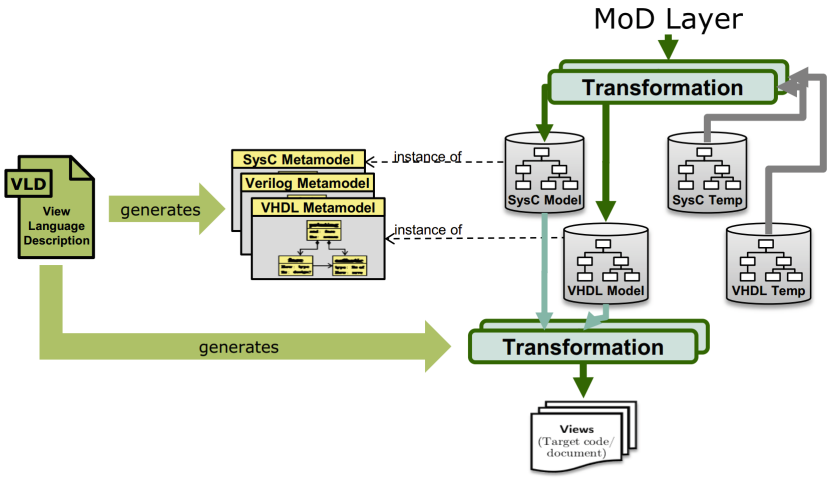


Fig. 9. View Language Description based metamodel-of-view generation

of the Model-of-View is built and based on this Metamodel, the metamodeling framework provides the API which used to read and write Model-of-View instances.

Furthermore, the code generation step, building the target views from any Model-of-View instance is completely automated. The general algorithm is based on tree-traversal, where leaf nodes produce view code that is encapsulated by view code produced by their parent nodes. Eventually, the view code generated for the root node, containing all code of its child nodes is printed to a view.

6.2 View Language Descriptions

Figure 10 contains a simplified example of a View Language Description (VLD). The similarity between this description and an Extended Backus-Naur Form (EBNF) description is apparent. Similar to EBNF, our description consists of a list of production rules where each of those rules in turn consists of a set of terminal or non-terminal symbols. The main goal of EBNF is to describe formal language grammars and thus to provide the rules for distinguishing grammatically correct code of a certain formal language from wrong code. As outlined in the previous section, our VLD has slightly extended goals: we utilize it to generate the Metamodel of the Model-of-View. This introduces three new requirements for our format.

1. While EBNF is mainly intended to describe the formal rules of a certain language, the VLD was designed to allow the automated generation of a concise Metamodel and thus an *intuitive* API. Here, it is not sufficient that all legal instances of this Metamodel form legal view instances. Instead, the Metamodel should be shaped so that it is easy to use for the developer. Artifacts

```
1 Entity ::= 'ENTITY ' <Name> ' IS\n'
2                [Ports]
3            'END ' <Name> ';\n';
4 Ports  ::= $indent('\t')$('PORT(\n'
5                $indent('\t')$(+Port%[0:-2]: ';\n';
6                                      [-1]  : '\n'  %+) ');\n');
7 Port   ::= <Name> ' : ' <Mode> <Type>;
```

Fig. 10. Simplified snippet from view language description of our VHDL MoV [19]

```
entity_declaration ::=
    ENTITY identifier IS
        entity_header
        entity_declarative_part
    [ BEGIN
        entity_statement_part ]
    END [ ENTITY ] [ entity_simple_name ] ;
```

Fig. 11. VHDL entity declaration from VHDL'93 (see `tams.informatik.uni-hamburg.devhdltoolsgrammarvhdl93-bnf.html`)

that are only grammatically relevant, however semantically irrelevant should not be part of the Metamodel. An example for such an artifact is the fact that in comma-separated lists, the last element is usually not followed by a comma. The Metamodel shall provide a list in this case and the generator shall treat the last element independent from the others.

2. While EBNF is structured to easily and efficiently build parsers, VLD shall provide a Meta-Model to construct code. So, an entity rule directly contains `ports` (see Fig. 10 in line 2) and `generics` as well as `library clause` and `use clause` (not shown in the simplified VLD example in Fig. 10). The VHDL grammar of the entity declaration shown in Fig. 11 is structured quite differently. So, `library clause` and `use clause` are defined in `context clause`, which is used in `design unit`. This makes it easy to reuse code in the parser, would require however that additional nodes have to be generated in the MoV, which reduces convenience of the approach.
3. It has to describe the indented formatting of the target view. The VLD thus describes exactly one correct target view belonging to each Model-of-View instance. In contrast EBNF grammars, do not specify things like whitespaces and indentation. From parser point of view, any number of identical views with different formatting will map onto the same Abstract Syntax Tree. From generation point of view, one formatting options shall be choosen when generating the code.

To meet those requirements, we introduce several formalisms into the VLD. When the Metamodel for a VLD format is generated, we add a Metamodel class for every production rule of the VLD descriptions. For every non-terminal symbol a rule consists of, we add associations to this class. An EBNF rule for a list of

at least one name would be similar to `names` ::= `firstName`, {′, ′, `otherNames`}. This clearly describes that there may be either one name or a list of comma-separated names. What it does not convey is that those items, `firstName` and `otherNames` belong together semantically. When generating a metamodel for that `names` rule, a `names` class would be inserted which would then contain two separate attributes one with the multiplicity `1` and the other one with a multiplicity of `0..*`. A developer working on the transformation from MoD to MoV would then have to take into account whether the `firstName` attribute is already set whenever he tries to add further names to the `nameList`. To avoid the accompanying overhead, we introduce a `+...+` symbol into our VLD. The use of this symbol will create one attribute of multiplicity `1..*`. As typical views still frequently require different generated view code for corner cases such as the first or last element in a list, we introduce a further artifact into our VLD. This artifact is used in the `Ports` rule of the VLD in Fig. 10. Here, the `%%` notation indicates that the last port (identified by [−1]) has to be treated differently from the others (identified by [0 : −2])) during view generation.

A further extension in our format is the distinction between non-terminal symbols that create attributes in the Metamodel (encapsulated by `<...>`) and symbols that create compositions (not encapsulated `<...>`). We further use the `:` operator to describe both attribute name and attribute type. When a rule contains e.g. <`Size` : `int`>, an attribute named `Size` of type integer will be added to the class generated for the rule.

A naïve approach to formatting target code in a nice way is inserting terminal strings containing whitespaces into the VLD. The problem of this approach is that it cannot handle indentation correctly as many production rules can occur at different levels of indentation: a concurrent signal assignment can e.g. take place at architecture level, inside a process or inside any number of nested conditionals, each requiring a different level of indentation. Our solution to this problem is the introduction of *formatting directives*. These directives are directly introduced into the view language description. They are implemented as Python code and work by post-processing code that has been generated. We provide a set of predefined directives for correct indentation of code, line breaks at certain line widths and correct alignment of neighboring lines.

To give an example: $`indent`(′\t′) in Fig. 10 in line 4 ensures an additional indent of keyword `PORT` and the following parts by a tabulator. Similarly, $`indent`(′\t′) in line 5 ensures, that each port item is indented one tabulator further.

7 Application and Results

To demonstrate the general feasibility and the advantages of our MDA approach to digital hardware design, we are implementing a CPU subsystem generator. The input models for our flow are Model-of-Things instances which are utilized to generate different aspects of the target views. Our generation system relies among other things on a model capturing the instruction set of the CPU core

part of the subsystem. The Metamodel of this model is called MetaRISC and suited to describe different RISC ISAs. We describe this Metamodel and how it is used by the CPU core generator in the following section. Our approach further relies on models that describe aspects other than the instruction set of the CPU. These models and their Metamodels cover peripherals such as the timer and interrupt controller and custom processing peripherals similar to the filter example in Sect. 4.

7.1 Model-of-Things of the CPU Core Defined by the MetaRISC Metamodel

MetaRISC is the Metamodel used to describe the ISA of CPUs our core generator can handle. The key requirement for identifying Metamodels for the MoT-layer is to formalize the possible MoT instances as much as necessary, yet to permit reuse and automation over a wide range of specifications of the same category of hardware component. In our case, this formalization constrains our ISA descriptions to RISC instruction sets. We can e.g. use it to describe the MIPS-II, RISC-V and ARM Cortex ISAs, is however not limited to these kind of instruction sets.

Conceptually, MetaRISC describes the programmer's view on the CPU core. The metamodel therefore contains elements for architectural state, instruction behavior and for instruction encoding. The architectural state of the core describes every state element visible to the compiler. Examples for these elements are the program counter, state flags and the memory and register file of the core. The description of the instruction behavior covers how the execution of any instruction modifies the architectural state of the CPU core. The instruction encoding covers both the detection of the correct instruction and the decoding of immediate parameters and register addresses encoded in the instruction.

For both the description of the architectural state and the description of the behavior, we rely on the auxiliary Metamodels described in Sect. 3.2. This is a key aspect that helps us transferring the Model-of-Things specification to Model-of-Design microarchitectures implementing them.

Before discussing the ToD, it is important to note that the behavior described in the MoT is not used to synthesize the implementation directly. Instead, it defines a reference behavior, the MoD being constructed by the ToD (without using the behavior) has to comply to. Therefore, the behavior can be used - amongst others as documentation generation - to generate a testbench or properties that validate the generated view.

7.2 Assembly of a Template-of-Design for Micro-Architecture Generation

The most substantial and elaborate part of the MDA flow is the assembly of a Template-of-Design that constructs Model-of-Design instances which describe an implementation of the formalized specification captured in the Model-of-Things.

From a birds-eyes view, a generic ToD pieces together parts of the Model-of-Design so that the functionality described by the Model-of-Things is provided. This is a relatively straightforward process for components such as the FIR filter described in Sect. 5.3.

For the CPU core generator that is implemented here, the ToD provides a pipelined architecture that implements the CPU's Instruction Set. Again, the bulk of this construction is a rather straightforward instantiation of MetaRTL structures and instances that are part of these structures as e.g. introduced in [8]. These for example define the overall pipeline structure and the functional units in the ALU of a CPU. The Template-of-Design provides automation for these tasks as sub-components may be instantiated or not depending on the MoT. For example, functional units of the ALU may be instantiated only if there are instructions that rely on them.

For some sub-components of the CPU core however, a large dependency between their inner structure and the Model-of-Things exists. Decoding of instruction arguments and implementation of Control Unit have high dependency on the encoding specified by the ISA Model-of-Things. From the Model-of-Design's perspective, a control unit is a decoder which can decode up to n-of-n out wires for a certain input combination. To provide this description, the Template-of-Design has to fulfill the behavior described in the MoT.

Here, Python as the framework's underlying language is very useful: it allows to use a scripting approach for fast and creative solutions. As both source and result data are embedded into the formalized MDA stack, re-usability is not sacrificed.

The construction routines which were built as part of the ToD for example allow to create a testbench which uses both the models of instruction behavior and a skeleton of the generated CPU to find working control signal vectors for each of the instructions part of the ISA. For our simplistic architecture, it was sufficient to brute-force all possible combinations of control signals for an instruction to find the combinations that yield a behavior consistent with the ISA description. More sophisticated methods such as functional equivalence checking with formal verification tools might be an option we follow in the future. Both approaches can be carried out from within the MoT-to-MoD transformation thanks to the flexibility of Python.

7.3 Results and Discussion

The presented hardware MDA approach was applied to a MIPS2 integer instruction set and the RISC-V [1] integer instruction set with the compressed instruction set extension being mapped to 2-, 3-, 4- and 5-stage pipelined microarchitectures. To evaluate the benefits of our multi-level transformation approach, we compared our hardware MDA approach to a direct MoT-to-code approach (utilizing a Python-based template engine called Mako). Our comparison showed that the MDA approach requires about a factor of 10x less code compared to generating VHDL with Mako templates. They key part of this reduction comes from the high re-usability of ToD code compared to template code and partly

also originates from a higher expressiveness of Python and the automation in the transformations.

The Metamodel based specification of MoT and MoD provides an intuitive interface to models and gives a low entry barrier. The APIs and object oriented approach also supports generic template components. For example, we implemented a pipeline template which automatically generates the registers and connections between different pipeline stages. This pipeline template was developed in a one-time effort, can however be reused for other designs. Everything else just works, shielding designers from thinking in simulation semantics allowing them to focus on the challenging aspects of the design task.

8 Related Approaches

There are many academic and commercial approaches generating RTL code from UML or other graphical notations. Although these approaches use Metamodels, none of them links to specification and has a hardware design aware model.

The idea of focusing on design – i.e. in our approach on the Model-of-Design (MoD) – and not on simulation semantics is not new. For example, UDL/I [2,9], the *unified design language for integrated circuits definition*, first targets the design intent and only as a second step the execution of its models. UDL/I was developed and standardized in the early 90ties by Japan Electronic Industry Development Association JEIDA, almost in parallel with Verilog and VHDL. It did however not make it to a wide distribution. Design aspects instead of simulation aspects are also the focus of UPF [11], since it permits the insertion of things like level-shifters or isolation cells without explicitly providing a specification of their simulation characteristics. Digital design related descriptions are also supported by EDA tools, either through the use of generic components in graphical editors or as intermediate step in the RTL synthesis process. In comparison to our approach, these description styles lack an explicity underlying Metamodel, support for hardware generation, an explicit, automated link to specification and powerful measures to describe connectivity and architecture alternatives.

The FIRRTL (Flexible Intermediate Representation for RTL, [15]) format is a textual intermediate that is used when Chisel hardware generators [3] are compiled to Verilog. Similar to our Model-of-Design model, FIRRTL models describe design items, can use powerful features to specify instances and connectivity in a simple way, provide hooks for further optimization and can be translated to an RTL view (Verilog).

FIRRTL offers the possibility to hook optimization routines and transformations. Chisel with FIRRTL thus shares the ideas of generation, intermediate transformations and design thinking with our approach.

It differs from our approach as it only defines an intermediate language and not an intermediate model, i.e. it has no explicit Metamodel. Also, FIRRTL does not share a common type and expression system with its high-level counter part Chisel. FIRRTL further does not provide a link to MoTs or specifications and a ToD approach allowing to merge (micro-)architecture specification with

methods deriving and processing data from MoTs. In other words, our approach utilizes one common infrastructure to automate design from specification to implementation while Chisel with FIRRTL start with the (micro-)architecture and relies on a language based description.

9 Summary and Outlook

We presented an MDA inspired approach to automate hardware design starting at the specification level. This approach introduces three models namely The Model-of-Thing, the Model-of-Design, and the Model-of-View structuring specification, design, and views. As MDA proposes, we translate these views from the more abstract one to the next concrete one. We used for the translation Python code that is described in a way that it reflects the design and the view. Therefore, we call this Python style Template-of-Design and Template-of-View. This style is enabled by APIs generated from the Metamodel definitions of the models mentioned before.

We have proven the usefulness of the approach by generating RTL code from specifications. By doing so, we have seen the potential for productivity increase when following our proposed approach.

Next steps will introduce further transformations, especially for translating one model in an other on the same abstraction level. In addition, we want to further elaborate the best way of transformation steps being needed to translate a specification into a design. Finally, we will continuously develop new components to harden the claim and to get experience with the transformations.

References

1. Asanović, K., Patterson, D.A.: Instruction sets should be free: the case for risc-v. Technical report UCB/EECS-2014-146, EECS Department, University of California, Berkeley, August 2014
2. Japan Electronic Industry Development Association. UDL-I : Unified Design Language for Integrated Circuits definition. UDL/I Language Reference Manual, Version 2.0.3, Translation from the Japanese Language Reference Manual. JEIDA (1993)
3. Bachrach, J., Vo, H., Richards, B.C., Lee, Y., Waterman, A., Avizienis, R., Wawrzynek, J., Asanović, K.: Chisel: constructing hardware in a scala embedded language. In: The 49th Annual Design Automation Conference 2012 (DAC 2012), San Francisco, CA, USA, 3–7 June 2012, pp. 1216–1225 (2012)
4. Collet, R., Pyle, D.: McKinsey on semiconductors: what happens when chip-design complexity outpaces development productivity, Autumn 2013. http://www.mckinsey.com/industries/semiconductors/our-insights
5. Ecker, W., Schreiner, J.: Metamodeling and code generation in the Springer Science+Business Media Dordrecht. In: Ha, S., Teich, J. (eds.) Handbook of Hardware/Software Codesign, pp. 1–41. Springer, Dordrecht (2016). doi:10.1007/978-94-017-7358-4_32-1

6. Ecker, W., Velten, M., Zafari, L., Goyal, A: The metamodeling approach to system level synthesis. In: Fettweis, G., Nebel, W. (eds.) DATE, pp. 1–2. European Design and Automation Association (2014)
7. Eker, J., Janneck, J.W., Lee, E.A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming heterogeneity - the ptolemy approach. Proc. IEEE **91**(1), 127–144 (2003)
8. Hennessy, J.L., Patterson, D.A.: Computer Architecture, Fifth Edition: A Quantitative Approach, 5th edn. Morgan Kaufmann Publishers Inc., San Francisco (2011)
9. Hoshino, T.: UDL/I version two: a new horizon of HDL standards. In: Proceedings of the 11th IFIP WG10.2 International Conference on Computer Hardware Description Languages and their Applications (CHDL 1993), Sponsored by IFIP WG10.2 and in cooperation with IEEE COMPSOC, Ottawa, Ontario, Canada, 26–28 April 1993, pp. 437–452 (1993)
10. IEEE. IEEE 1685^{TM}: IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP Within Tool Flows. IEEE
11. IEEE. IEEE 1801^{TM}: Standard for Design and Verification of Low Power Integrated Circuits. IEEE
12. Keutzer, K., Newton, A.R., Rabaey, J.M., Sangiovanni-Vincentelli, A.: System-level design: orthogonalization of concerns and platform-based design. Trans. Comp. Aided Des. Integ. Circ. Sys. **19**(12), 1523–1543 (2006)
13. Koeplinger, D., Delimitrou, C., Prabhakar, R., Kozyrakis, C., Zhang, Y., Olukotun, K.: Automatic generation of efficient accelerators for reconfigurable hardware. In: Proceedings of the 43rd International Symposium on Computer Architecture (ISCA 2016), Piscataway, NJ, USA, pp. 115–127. IEEE Press (2016)
14. Liangora Research Lab. What is MDA? Why considering BNPM
15. Li, P.S., Izraelevitz, A.M., Bachrach, J.: Specification for the FIRRTL language. Technical report UCB/EECS-2016-9, EECS Department, University of California, Berkeley, February 2016
16. Mellor, S.J., Scott, K., Uhl, A., Weise, D.: Model-driven architecture. In: Bruel, J.-M., Bellahsene, Z. (eds.) OOIS 2002. LNCS, vol. 2426, pp. 290–297. Springer, Heidelberg (2002). doi:10.1007/3-540-46105-1_33
17. Nikolic, B.: Simpler, more efficient design. In: ESSCIRC Conference 2015–2041st European Solid-State Circuits Conference, Graz, Austria, 14–18 September 2015, pp. 20–25 (2015)
18. OMG. MDA - The Architecture of Choice for a Changing World (2016)
19. Schreiner, J., Willgerodt, F., Ecker, W.: A new approach for generating view generators. In: Proceedings of DVCON US 2017 (DVCON 2017). IEEE Press (2017, unpublished)
20. Shacham, O., Azizi, O., Wachs, M., Richardson, S., Horowitz, M.: Rethinking digital design: why design must change. IEEE Micro **30**(6), 9–24 (2010)
21. Steinberg, D., Budinski, F., Paternostorno, M., Merks, E. (eds.): EMF Modeling Framework. Addison Wesley, Reading (2008)
22. Truyen, F.: The Fast Guide to Model Driven Architecture (2006)

Improving the Efficiency of Formal Verification: The Case of Clock-Domain Crossings

Guillaume Plassan[1,2(✉)], Hans-Jörg Peter[1], Katell Morin-Allory[2], Shaker Sarwary[1], and Dominique Borrione[2]

[1] Synopsys Inc., Mountain View, USA
{guillaume.plassan,HansJorg.Peter,shaker.sarwary}@synopsys.com
[2] Univ. Grenoble Alpes and CNRS, TIMA Laboratory, 38031 Grenoble, France
{katell.morin-allory,dominique.borrione}@univ-grenoble-alpes.fr

Abstract. We propose a novel semi-automatic methodology to formally verify clock-domain synchronization protocols in industrial-scale hardware designs. To establish the functional correctness of all clock-domain crossings (CDCs) in a system-on-chip (SoC), semi-automatic approaches require non-trivial manual deductive reasoning. In contrast, our approach produces a small sequence of easy queries to the user. The key idea is to use counterexample-guided abstraction refinement (CEGAR) as the algorithmic back-end. The user influences the course of the algorithm based on information extracted from intermediate abstract counterexamples. The workload on the user is small, both in terms of number of queries and the degree of design insight he is asked to provide. With this approach, we formally proved the correctness of every CDC in a recent SoC design from STMicroelectronics comprising over 300,000 registers and seven million gates.

Keywords: Formal verification · Clock-domain crossing · Synchronizers · CEGAR · SOC

1 Introduction

Modern large hardware designs typically contain tens of clock domains: different modules use different clocks, adapting consumption and performance to the ongoing tasks, thereby reducing the overall power consumption of the chip.

Moreover, an SoC typically assembles IP blocks coming from various teams, and each block may be optimized for a specific operating frequency. As a result, such architectures create many interconnections between the various clock domains, so-called *clock-domain crossings* (CDCs). To ensure a correct propagation of data through a CDC, hardware designers have to implement specific protocols and modules: *synchronizers*.

With the increasing number of CDCs and synchronization protocols as well as the huge complexity of modern SoCs, proving the functional correctness of

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 108–129, 2017.
DOI: 10.1007/978-3-319-67104-8_6

all synchronizers became a major challenge. While incomplete functional verification methods, such as testing based on simulation, scale for large designs, they are only able to show the absence of functional errors in a subset of the full design behavior. For exhaustively and automatically proving the correctness of functional properties, *model checking* is the prevalent technique in a modern VLSI design flow.

But, as model checking a property is not scalable on large hardware designs, the question arises whether it is really necessary in practice to have a *completely* automatic verification procedure. That is, can we somehow take the user into the loop and abandon the high degree of automation of model checking to make formal verification scalable?

This paper addresses this question and proposes a new comprehensive methodology for verifying clock-synchronization properties over industrial-scale SoC hardware designs.

Unlike other semi-automatic approaches that require non-trivial manual work in form of deductive reasoning, our approach produces a small sequence of easy queries that only require *local* design knowledge from the user. The key idea is to use the CEGAR principle [8] as the algorithmic back-end, where we let the user influence the course of the algorithm based on information extracted from intermediate abstract counterexamples. The workload on the user is deliberately kept small both in terms of number of queries and the degree of design insight to be provided.

More precisely, this paper makes the following contributions:

- A general semi-automatic algorithm based on CEGAR-based model checking.
- A heuristic to automatically infer design constraints from abstract counterexamples, which are then proposed to the user.
- A comprehensive methodology for verifying CDCs, based on this interaction between the model checking algorithm and the user.
- The application of this new methodology to conclusively prove the correctness of all CDCs on a recent industrial SoC design.

The paper continues as follows. Section 2 recalls the CDC challenges and the various synchronizers. Section 3 presents the state-of-the-art CDC verification flow and its limitations. We provide a novel automated flow in Sect. 4, and the results obtained with it in Sect. 5. We finally compare our approach with the related works before concluding the paper.

2 Clock-Domain Crossing Issues

A CDC typically manifests itself as a digital signal path between two sequential elements receiving clocks from out-of-phase domains (Fig. 1). Even if those clocks have the same frequency, any difference between their phases introduces a latency between their rising edges. This non-predictable behavior is precisely the challenge of designing CDCs.

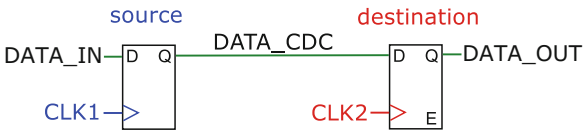


Fig. 1. A simple CDC

Multiple problems arise from CDCs [25], and designers need to implement specific structures to avoid any issue [13]. Consequently, the CDC verification tool must check that all the potential problems have been addressed and corrected.

2.1 Metastability and Multi-flops

The definition of clock domains directly implies that when data changes in the source domain of a CDC, the destination register can capture it at any moment: compliance with the setup and hold time requirements is not guaranteed. Hence, because of a small delay between the rising edges of the two clocks, the data is captured just when it changes, and a metastable value may be propagated (as shown in Fig. 2).

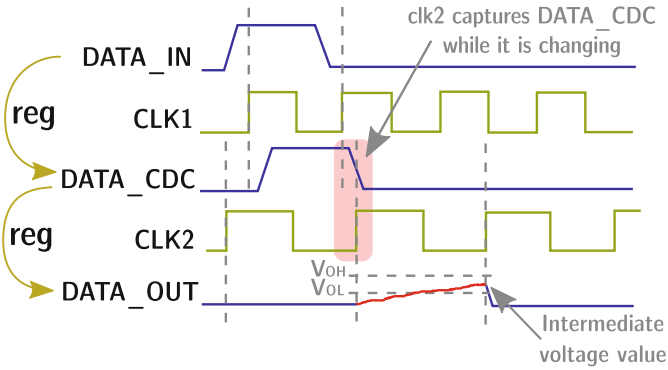


Fig. 2. Metastability behavior

The metastability phenomenon has been identified a few decades ago [7]: if a metastable value is propagated through combinational logic, it can lead to a so-called *dead system*. And it would be very difficult to find the source of this issue after fabrication, as post-production testers do not understand non-binary values.

A first solution would be to introduce a latency in the destination domain, in order to wait for a stabilization of the value. This timing can be estimated

by considering the clock frequencies and the production technology, as is commonly performed in the *Mean Time Between Failure* computation [12]. A single dedicated register could then, if properly sized, output a stable data. However, such synchronizing registers would result in a significant overhead on the circuit size. Another technique [16,20] involves embedding a monitor in the design which detects and corrects metastable values. However, the overhead would also be significant.

The most common solution is to add latency by implementing cascaded registers [14] (see Fig. 3). While this *multi-flop* structure guarantees within a certain probability that the propagated value is stable, there is no way of telling if it is a '0' or a '1'. Indeed, the data being captured during a change, the multi-flop may output the old or the new value during one cycle; then, at the next destination cycle, the new value is propagated. The drawback of this structure is thus a delay in the data propagation.

2.2 Coherency with Gray-Encoding or Enable Control

When synchronizing buses, there can be *coherency* issues: if some bits of a bus have separate multi-flop synchronizers (see Fig. 3), it cannot be guaranteed that all these synchronizers require a strictly identical latency to output a stable value. When capturing a toggling signal, some multi-flops may settle to the old value and some to the new one. The resulting bus value may then become temporarily incoherent. If multiple synchronized bits converge on a gate, a transient inconsistent value may even be generated.

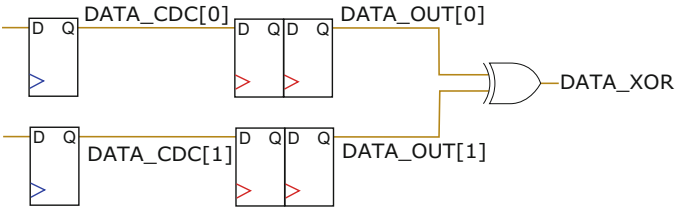


Fig. 3. Bus synchronization

For instance, in Fig. 4, two bits of *DATA_CDC* are toggling at the same cycle. After being synchronized by separate multi-flops, the *DATA_OUT* bus value is not consistent anymore. If both bits are converging on an exclusive OR gate, a glitch can be observed. However, we can add some encoding so that only one bit can change at a time [10] (Gray-encoding in the case of a counter, or mutual exclusion in some other cases). Even if the multi-flops stabilize this toggling signal with different latencies, the bus output value will either be correct regarding the previous or the next cycle. Thus, no false value is propagated. This can create some data loss, but avoids incoherency.

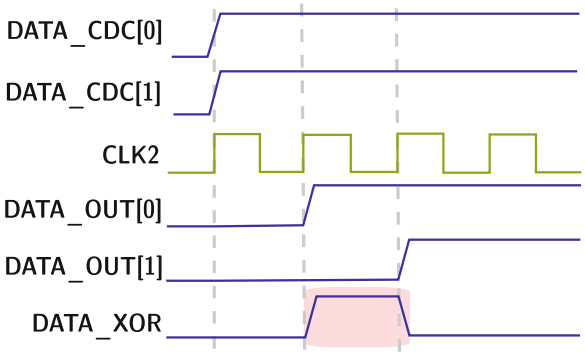


Fig. 4. Bus incoherency behavior

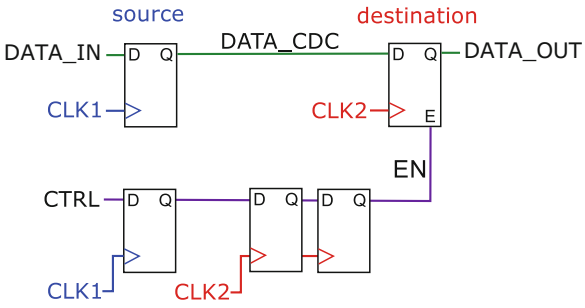


Fig. 5. Enable-based synchronization (Color figure online)

An alternative solution is using a control signal connected to the enable pin of the destination registers (Fig. 5). This *CTRL* signal is set to '1' only after *DATA_IN* stabilizes. Hence, no metastability can be propagated in the destination domain, and there is no need for further resynchronization [10]. Of course, this *'stable'* information comes from the source clock domain, so this control signal must be resynchronized in the destination domain (here with a multi-flop). Note that different synchronization schemes are derived from this structure. The control signal is here connected to the enable pin of the flop (the selection of a recirculation mux), but it could also be connected to a clock-gate enable, or even an AND gate on the data path.

2.3 Data Loss and Handshake

The enable-based synchronizer structure only propagates stable data. However, if the source register keeps sending data, the destination might wait for their stability and lose some packets. The source should then wait for the data to be captured before sending a new one. This can be done with a handshake protocol using request/acknowledge signals, as shown on Fig. 6. (In Figs. 5, 6 and 7, clock

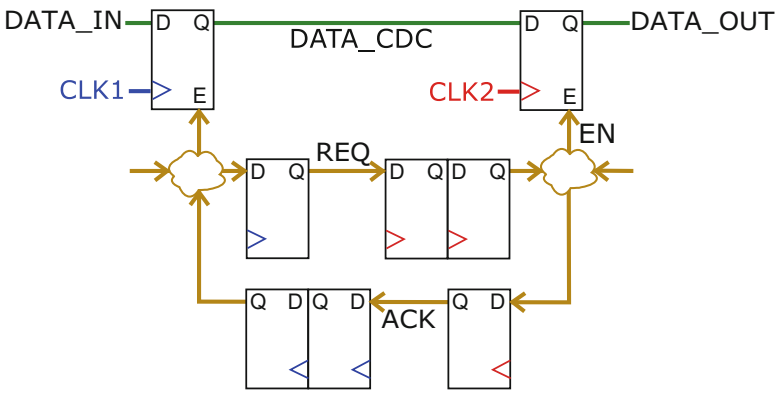


Fig. 6. Handshake synchronization (Color figure online)

domains are shown in blue or red, data is in green and control logic is in yellow or purple.)

2.4 Performance with FIFO

The delay introduced by handshake protocols may not be acceptable for a high-rate interface. Putting a FIFO in the CDC allows the source to write and the destination to read at their own frequencies, and increases the data propagation efficiency. In a FIFO, all the previous schemes are implemented (see Fig. 7). The main controls of the CDC are the write and read pointers, which need to be Gray-encoded before being synchronized by a multi-flop. In order to activate

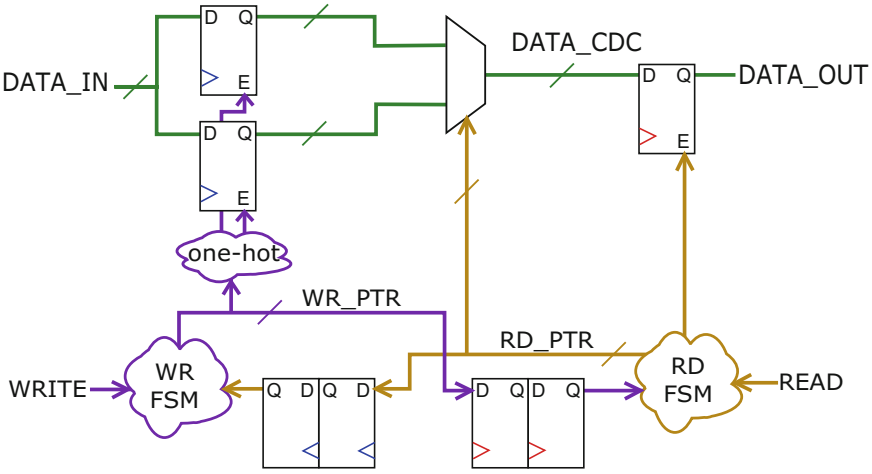


Fig. 7. FIFO synchronization (Color figure online)

the source or destination access, global write and read control signals can be implemented in a handshake protocol.

Using a FIFO implies some data latency (caused by the handshake and the resynchronization of pointers), but allows a higher transfer rate. All the previously mentioned issues are avoided (metastability, coherency, data loss), but its complexity makes the FIFO the most difficult synchronizer to design and verify.

3 Current Verification Approach

While some hardware bugs can sometimes be resolved by the firmware or software layers, incorrect synchronizers typically lead to non-correctable, so-called *chip-killer* bugs. To guarantee the absence of CDC issues in a design, a methodology is needed to check that all the necessary synchronizers are implemented, and that their protocol is followed (Fig. 8).

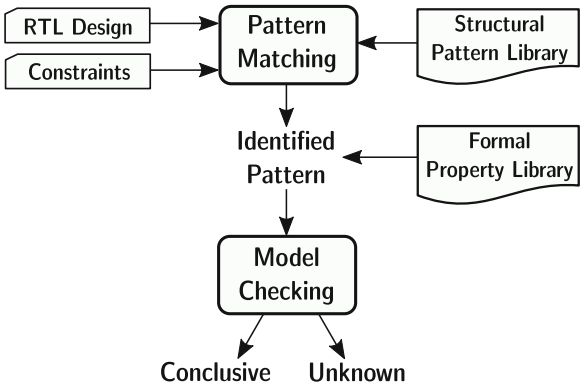


Fig. 8. CDC formal verification methodology

3.1 Structural Checks

After register-transfer-level (RTL) synthesis, on a net-list, it seems easy to structurally detect a CDC between two registers, and even to detect a multi-flop. In contrast, for complex FIFO protocols, identifying the correct control logic is non-trivial. If a single synchronizer structure was used, a proper pattern matching could try to identify it on each CDC. Unfortunately, in industry, many designers create their own synchronizing structures, and the structural library used for pattern matching would never be exhaustive on complex structures such as FIFOs.

In order to provide a robust and automated analysis, state-of-the-art CDC tools provide a more flexible approach which identifies more general patterns (like the one based on enables), without relying on the rigid FIFO or Handshake structures. This is a first quick step to check multi-flops and sort out missing synchronizers. However, a structural approach cannot check protocols and assumptions on the control signal. A functional check must then be run.

3.2 Functional Checks

From the recognition of a synchronizer structure, the extracted information is reused to run functional checks. Formal safety properties to be checked are associated to the generic structures we are using, among which:

1. **Stability**
   The destination register only propagates `DATA_CDC` *when it is stable.*
2. **Coherency** (*Gray-encoding check*)
   At most one bit at a time can change in `DATA`.

As an aid to the user, these formal properties are embedded in the CDC tools and linked to the matched patterns. When a pattern is detected, the properties are automatically synthesized in hardware, mapped to the corresponding RTL signals and formally checked.

3.3 Limitation

After running the structural and functional checks of Fig. 8, the user expects to know which data is correctly resynchronized and which is not. However, experience shows that model checking may not achieve a conclusive result on the properties: some of them reach a timeout even after several days. When this occurs, no information is returned on the cause of the timeout, and the designer is left with no clue on the possible presence of a metastability in the design.

It is well-known that inconclusive results in formal verification are caused by the so-called *state-space explosion problem* which is intrinsic to model checking of hardware designs. In practice, the typical approach to overcome this challenge is, for each property, to extract the CDC logic. The verification is then focused on just a small but relevant part of the design. However, this approach comes with the following issues: First, the verification engineer needs to have a very good understanding of the underlying design, which is not realistic for large RTL models; Then, strong time-to-market constraints do not allow a manual labor-intensive selection of appropriate abstractions for each property; Finally, even with such a high manual effort, a conclusive result cannot be guaranteed.

The approach presented in this paper is also based on focusing on a subpart of the design, but tries to overcome the aforementioned issues by following a CDC-oriented methodology that is based on an interaction between the user and a refinement algorithm.

3.4 Root Causes of Inconclusive Results

In this subsection, we report on common root causes of inconclusive results we observed in the verification of CDCs.

Operation Modes. An SoC can operate in many different modes (initialization, mission mode, test, scan, etc.) controlled by configuration signals, the values of which cannot be automatically inferred by the verification tool. The user must then provide functional design constraints such as clock frequencies, static value of configuration signals, etc. to perform the verification on a realistic mode. This method is user time consuming and error prone, as the user may fail to provide some essential signal constraint.

Clock Gating. In complex low-power designs, some modules can be enabled or disabled via a clock-gate for power saving. If the clock enable signals take inconsistent values, the tool produces unrealistic failures by exercising unreachable states of the design. The user should provide constraints on the value of the clock enable signals.

Protocols. In addition to design setup, functional assumptions should be given on the primary inputs of the design, e.g., for handshake protocols.

Considering all the above, in all practical cases we encountered, inconclusiveness was primarily caused by missing constraints. But even with all this information – that is not always trivial to write – a model checker may still not reach a conclusive result, due to the design complexity. We need a new approach both to tackle this complexity issue and to identify missing constraints.

4 User-Aided Abstraction Refinement

The objective of our approach is to avoid the state-space explosion problem in model checking hardware designs. To that end, our key idea is to let the user *aid* the model checking process by replying yes/no to a series of questions whose answers only require local design knowledge.

Technically, our underlying framework is a *counter-example-guided abstraction refinement* (CEGAR) [8] algorithm: we maintain a sequence of abstractions with increasing precision until a definite result can be established. In contrast to fully automatic CEGAR approaches, the user here influences the refinement process. We therefore call our approach *user-aided abstraction refinement* (UsAAR). Figure 9 gives an overview on the semi-automatic algorithm in the context of the overall methodology.

4.1 Localization Abstractions

Our abstractions are obtained via *localization reduction* [18]: we replace some nets in the original design with primary inputs, called *cut points*. An abstraction A is more precise than an abstraction B if the cone-of-influence (with cut-points) of the property in A is an extension of the one in B.

The rationale for this notion of abstraction is that, in practice, all the relevant control logic for a given CDC is implemented *locally*. Thus, properties requiring

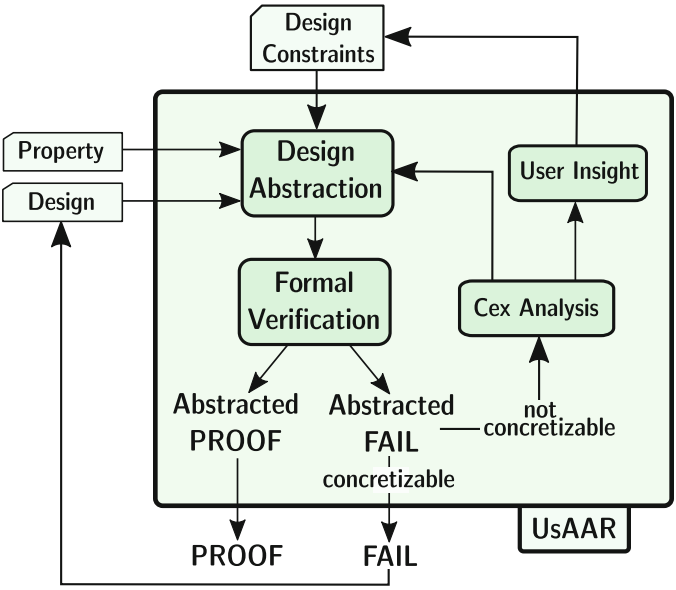


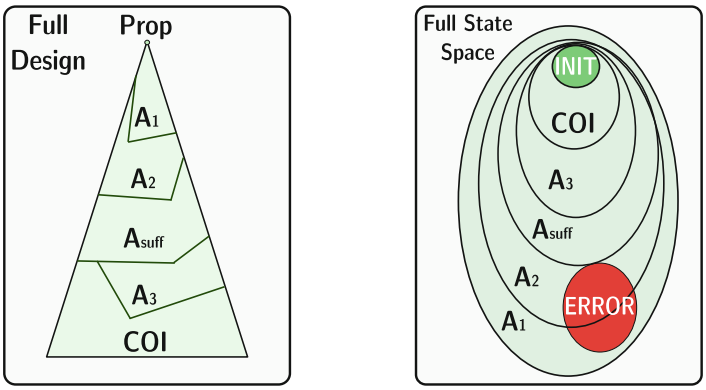
Fig. 9. The UsAAR algorithm of our CDC verification methodology

the correctness of synchronizing protocols should have small abstractions that suffice to either prove the property or to reveal bugs.

Figure 10 illustrates the abstraction process for a correct, hard to prove, property. By removing parts of the circuit from the cone-of-influence of the property (keeping only A_1 from COI), and leaving the unconnected nets free, the set of reachable states is enlarged (i.e., it represents an over-approximation). As a result, states in which the property fails (the error states), initially unreachable, may become reachable (A_1). In this context, refining the abstraction consists in iteratively adding back some of the removed circuit, and as a consequence reducing the reachable state space (from A_1 to A_2), until a sufficiently precise abstraction is obtained ($\mathsf{A}_{\mathsf{suff}}$), for which no error state is reachable. The challenge here is to find that part of the design that can be pruned away without spuriously making any error state reachable.

4.2 The Core Algorithm

The algorithmic core of our methodology (Algorithm 1, in pseudo code) is a semi-automatic algorithm which is based on the automatic *counterexample-guided abstraction refinement* (CEGAR) [8] principle. A localization abstraction of the design is incrementally made more precise in a sequence of *refinement rounds*. In each round, the safety property is checked on the abstraction: if the property is satisfied, the algorithm terminates with Result "proof"; if a counterexample is found, a refinement heuristic decides whether and how the abstraction should be refined, or it concludes that the counterexample is *concretizable*, i.e., the



(a) Localization abstractions with different degrees of precision

(b) Corresponding sets of reachable states

Fig. 10. Various abstractions for a given design and property

counterexample is also valid for the full design, and the algorithm can terminate with Result "fail".

Throughout the algorithm, we maintain a set of constraints C_{global} and a set of nets F. We call F the *focus*: it induces a localization abstraction $D^{\#}$ of design D (Line 5). The constraints C_{global} are used when property P is checked on $D^{\#}$ (Line 6). Starting with no constraints (Line 2) and F just holding the nets in the combinational fan-in of P (Line 3), we incrementally add elements to both sets, thereby making the over-approximation more precise. Based on the abstract result $R^{\#}$ obtained in Line 6, we either immediately terminate (in case $R^{\#} = \textbf{proof}$), or continue analyzing the abstract counterexample $cex^{\#}$ (in case $R^{\#} = \textbf{fail}$). The next subsection details this analysis.

4.3 Analysis of Abstract Counterexamples

In our variant of CEGAR, the refinement heuristic first determines a set of cut points that are logically relevant for the abstract counterexample. This is done by computing (an over-approximation of) the *justifiable set* of cut points J (Line 11). The validity of the counterexample is independent of any cut point that is not part of J. Intuitively, any net that is not contained in a minimal justifiable set can be set to a random value without invalidating the reachability of the error state. But since computing a minimal justifiable set is a hard problem on its own, heuristics are used to compute a small but not necessarily minimal set. A common technique is to use ternary simulation to identify inputs that do not impact the overall validity of the counterexample.

Once J is obtained, the heuristic Analyze classifies each element in J into specific categories (Line 12): clock, reset, data, control, etc., using a backward

Algorithm 1. UsAAR for a design D and a property P

```
1: R ← unknown
2: C_global ← ∅
3: F ← CombFanin(P)
4: while R = unknown do
5:    D# ← Abstract(D, F)
6:    (R#, cex#) ← Check(P, D#, C_global)
7:    if R# = proof then
8:       // Terminate and report proof
9:       R ← proof
10:   else if R# = fail then
11:      J ← Justify(cex#, D#) \ F
12:      (C_prop, ref) ← Analyze(J, D)
13:      (C_acc, ref') ← Review(C_prop)    // User interaction
14:      ref ← ref ∪ ref'
15:      if C_acc = ref = ∅ then
16:         // Terminate and report fail
17:         R ← fail
18:      else
19:         // Refine the abstraction and continue
20:         C_global ← C_global ∪ C_acc
21:         F ← F ∪ ref ∪ Nets(C_acc)
22:      end if
23:   end if
24: end while
25: return R
```

traversal of the RTL which starts at the synchronizer pattern. Then, realistic constraints corresponding to each category are inferred. For instance, when encountering a potential setup issue such as a missing clock-gating constraint, Analyze proposes to assume that the control input of the clock-gate is always set to an enabling value. Or if a net is found to be logically irrelevant for the user, Analyze infers a stopper constraint to ensure that the net (and its fan-in) will not be part of any future abstraction. In the asynchronous FIFO of Fig. 7, this stopper constraint would be applied on the net *DATA_IN*. Indeed, in this case, the property is independent of the *DATA_IN* value. Only the following control logic is relevant.

All constraints C_{prop} inferred by Analyze are then reported for review (Line 13). In case the user rejects a constraint, the corresponding net is marked for automatic refinement. After the manual classification process, the accepted constraints C_{acc} are added to the set of global constraints C_{global} (Line 20). For all nets that are marked for automatic refinement *ref*, we extend the focus so that the subsequent abstraction are more precise by additionally comprising those nets (Line 21).

4.4 Soundness, Completeness, Validity

The algorithm terminates if either the model checker reports a *proof* or if no new constraints or nets for automatic refinement can be inferred, in which case a *fail* is reported (Line 15). The soundness of reported *proofs* follows straight forward from the fact that our localization abstraction represents an over-approximation. The soundness of reported *fails* follows from the definition of the justifiable set: the abstract counterexample $cex^{\#}$ only depends on nets within the focus, i.e., on nets that were not abstracted out or for which the user provided constraints. Hence, $cex^{\#}$ remains a valid counterexample for any greater set $F' \supset F$, and in particular, on the full design D.

In every non-terminating round, we either monotonically make the abstraction more precise or constrain the design behavior. Hence, since the underlying design is finite, the algorithm terminates. Completeness follows from the fact that the algorithm either terminates with a sufficient abstraction, or it ultimately reaches the full design, i.e., $D^{\#} = D$.

When manually adding constraints one runs the risk of over-constraining the design's behavior, which can lead to vacuous proofs. However, our UsAAR methodology is designed to minimize the risk of over-constraining. The setup constraints inferred by our heuristics are combinational and structurally close to the CDC control logic, which makes them easy for the user to review. Then, they do not over-constrain but ensure that the design does not exhibit spurious behavior. On the other hand, stopper constraints (i.e., static cut points) are conservative: they lead to an over-approximation that preserves all safety properties.

5 Case Study

We applied our new methodology on two hardware designs: a small parametric FIFO and a complex SoC from STMicroelectronics. The first one reveals the benefits of the different steps of the flow. The second one proves the validity of the methodology on an SoC from industry.

5.1 Asynchronous FIFO

Design Presentation. This hardware design includes a FIFO similar to the one presented in Fig. 7. To mimic a state-space explosion on the *DATA_IN* and *WRITE* paths of the source domain, an FSM was implemented with a self-looping counter on 128 bits, along with some non-deterministic control logic. Also, the source and destination clocks are enabled by sequential clock-gates, controlled by two independent primary inputs.

This design is parameterized by the width of the data being propagated, and by the depth of the FIFO. By varying these two size parameters, we increase the design complexity and analyze the corresponding performance of the methodology.

Results. Using an industrial tool to structurally analyze the design, three formal properties were extracted.

- A data stability property is created on signal DATA_CDC.
- Two coherency properties are extracted on the address buses after synchronization, one on RD_PTR and one on WR_PTR. Indeed, the write and read pointers are synchronized with multi-flops, and should then follow Gray-encoding (see Sect. 2.2).

To verify them, the open source model checker ABC [5] is used with the engines *PDR* [11] and *BMC3* [4] in parallel. For each property, the runtime limit for timeout is set to 15 min (denoted *T/O* in Table 1). We run the experiments on a workstation with 24 Intel Xeon 2.6 GHz CPUs and 220 GB of memory. Four different schemes are applied to generate the results in Table 1:

1. **Standard:** Model-checking each property on the full (non-abstracted) design.
2. **CEGAR:** A UsAAR variant where we reject all constraints. It can be seen as a reduction of UsAAR to standard CEGAR.
3. **UsAAR:** The full semi-automatic algorithm presented in Sect. 4 including automatic refinement and constraint inference together with manual constraint classification.
4. **Standard w/ constraints:** Repeated run of the standard scheme with all the accepted constraints from the UsAAR scheme.

A first observation is that the coherency properties are proved in less than a second in all four schemes and variations of the design. This is not surprising considering that the Gray-encoding implemented in this design does not depend on any non-deterministic control logic. Henceforth, we will then focus on the data stability property.

The standard scheme is not able to prove the property in all 35 variations of the design (Column "Standard"). Using the simple CEGAR approach, the property is proved in all variations within 4 to 15 min (Column "CEGAR"). Interestingly, the proof runtime is stable when the FIFO depth is fixed and the data width increases. By looking at the last abstraction exercised, we notice that *DATA_IN* is always abstracted out. Its value does not depend on the source logic. Hence, heuristics from the proof engine inferred that the proof does not depend on the data value, which make the analysis as simple for 8 bits as it is for 128 bits. Actually, even if the source logic of the data was greatly more complex, the CEGAR result would be the same.

Along the UsAAR run, two static constraints are automatically inferred on the enables of the clock-gates. Because having a non-deterministically enabled clock is not a realistic design behavior, we decide to accept them. As a result, the stability property is solved in all 35 variations of the design within 10 s each (Column "UsAAR"). Same as with simple CEGAR and contrary to the standard scheme, the complexity of the data source logic is irrelevant for the proof.

Interestingly, even when applying the inferred enabling constraints on the standard scheme, not all properties can be solved (Column "Standard w/ constraints"). Also in this case, by comparing with column "UsAAR", we notice

Table 1. CDC properties proof CPU runtime (in sec) on the asynchronous FIFO

FIFO depth	Data width	Standard	CEGAR	UsAAR	Standard w/ constraints
3	8	T/O	389	7	22
	16	T/O	390	7	35
	32	T/O	392	7	66
	64	T/O	390	7	870
	128	T/O	391	7	T/O
4	8	T/O	592	6	15
	16	T/O	591	6	28
	32	T/O	594	6	57
	64	T/O	593	6	145
	128	T/O	594	6	243
5	8	T/O	641	7	14
	16	T/O	651	7	53
	32	T/O	641	7	69
	64	T/O	640	7	180
	128	T/O	693	7	374
6	8	T/O	558	7	13
	16	T/O	558	7	55
	32	T/O	563	7	62
	64	T/O	563	7	203
	128	T/O	562	7	414
7	8	T/O	574	7	10
	16	T/O	574	7	49
	32	T/O	575	7	68
	64	T/O	574	7	150
	128	T/O	575	6	841
8	8	T/O	589	7	11
	16	T/O	590	7	36
	32	T/O	579	7	60
	64	T/O	580	7	150
	128	T/O	580	7	463
9	8	T/O	868	9	14
	16	T/O	863	9	43
	32	T/O	868	9	74
	64	T/O	864	9	210
	128	T/O	865	9	475
TOTAL PROVED		0	35	35	34

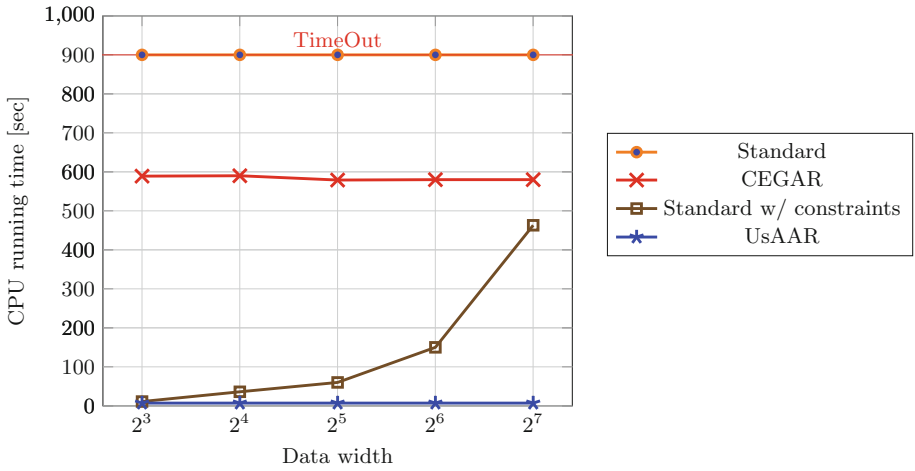


Fig. 11. Performance comparison for FIFO depth 8

that the runtime is always higher than when using both the inferred constraints and CEGAR. This observation along with Fig. 11 points out the importance of using both CEGAR and constraints in order to reach a conclusive result.

5.2 CPU Subsystem

Design Presentation. The second case study is a complex SoC hardware design from STMicroelectronics, intended for a gaming system. It is a low-power architecture, with a state-of-the-art quad-core CPU and many different interfaces. In total, it holds over 300,000 registers and 7 million gates. The CDC setup is mainly done in a clock and reset control module, which selects configurations for the whole system among its 38 clock domains and 17 primary resets. However, many configuration signals (such as clock-enable signals) are not controlled by this module. Since the design has a Globally Asynchronous Locally Synchronous (GALS) intent, CDC signals are always synchronized in the destination module.

Figure 12 gives an overview of some synchronizations around the CPU. Data communication with the CPU environment (the rest of the SoC) is synchronized by a customized FIFO with a 4-phase protocol based on the one described in Sect. 2.4, with additional low power and performance optimizations. Only one communication is shown in Fig. 12, among the ten in each direction. The figure also shows the communications with the clock and reset controller, and the handshake with the low power management block. Note that the CPU is one central module which, due to its complexity, is likely to cause a timeout in the model checking algorithm when considered in its entirety.

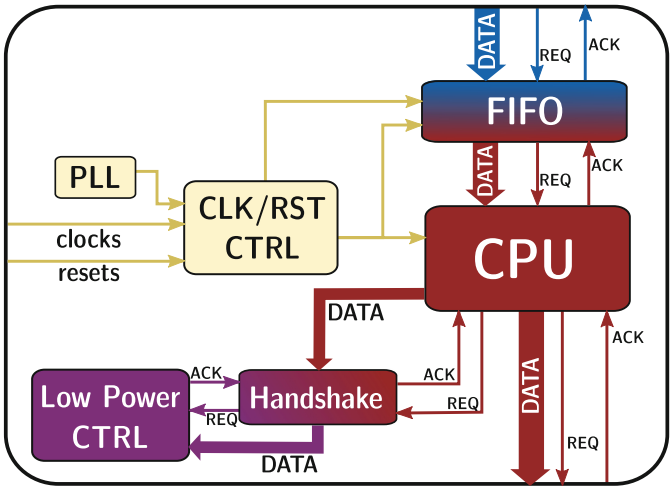


Fig. 12. Overview on the synchronizers at the interface of the CPU

Many synchronizers (mostly FIFOs) are split between modules of this subsystem. Hence, we cannot proceed in a module-by-module CDC analysis. Working at this hierarchy level is particularly relevant for us.

Results. We used an industrial tool to structurally analyze the STMicroelectronics design. Only some straightforward constraints regarding reset and clock setup were applied; we did not use any other design insight. All clock multiplexers were constrained to select the mission-mode clock, and static primary inputs were constrained to the value given in the design specification (subsystem configuration). The structural analysis identified several thousand synchronizers, most of them multi-flops which do not need a functional check. It also extracted 78 stability and 47 coherency properties. We verify each extracted property in the same four schemes that were presented previously.

Table 2 shows the results for model checking the stability and Gray-encoding properties. Without any automatic refinement, the standard scheme can only prove 40 out of 125 properties (Column "Standard"). After increasing the timeout limit to several hours, the same results are obtained. Using automatic refinement (the CEGAR scheme), 33 more properties can be proved (Column "CEGAR"). Also, it should be noted that all proofs from the standard scheme get confirmed by the CEGAR scheme. CEGAR proves to be particularly efficient for proving Gray-encoding properties, as the encoding logic is generally local to the synchronizer.

The most striking observation, however, is that during the first UsAAR run, 40 setup constraints are automatically inferred and are all easily accepted. These include global interface enables (scan or test enables, internal configuration signals, ...), and also internal soft resets and clock-gate enables which were missing

Table 2. Results on the CDC stability and Gray-encoding properties

		Standard	CEGAR	UsAAR first run	**UsAAR second run**	Standard w/ constraints
Stability	# Proof	29	31	45	**78**	43
	# Fail	0	0	33	**0**	0
	# Inconclusive	49	47	0	**0**	35
	CPU time [min]	771	734	583	**31**	557
Gray-enc.	# Proof	11	42	42	**47**	11
	# Fail	0	0	5	**0**	0
	# Inconclusive	36	5	0	**0**	36
	CPU time [min]	540	86	27	**15**	540

in the design specification. It leads to 87 proved properties and provides counterexamples for the remaining 38. Note that in those abstract counterexamples, many irrelevant signals are automatically hidden using the justifiable subset, which makes debugging easier.

By reviewing them, we observe spurious behaviors in the handshakes, which are fixed by adding 22 missing Boolean assumptions enabling the protocols. Indeed, in some cases the *WRITE* or *READ* of the FIFO represents an information coming from the CPU, and would depend on a software execution. When these signals are abstracted out, they take random values which do not follow the handshake protocol, hence creating a failure. After consulting STMicroelectronics, we decide to constrain them to a realistic behavior. Here, the worst case would be to set them to '1' which would mean the CPU always transfers data. We stress the fact that no deep design knowledge is needed during this process, and the constraints represent a realistic design behavior.

With these new constraints, the second UsAAR run is able to conclude all 125 properties correctly. Compared to the fully automatic approaches, the final UsAAR proof runtime is accelerated by more than 20×. In fact, the most difficult property concludes in only 7 min.

Finally, the last column shows that having the proper constraints is not sufficient to get proofs; the efficiency of UsAAR indeed relies on the *combination* of automatic CEGAR and manual constraint classification.

Regarding the size of the abstractions: on the full design, some properties have a cone-of-influence of more than 250,000 registers. Interestingly, our variant of CEGAR is able to find sufficient abstractions containing only up to 200 registers. This ratio confirms our assumption that only the local control logic has a real influence on the correctness of a CDC property.

Overall, a relevant metric to score the different flows would be the total time spent by the verification engineer starting with the design setup and ending with achieving conclusive results for all properties. It would allow us to conclude on the complexity and usability of different methodologies, as for instance the manual extraction and constraining explained in Sect. 3.3. However, this time depends very much on the design complexity, reuse, and user insight. Such an

experiment would assume the availability of two concurrent verification teams on the same design, an investment that could not be made by our industrial partners.

6 Related Work

The implementation of CDC synchronizers recalled in Sect. 2 is well known in the hardware design community. Tools for verifying such synchronizers are provided by leading EDA vendors (Synopsys SpyGlass CDC [27], Mentor Questa CDC [23], Real Intent Meridian CDC [24], ...). Most of these tools provide a verification flow including structural checks up to the generation of related formal properties.

Academia is also active in this research area. Some approaches focus on functionally verifying CDC synchronizers; e.g., Burns et al. proposed a new verification flow using xMAS models [6]. However, the user needs to define the boundaries of the synchronizers, which is not scalable.

Kwok et al. presented a verification flow [19] purely based on a structural analysis that matches parts of the design with a property library to generate assertions. These assertions can be model checked for functional verification. Litterick proposed a similar flow [22], replacing model checking by simulation on SVA assertions. Kapschitz and Ginosar published an overview [15] on the general CDC verification flow, showing the need for multiple clock modeling and formal verification. However, they did not detail how synchronizers can be identified, nor their flow automation, nor how to deal with a high design complexity.

Li and Kwok described a CDC verification flow [21] similar to ours, including the extraction of a formal property from an automatic structural identification. They performed abstraction refinement along with synthesis to prove some properties, but the underlying techniques were not explained in detail. In their flow, inconclusive properties after the abstraction refinement are *promoted* to the top-level and the user needs another methodology to proceed with the formal verification.

Recently, Kebaili proposed to improve the structural checks in order to detect the main control signals of the synchronizer [17]. The properties to be verified would only rely on these control signals (with a handshake-based protocol), hence avoiding the state-space explosion in the data path.

In other hardware verification domains, some methodologies combine manual with automatic reasoning. For instance, for verifying the FlexRay physical layer protocol, Schmaltz presented a semi-automatic correctness proof [26] in which the proof obligations are discharged using Isabelle/HOL and the NuSMV model checker. This proof was also applied to larger verified system architectures [1].

Localization abstractions and related refinement techniques were pioneered by Kurshan in the 1980s and eventually published in the mid 1990s [18]. The fully automatic variant of the CEGAR principle was introduced by Clarke et al. in the context of over-approximating abstractions defined through state-space partitionings [8,9]. The works by Andraus et al. propose a CEGAR approach for

data-paths in hardware designs [2,3]. Orthogonal to our approach, their abstractions are obtained by replacing data-path components by uninterpreted functions which, in turn, also requires a more powerful model checker based on SMT. Our methodology can be seen as an extension of these works mentioned above, as it enables the integration of user insight into the refinement process.

7 Conclusion and Outlook

This paper presents a complete formal verification flow for conclusively proving or disproving CDC synchronizations on industrial-scale SoC hardware designs. Our core contribution is a semi-automatic model checking algorithm, where the user aids the (otherwise fully automatic) verification process by classifying a sequence of automatically inferred constraints.

We demonstrated the efficiency of our approach on an STMicroelectronics SoC design which was persistently difficult to verify: prior approaches required to manually extract the cone-of-influence of the synchronizers, which resulted in a tedious (and costly) work for verification engineers.

In contrast, our new methodology allowed the full verification without requiring any deep design knowledge. This very encouraging practical experience suggests that we identified an interesting sweet-spot between automatic and deductive verification of hardware designs. On the one hand, it is a rather easy manual task to classify simple design constraints that refer to single nets where, on the other hand, this information can be crucial to guide an otherwise automatic abstraction refinement process.

Another positive side-effect of our methodology is that it gradually results in a functional design setup. Note that all accepted constraints (except the stopper constraints) do not depend on a certain property, but reflect general design properties and are therefore globally valid. This does not only speed-up the overall CDC verification time, when constraints are reused while verifying multiple properties, it also helps further functional verification steps in the VLSI flow. For instance, the same design constraints can be reused for functionally verifying false and multi-cycle paths.

As a next step, we plan to improve the constraint inference in order to generate sequential SVA assumptions. This feature would guide the user into creating more complex constraints representing realistic design behaviors without decreasing the proof coverage. Also, we investigate into other functional properties. The long-term goal is to extend our methodology to many critical functional verification steps in the VLSI flow.

Acknowledgement. We wish to thank Mejid Kebaïli and Jean-Christophe Brignone from STMicroelectronics for reviewing and confirming the validity of our methodology.

References

1. Alkassar, E., Böhm, P., Knapp, S.: Formal correctness of an automotive bus controller implementation at gate-level. In: Kleinjohann, B., Wolf, W., Kleinjohann, L. (eds.) DIPES 2008. ITIFIP, vol. 271, pp. 57–67. Springer, Boston, MA (2008). doi:10.1007/978-0-387-09661-2_6
2. Andraus, Z.S., Liffiton, M.H., Sakallah, K.A.: Refinement strategies for verification methods based on datapath abstraction. In: ASP-DAC, pp. 19–24 (2006)
3. Andraus, Z.S., Sakallah, K.A.: Automatic abstraction and verification of Verilog models. In: Design Automation Conference (DAC), pp. 218–223 (2004)
4. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999). doi:10.1007/3-540-49059-0_14
5. Brayton, R., Mishchenko, A.: ABC: an academic industrial-strength verification tool. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 24–40. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14295-6_5
6. Burns, F., Sokolov, D., Yakovlev, A.: GALS synthesis and verification for xMAS models. In: DATE (2015)
7. Chaney, T., Molnar, C.: Anomalous behavior of synchronizer and arbiter circuits. IEEE Trans. Comput. **C–22**(4), 421–422 (1973)
8. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 154–169. Springer, Heidelberg (2000). doi:10.1007/10722167_15
9. Clarke, E., Grumberg, O., Long, D.E.: Model checking and abstraction. In: ACM (1991)
10. Cummings, C.E.: Clock domain crossing design & verification techniques using systemverilog. In: SNUG, Boston (2008)
11. Eén, N., Mishchenko, A., Brayton, R.K.: Efficient implementation of property directed reachability. In: FMCAD, pp. 125–134 (2011)
12. Gabara, T.J., Cyr, G.J., Stroud, C.E.: Metastability of CMOS master/slave flip-flops. In: IEEE Custom Integrated Circuits Conference. pp. 29.4/1–29.4/6, May 1991
13. Ginosar, R.: Fourteen ways to fool your synchronizer. In: Asynchronous Circuits and Systems, pp. 89–96 (2003)
14. Ginosar, R.: Metastability and synchronizers: a tutorial. IEEE Des. Test Comput. **28**(5), 23–35 (2011)
15. Kapschitz, T., Ginosar, R.: Formal verification of synchronizers. In: Borrione, D., Paul, W. (eds.) CHARME 2005. LNCS, vol. 3725, pp. 359–362. Springer, Heidelberg (2005). doi:10.1007/11560548_31
16. Karimi, N., Chakrabarty, K.: Detection, diagnosis, and recovery from clock-domain crossing failures in multiclock SoCs. Comput. Aided Des. Integr. Circuits Syst. **32**(9), 1395–1408 (2013)
17. Kebaili, M., Brignone, J.C., Morin-Allory, K.: Clock domain crossing formal verification: a meta-model. In: IEEE International High Level Design Validation and Test Workshop (HLDVT), pp. 136–141, October 2016
18. Kurshan, R.P.: Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach. Princeton University Press, Princeton (1994)
19. Kwok, C., Gupta, V., Ly, T.: Using assertion-based verification to verify clock domain crossing signals. In: Design and Verification Conference, pp. 654–659 (2003)

20. Leong, C., Machado, P., et al.: Built-in clock domain crossing (CDC) test and diagnosis in GALS systems. In: Proceedings of the DDECS 2010, pp. 72–77, April 2010
21. Li, B., Kwok, C.K.: Automatic formal verification of clock domain crossing signals. In: ASP-DAC, pp. 654–659, January 2009
22. Litterick, M.: Pragmatic simulation-based verification of clock domain crossing signals and jitter using SystemVerilog Assertions. In: DVCON (2006)
23. Mentor Graphics: Questa CDC. https://www.mentor.com/products/fv/questa-cdc/. Accessed Jan 2017
24. Real Intent: Meridian CDC. http://www.realintent.com/real-intent-products/meridian-cdc/. Accessed Jan 2017
25. Sarwary, S., Verma, S.: Critical clock-domain-crossing bugs. Electron. Des. Strateg. News **53**, 55–64 (2008)
26. Schmaltz, J.: A formal model of clock domain crossing and automated verification of time-triggered hardware. In: FMCAD. pp. 223–230, November 2007
27. Synopsys: Spyglass CDC. https://www.synopsys.com/verification/static-and-formal-verification.html. Accessed Jan 2017

Improving Stress Quality for SoC Using Faster-than-At-Speed Execution of Functional Programs

Paolo Bernardi[1], Alberto Bosio[2(✉)], Giorgio Di Natale[2], Andrea Guerriero[1], Ernesto Sanchez[1], and Federico Venini[1]

[1] Politecnico di Torino, Corso Duca degli Abbruzzi 24, Turin 10129, Italy
{paolo.bernardi,andrea_guerriero,ernesto.sanchez,federico.venini}@polito.it
[2] LIRMM, Rue Ada 161, Montpellier 34095, France
{bosio,dinatale}@lirmm.fr

Abstract. At the end of the manufacturing cycle of digital circuits, a stress phase is mandatory in order to remove from the final device population the weak devices that may result in early life failures. Devices used in safety critical environments must undergo this phase that is usually accomplished by exploiting the Burn-In (BI) process. Unfortunately, BI has elevated costs for companies and current state of the art techniques are trying to reduce its cost.

In recent days, Faster-than-at-Speed-Test (FAST) has become a useful technique to discover small delay defects. At the same time, overclocking methods to enhance system performances have been studied, which focus on temperature management to preserve system functionalities. In this contribution, a FAST technique is proposed with the aim of intentionally provoking a thermal overheating in the microprocessor by mean of the execution of FAST functional test programs; in other words, functional procedures are executed at higher than nominal frequencies. The goal is to introduce an internal stress stronger than current procedures used during BI in order to speed up early detection of latent faults.

Being the functional stress procedures executed at faster than nominal speed, the original behaviour may not be preserved and therefore non-functional states may be reached. In this contribution, it is illustrated how to avoid blocking configurations due to timing constraints violation and how to obtain a significant increase of the switching activity by carefully increasing the clock frequency. Furthermore, a novel strategy is proposed to generate a suitable set of Faster-than-At-Speed stress programs capable to thoroughly stress processor cores.

Experimental results carried out on a MIPS-like architecture show major achievements of the methodology: the processor may work at frequencies up to about 20 times higher than the nominal one without falling into an unpredictable state and the switching activity is increasing up to 300% per ns.

Keywords: Faster-than-At-Speed-Test · Functional test · Stress test · Switching activity · Evolutionary algorithm

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 130–151, 2017.
DOI: 10.1007/978-3-319-67104-8_7

1 Introduction

The reduced scale, predicted years ago by the Moore's law, and the low costs of Integrated Circuits (IC) have been the principal factors of spreading high performance microprocessor devices in mass products and safety critical applications. At the same time, the increasing density of transistors on System-on-Chip (SoC), as a subsequence of their smaller size, and the high frequencies they work, have leaded these products to vulnerability to faults and defects, not only during manufacturing processes but also along with the device lifetime.

To guarantee circuit robustness, these products are subjected to several stress phases aiming at bringing the circuit to stressful conditions oriented to detect and discard weak devices. In particular, the so-called Burn-In (BI) test process aims at producing a degradation similar to that introduced by the normal operation during a long period of time (i.e., aging). It applies an external thermal overheating to the device, that in some cases also includes the application of supply voltages higher than the nominal ones. Thermal-based accelerated aging is produced in order to exacerbate time-related degradation defects such as electro-migration affecting the circuit metal lines, leakage current increasing, and variation of threshold voltage in the gates belonging to the design [1]. BI process is necessary for those devices that have to carry out safety critical applications, and for non-repairable systems. By applying a Burn-In phase at the end of manufacturing, it is expected that fewer defective components are delivered and the consequent reduction in failure rates lowers production costs [2]. However, BI is usually associated with long test time and costs, aspects that make it a bottleneck of the entire IC manufacturing process.

In the last years, some approaches tried to apply stress patterns resorting to functional approaches. Functional test programs verify the system integrity for functional and performance-related specifications. It may test all or most of the system functionalities along with the availability of subsystems. In [1, 3], for example, the authors demonstrated that it is possible to apply electrical stress on a given device resorting to functional-based stress programs.

It is clear that a relationship exists between the circuit switching activity (SA) and the temperature developed on the surface of the die [1]. As a matter of fact, it is reasonable to assume the possibility to induce a thermal-aware stress by modulating the switching activity of the target device.

In this contribution, we investigate the possibility of running functional programs at a frequency higher than at-speed with the intention of increasing the switching activity of the system. Once the research claims that processor can continue working even at higher frequencies than nominal thanks to system architecture features, a generic methodology to create functional stress programs targeting pipelined processor cores is investigated. The introduced approach presents the rules to estimate a set of frequency ranges in which the processor is not falling into a blocking state and, by means of these rules, it is possible to generate meaningful functional stress programs. Initially programs are generated targeting every relevant module. Then, some positive side effects on the rest of the modules are measured and a new generation campaign is performed on the missing modules. This process iterates until satisfactory results are reached.

In order to guide the generation process, the method computes a set of elements that helps to assess the quality of each stress functional program in terms of spatiality (i.e., how well the switching activity is spread across the die surface), and intensity (i.e., how high the switching activity is among a given number of gates). Interestingly, the proposed approach is also able to automatically setup the processor frequency in order to find the most appropriate faster-than-at-speed stress programs. This generation feature allows the creation of functional stress programs operating at frequency ranges higher than the nominal ones, avoiding to incur in a blocking state of the processor. The chapter finally presents valid ranges of faster-than-at-speed frequencies in terms of switching activity increase for a functional program, and then it provides a comparison between at-speed and faster-than-at-speed functional stress programs automatically generated.

The rest of the chapter is organized as follows: Sect. 2 provides the framework background, Sect. 3 describes the proposed approach both for discriminate the processor reaction to FAST, both for describe the automatic stress program generation process, while Sect. 4 introduce the case of study and show the experimental results carried out on a MIPS-like processor core synthetized with an industrial library. Finally, Sect. 5 concludes the chapter drawing some future works.

2 Background

Regarding new stress techniques, recent studies are beginning to consider the execution of patterns with a clock frequency higher than the nominal at-speed. This technique, known as faster-than-at-speed test (FAST), finds satisfying results when applied for detecting small delay defects (SDD) which are not easy to screen out with at-speed test [4, 5].

Several application methods targeting FAST technique have been proposed so far in the literature: external applications [5, 6], which are not trivial due to skews and other physical effects that could affect the measurements, therefore, this method requires a more expensive automatic test equipment (ATE). On the other hand, built-in FAST using programmable on-chip clock generation is comfortable with these issues [6, 7], as frequency increase is functionally obtained (e.g., executing assembly instructions). Current works concentrate their efforts on classifying faults according to the optimal frequency they can be tested, or on finding an optimized selection of clock frequencies [4, 8]; usually the maximum clock rate value may be up to 3 times higher than the nominal clock rate [4, 9, 10].

Considering the generation of functional stress patterns for processor cores, an interesting solution can be based on the so-called Software-Based Self-Test (SBST) techniques. The main idea on SBST is to create a functional program intended to detect the processor faults [11]. Roughly speaking, the processor core executes a program allocated in an available memory zone. The computed results are evaluated in order to determine whether the processor is faulty or not. Interestingly, the processor operates in normal mode, thus, there are no requirements for any hardware modification. Nevertheless, SBST methodologies have a limited application in industry due to the difficulty

to write efficient and effective test programs and to devise suitable methodologies for test application.

To the best of our knowledge, this study is the first approach to Faster-Than-at-Speed Test technique while functional programs are running in a SoC. In this contribution, the objective does not concern the coverage of SDD, in contrast with the works listed before, but rather the switching activity increase, which is useful for stress purposes.

3 Proposed Approach

The main objective of this work is to provide detailed understanding about CPU frequency overclocking of a SoC targeting switching activity increase. As already stated in the introduction, increasing the switching activity is beneficial in terms of stress capability and thermal objectives in test. The basic idea of our approach is to analyse the circuit transitions when a functional test program is running, spanning several clock frequencies starting from the nominal one, until reaching very high values. For this reason, in an overclocked situation, it is crucial to understand if:

- The microprocessor can produce *stable* results and keep some functionalities, although some design timing constraints are violated. In this case, we expect a dramatic increase of the internal temperature, thus a further acceleration of aging. Later, this situation is named *functional state.*
- The microprocessor is *stuck* to some unpredictable state, similarly to a "forced functional idle", that leads in a situation in which there will be no gain in terms of temperature exacerbation. Later, this situation is called *unstable state.*

Functional state is highly desirable, while unstable state need to be avoided, in case the induction of a strong stress is the objective [12].

In the following paragraphs, an analysis flow is proposed aiming at determining microprocessor behaviour, functional or unstable, at frequencies higher than the nominal one. Thanks to this methodology, it is possible to identify the causes of the *unstable state*, which will be discussed in details; specific cases and processor configurations need to be avoided in order to maintain a functional state even at very high frequencies.

Once this analysis is complete, purpose becomes to provide an automatic technique able to generate a suitable set of stress programs for processor cores. In fact, the approach creates faster-than-at-speed functional test programs maximizing the switching activity (SA) for the whole processor core. One of the most relevant drawbacks of SBST-based strategies is related to the high amount of resources required to generate test programs. As detailed in [11], most of the generation processes are time consuming and/or require expert engineers able to write the test programs. Thus, the proposed strategy exploits the possibility to generate stress programs for different processor modules in parallel, taking advantages of the positive side-effects that may appear on different processor modules when targeting a particular one. The proposed methodology will be discussed in the following subsections.

3.1 Analysis Flow for Understanding SoC Behaviour

The first objective of our work is to understand when the processor under analysis is falling in an unstable state or not at a certain frequency. To achieve this important understanding, a tool chain setup was devised, enabling the classification of several behaviours.

When overclocking and setting up a clock period tighter than the nominal, unknown (*X*) values can appear in the simulation. These values are caused by the violation of timing constraints of Flip-Flops (FFs) with respect to critical path timing. Figure 1 illustrates an explanatory scenario where the SoC is entering into an unstable state.

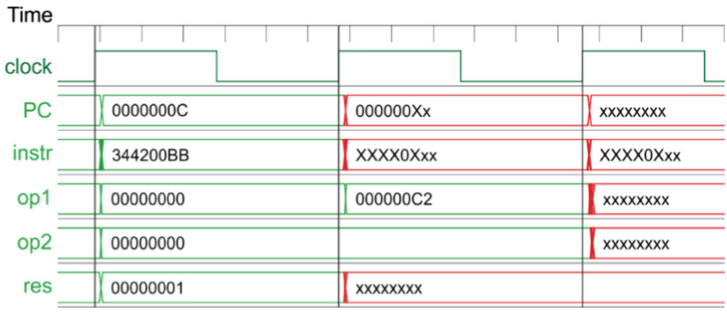


Fig. 1. Some of the CPU registers at the beginning of *unstable state*.

In some cases, the unknown values are bounded in a set of FFs, possibly not compromising the execution of the program. On the contrary, if the unknown values affect a huge number of FFs, CPU reaches an unpredictable state.

The consequences are showed in Fig. 1: when timing constraints of FFs are violated, registers will contain unknown values which can quickly propagate through others (the propagation will depend on the hardware connection between modules). In the showed example, Program Counter (PC) has been affected. Therefore, CPU will never move to execute the next instruction, and it will be stuck into an idle-like state. The affection of instruction-to-decode register (*instr*), or others, has similar effects.

Figure 2 visualizes the practical cause of the X values generation. If a faster-than-at-speed clock is used, the propagation along path may not complete even without a delay fault affecting the circuit. In this case, FFs will not sample a stable value, resulting in an unknown output.

Figure 3 depicts the analysis flow to detect unstable behaviour, which was devised and implemented as a tool chain including both commercial EDA tools and ad-hoc tools. The main idea is to look at the simulation and figure out whether the simulation converges on an unstable state. Furthermore, a loop is implemented where the frequency is increased every time the current analysis is concluded. The logic simulator is fed with critical paths that are extracted by a Static Timing Analysis (STA) tool. The simulator is required to dump the state of the critical paths along the simulation time.

This selective simulation dump is finally provided to an ad-hoc tool able to return the following information from the simulation dump:

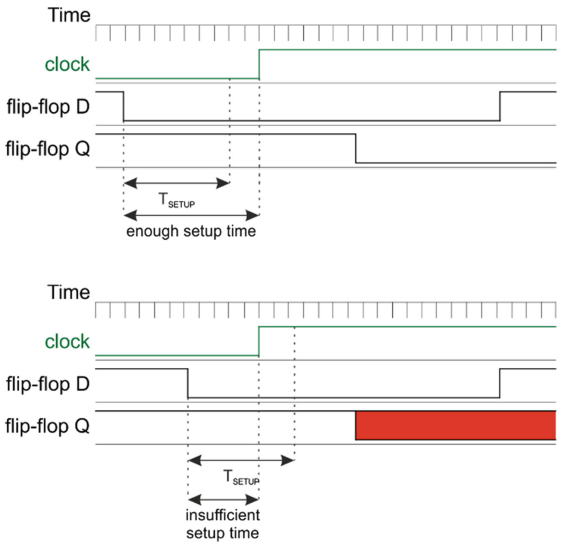


Fig. 2. Violation of setup time of FF (in the lower graph), compared to a good sampling.

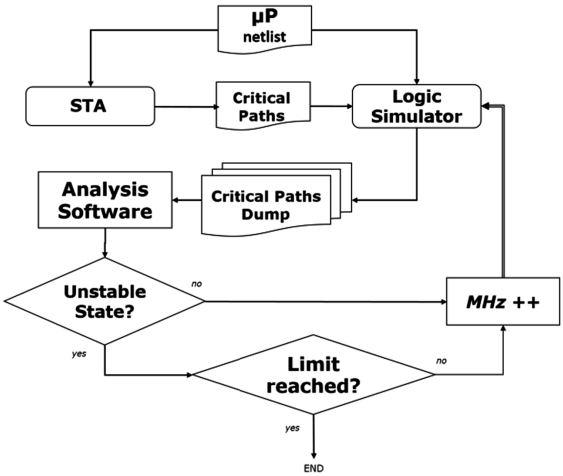


Fig. 3. Proposed analysis workflow for detecting *unstable states*.

- The program *execution time*, by simply counting the number of clock cycles resulted in the simulation;
- The *switching activity* computed as the number of transitions resulted in the gates normalized per ns.

Execution time is enough to identify an unstable state of the processor. In the simulation setup, a timeout time is set and several traps are implemented in order to stop the simulation and classify it as functional; these termination conditions are

showing specific "footprints" left by the processor when correctly running a program. As far as we saw, when the gates of the processor start to show unknown behaviours it is highly improbable that it can reach one of the termination conditions. In this case, test program will be forced to terminate by the timeout time set by the simulator.

In case the simulation does not fall in an unstable state, the process is repeated at a higher frequency. The implemented toolchain permits to decide which step to use along the loop, and automatically adjust sampling time according to the new frequency. In case the simulation falls in the unstable state, a simple formula, detailed in next subsection, permits to decide if continue in the frequency increase or stop the iterations.

3.2 Processor Level Analysis and Classification of Functional States

The common perception is that if a circuit is supplied with a frequency higher than nominal, it will not work. This is an incomplete analysis of the problem as falling in the unstable state may be prevented by some processor architectural features. The first one is the mechanism of pipeline stalls and memory wait-states. They are needed as the memory cores in SoC are usually accessed asynchronously. As a matter of fact, as system stalls because the memory is slow and asks for more time, signals travelling even through the longest paths can reach stable values. Hence, even if another clock period is started, stalled units will not ask for values until the reactivation of pipeline. Figure 4 illustrates the situation: the pipeline is stalled by *stop_all* signal in case the *ram_ready* from the memory goes low too late and it is not sampled. The *ram_ready* signal falling time depends on the latency of the memory. In this example, *ram_ready* is correctly sampled after few clock cycles of pipeline stall as the latency of the memory was set to 5 ns; as well this is feasible because the latency is larger than the propagation time through the most critical path, 4.5 ns in the shown example. The result of the previous ALU operation, such as stored in *EX_data* register, will be read only after the reactivation of CPU pipeline, certainly showing correct values.

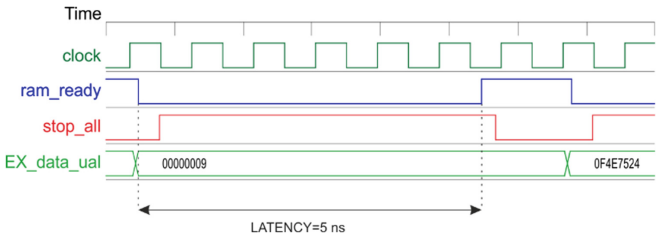


Fig. 4. Typical pipeline stalls during memory latency.

As experimental results proven, this is a lucky configuration that happens only within some ranges of clock frequency. A general very strong rule that can be deduced by the proposed analysis is the following: If the clock period is approximately equal or less than latency, and the latency is larger than the propagation time of functional critical paths, the unstable state depends only on propagation times of asynchronous

data-paths. Usually, these data-paths belong to processor boundary (such paths interfacing RAM signals to CPU, such as *ram_ready*, again) or internal CPU units (such as Prediction Unit), which are not synchronized with the pipeline. In most of architectures, propagation times of asynchronous data-paths are usually less than for processor logic synchronously controlled by the clock. As a consequence, asynchronous data-paths may sustain a certain increase of the clock frequency without incurring into unpleasable events. But, as we are pulling up the frequency much higher than nominal, asynchronous data-paths take a significant role in the study. Figure 5 graphically visualizes the concept of asynchronous data-path; two situations can be identified, with incoming and out-coming paths.

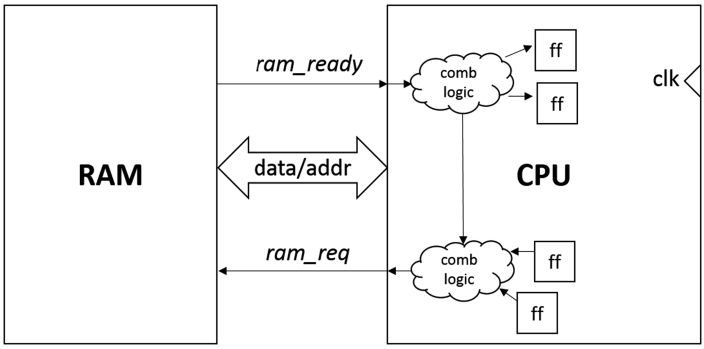


Fig. 5. Example of connections between CPU and companion modules such as memory cores.

Let us consider the *ram_ready* and *ram_req* signals as a reference to clarify the concept. Figure 6 details the acknowledge protocol and relevant times. Once the RAM is rising the *ram_ready* signal, the *ram_req* is falling and again rising after a fixed time T_{REQ} elapsed from the current clock rising edge. Thus, *ram_ready* is falling after a fixed time $T_{\overline{READY}}$, corresponding to the beginning of the next latency time. The time from the falling transition of *ram_ready* to the first clock rising after sampling is called T_{FIRST_RISE}. As well, the time passing from the last rising of the clock before the *ram_ready* is rising is called T_{LAST_RISE}. Most importantly, the time passing from *ram_ready* rising and clock rising edges, namely T_{RES}, is key to understand whether the processor will fall into an unstable state or not. This time should be large enough both to permit propagation along the asynchronous incoming data paths, namely $T_{ASYNCRONOUS\_PROPAGATION}$, and to afford a signal stability time long enough to respect the setup time T_{SETUP}, as preliminarily described in this section. This scenario is depicted in Fig. 7, where T_{DES} is the minimum amount of time that guarantee a correct behavior; in opposition, T_{RES} is the real measurement of time elapsed from the *ram_ready* rising to the following rising edge of the clock signal. T_{DES} can be estimated by simply analyzing the circuital configuration in order to identify propagation cones of asynchronous data paths:

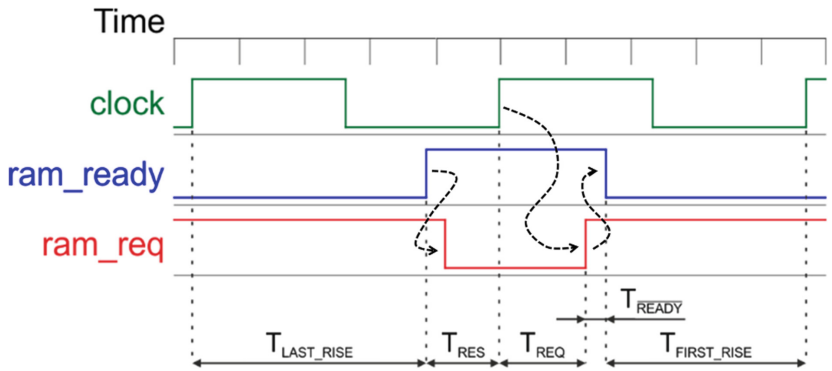


Fig. 6. BUS timing diagram with *ram_ready* and *ram_req*.

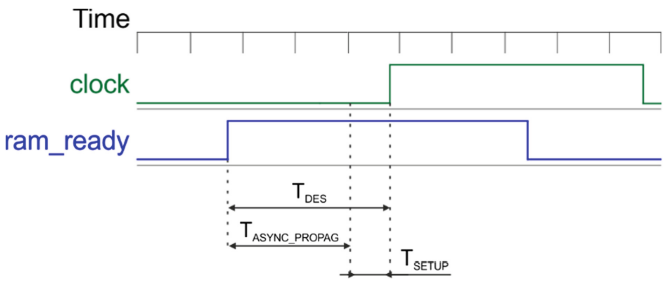


Fig. 7. Timing diagram showing stability requirements for *ram_ready* correct sampling.

$$T_{DES} = T_{ASYNCHRONOUS\_PROPAGATION} + T_{SETUP} \tag{1}$$

where $T_{ASYNCHRONOUS\ PROPAGATION} >> T_{SETUP}$.

T_{RES} can be calculated by adding information related the selected frequency; period in formulas refers to the clock period. To obtain T_{RES}, T_{FIRST_RISE} and T_{LAST_RISE} are needed.

$$T_{FIRST\ RISE} = period - \left(T_{REQ} + T_{\overline{READY}}\right) \tag{2}$$

$$T_{LAST\ RISE} = (latency - T_{FIRST\ RISE})\%period \tag{3}$$

$$T_{RES} = period - T_{LAST\ RISE} \tag{4}$$

At this point, with T_{RES} calculated, it is possible to compare it with T_{DES} for an estimation of the system behavior under higher than at speed frequencies:

$$\begin{aligned} &if\ T_{RES} > T_{DES} \rightarrow FUNCTIONAL\ STATE \\ &if\ T_{RES} < T_{DES} \end{aligned} \tag{5}$$

$$\begin{aligned} &T_{RES} > T_{ASYNCHRONOUS\ PROPAGATION} \rightarrow UNSTABLE\ STATE \\ &if\ T_{RES} < T_{DES} \end{aligned} \tag{6}$$

$$T_{RES} < T_{ASYNCHRONOUS\ PROPAGATION} \rightarrow FUNCTIONAL\ STATE \tag{7}$$

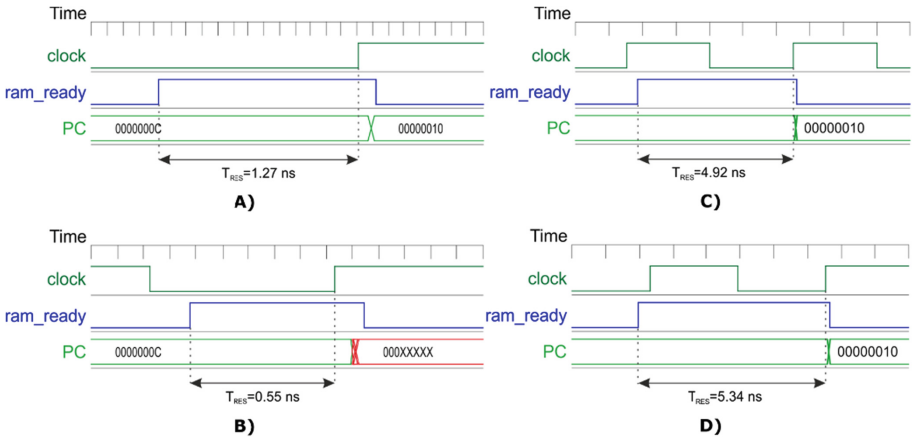


Fig. 8. Dependency on asynchronous path.

Let us observe some examples. In the first case, Fig. 8A, the *ram_ready* transition arrives with a sufficient time to permit propagation and setup, and program counter is not affected Program Counter (PC) register. Frequency there is quite high, 156 MHz.

On the contrary, according to (6) Fig. 8B shows a wrong sampling scenario, caused by higher frequency, 704 MHz. The rising edge of clock is arriving enough early to have propagation of the asynchronous signals up to FFs, but not enough for setup. The wrong sampling will affect PC register since the next clock cycle.

Figure 8C permits a simple understanding of the reason why the processor stops working for a range of addresses and resume for higher frequencies. The clock is quite high in this example, 190 MHz. Even if the clock period is comparable with T_{DES}, Eq. (5) can be satisfied if the *ram_ready* signal is in a favorable phase with clock, resulting in enough T_{RES}.

Last of the cases, Fig. 8D shows that the period can be even smaller than T_{DES}. In this case, the frequency is 198 MHz and the specific case (7) is satisfied because from *ram_ready* to clock rising there is not enough time even to complete the propagation of

asynchronous paths to the FFs. The old value is captured and the true value is latched at the next clock rising edge. This is essentially similar to have a latency overhead of 1 clock cycle, but not compromising the system functionalities.

In the analysis script-chain, the limit of clock frequency value is reached when:

$$Period < T_{ASYNCHRONOUS\ PROPAGATION} + T_{SETUP} \tag{8}$$

3.3 Proposed Methodology for Generating Stress Programs

Figure 9 sketches the main steps for automatically generate functional stress programs targeting processor cores. The first step of generation consists on the *Processor division*. The idea here is to apply a *divide et impera* strategy that allows the next generation steps to be executed in parallel. Such a division is performed following the internal processor hierarchy. Then, the processor is divided into its more important sub-modules that would be tackled in the following steps in a separate way.

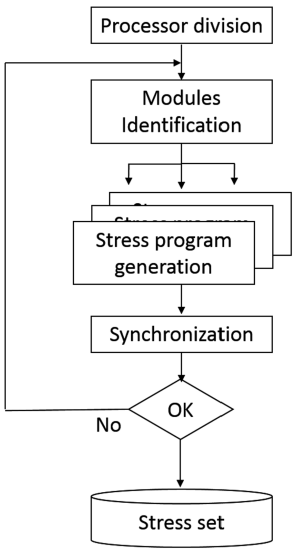


Fig. 9. Stress program generation flow diagram.

It is important to consider that, since the main goal during the generation process is to obtain stress programs maximizing the SA on every processor module, the processor division step must produce not only a hierarchical modules division, but also should be able to distribute all the processor gates on every one of the modules belonging to the processor core. Finally, the most relevant instructions belonging to the processor Instruction Set Architecture (ISA) must be correlated with the derived processor sub-modules.

The second step is called *Module Identification and setup*. This is the first step composing the methodology iterative loop. Here the processor sub-modules to be targeted during each iteration are selected. The idea is to identify and select at the very beginning the functional units of the processor, then, in a successive step, the control parts. As mentioned before, the previous step provides not only the structural division of the processor core but also the set of possible instructions that are able to excite every one of the processor sub-modules. The idea behind this consideration is that the processor functional units are usually independent on each other. It means that each of them performs a task that does not require the intervention of other functional modules. For example, we can consider the processor arithmetic module and the multiplier. Then, a stress program may try to thoroughly stress one specific module without targeting a different one at the same time. On the contrary, most of the processor modules related to the control are highly correlated to the rest of the processor. Thus, these modules should be tackled in a successive step in order to take advantages of the positive effects produced by the stress programs of the already targeted modules. The *Module Identification and setup* step is also in charge to check the list of gates belonging to the considered module in order to discard from the list the gates that at this point are satisfactory stressed. Clearly, during the first iteration no one gate is discarded.

Once the different modules to be tackled for the current iteration have been identified, the generation process can be parallelized. This step is called *Stress program generation* and can use different strategies, for example deterministic, manual, formal-based, and evolutionary-based strategies [11]. A suitable stress program is characterized by the high switching activity on the targeted module involved gates. Even though the SA is one of the most relevant, some other parameters require to be considered. In the proposed case, we define a parameter related to the stress quality for every gate (G_{SQ}), that is given by:

$$G_{SQ} = SA \cdot \frac{1}{d} \cdot Th \cdot g_{CH} \tag{9}$$

where *SA* is the actual gate switching activity, *d* represents the physical distance of the gate to the centre of the module, considering the actual placement of the gate in the final device, *Th* is the thermal factor related to the chip structure, and g_{ch} is a parameter related only to the gate electrical characteristics (fan out, fan in). Please note that all the parameters except the SA are constants, and are related to the gate type and position in the considered module. Thus, during our experiments the target is to increase the SA on every gate.

During the generation process, it is important to provide the selected tool with all the module information required to carry out the generation process. In particular, the tool receives the module gates, as well as the instructions exciting the module itself. In our proposal, we used an evolutionary optimizer called μGP (MicroGP). The tool is developed by Politecnico di Torino since 2000, and is freely available under the GNU Public License [13]. μGP is able to generate syntactically correct assembly programs by acquiring information about the ISA of the processor, and in particular about every one of the modules under consideration from a user-defined file called *Constraint*

Library. It is important to highlight that the evolutionary optimizer defines not only the different instructions composing every program but also the operation frequency. This provides the generation process to create programs that run faster than the nominal frequencies. The faster-than-at-speed execution of these functional test programs permits to reach a grade of stress quality of the circuit in a shorter time interval with respect to the at-speed execution of same test programs. However, as shown in [11], due to frequencies higher than the nominal ones, the processor core may fall in an unstable state. Then, during the test program evaluation, it is important to assure that regardless the operating frequency, the processor behavior is still stable.

When the generation process starts, the evolutionary tool generates an initial set of random assembly programs (called population), or *individuals*, exploiting the information provided by the library of constraint. Every individual also provides the operation frequency as a parameter defined by the evolutionary core. Then, these stress programs are cultivated in order to be improved following the Darwinian concepts of natural evolution. Every individual is evaluated resorting to external tools that simulate the processor by running the stress program at the given frequency and information about the SA in the consider module is gathered. Additionally, processor stability is also considered. In the performed experiments, we compute the average of the switching activity on all the gates composing the module under evaluation. This value is provided to the evolutionary tool as an indicator of the stress program goodness, this is usually called *fitness value*. Once all the individuals are evaluated, resorting to the fitness values, the individuals are ordered and the best ones are maintained in the population, while the others are discharged from the population. Then, a new evolutionary step starts again trying to improve the remaining individuals in the population.

The last step belonging to the iteration process consists on the *synchronization* of the stress programs obtained until this point. As mentioned before, there are independent generation processes for every processor module, however, until now there are no information regarding the impact of the generated programs in the rest of the processor core. Then, it is important to perform a new simulation that involves the full processor core during the execution of all the programs obtained in the current iteration. At the end of this step, a complete figure about the stress capacity of the temporary set of programs in all the processor modules is obtained. Then, the obtained values are compared to the global satisfactory stress ones, and if the results are not in the satisfactory range, the iteration starts again. Otherwise, the process terminates providing the user with the final set of stress programs.

4 Experimental Results

To demonstrate the feasibility and effectiveness of the proposed methodology, two different experiments have been carried out on a *MIPS-like* core [9] synthesized with ST-corelibrary for 65 nm CPU. After the synthesis, the netlist is composed by 50,000 gates circa, which 3,000 are FFs. The *MIPS-like* processor is based on MIPS architecture [10] and contains a 5-stages pipeline which includes hazard detection and pipeline interlock, branch prediction unit and system co-processor. The processor ISA

is composed of 52 instructions, including arithmetic, logic, load and store, and branch related ones. The nominal frequency for normal mode operation is 100 MHz.

This section provides two sets of experimental results: the first one is related to the analysis of the execution of generic FAST functional programs, the second one is related to the automatic generation of FAST functional stress programs.

4.1 Execution of Functional Programs

By mean of scripts, it has been possible to iterate the execution of a functional program increasing CPU clock frequency by 0, 5 MHz for each iteration. *ModelSim* [13] and *Primetime* [14] have been the commercial software used for logic simulation and extraction of critical paths respectively. The delays of single gates are listed in the Standard Delay Format (SDF) file. RAM has been written in System-C language. The software proposed collects data into CSV files for each clock frequency tested.

Several types of functional test programs have been experimented with this framework. Here, we report two cases of programs that are significant in our view:

- Oriented to arithmetic units;
- Stressing branch conditions.

In Fig. 10a and b, execution time is shown at growing frequencies. Blue points correspond to functional states, red to unstable states. It can be noticed that execution time regrows in some cases, passing from unstable to functional state, as the effect of

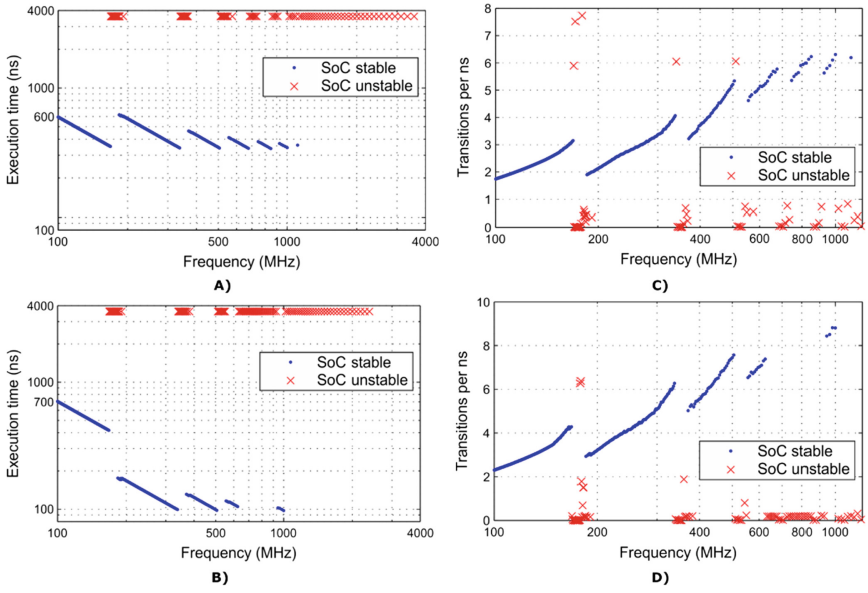


Fig. 10. Execution time of (A) arithmetic units and (B) branches; switching activity of (C) arithmetic units and (D) branches. (Color figure online)

the additional waiting clock cycles inserted as described in Sect. 3.1, Formula (7). Functional states are identified by means of simulation, as in the loop proposed in Sect. 3.1; the processor is expected to store, in a specific RAM address, the signature computed along program execution. The simulation environment is stopped as soon as both these conditions (e.g., correct address and correct signature) are matched, independently on the arrival time that can vary because of the pipeline stalls induced. If the execution time reached a timeout, it will be considered unstable.

Figure 10c and d shows results in terms of switching activity. In this case, solid transitions are completely considered and counted, transitions from *X* value to 0 or 1, or from 1 or 0 to *X* are only considered for the 50% of their weight. In fact, it is quite usual in unstable states that the *X* values are overwritten with solid ones and vice versa. Switching activity value is normalized to 1 ns, meaning that the reported values correspond to the overall switching activity of the program over its length. For example, at

Table 1. Analysis of functional programs results.

	Max switching activity [toggle/ns]	Increase from nominal [%]	Frequency [GHz]
Arithmetic	6.31	361	1.11 GHz
Branch	8.82	383	1.00 GHz

nominal speed (STA reported about 100 MHz) the switching activity is 1.75 per ns, which is average for all observed gates.

Table 1 shows the best increase in terms of switching activity per time unit. The maximum working frequency is about 1 GHz, which is around 10x of the nominal frequency. In this case the increase in the switching activity value is significant. It is also interesting to note that even at lower frequencies (i.e., within 400 and 500 MHz) the increase in the switching activity is already relevant as up to 6 toggle/ns.

A careful analysis of the switching activity result highlights some frequencies driving to unstable states, but with a large count of transitions per nanoseconds. In this case, the circuit is showing a kind of intermediate behaviour due to some specific timing violations. There are several asynchronous paths that are involved in the management of the memory operations; the longest in our case is relative to data bus management, and shows a propagation time of 0.92 ns, while the longest path involved to manage the code bus is slightly shorter, 0.78 ns. At some frequency, the data bus path timing requirements are violated, but not those related to code bus as it is less critical. In this case, the processor is only partially showing an unstable state, as *X* values are propagated only in a part of the synchronous circuits; switching activity value is mainly conditioned by a larger count of transitions form *X* to solid value and vice versa.

Table 2. Processor modules description and correlated instructions counting.

Processor module		# of Gates	# of Instructions
1	Adder	263	2
2	Decoder	2,700	52
3	Bus Control	1,172	30
4	Memory control unit	879	30
5	Forwarding unit	1,355	23
6	Prefetch	691	30
7	Fetch	647	30
8	Sys-coprocessor	1,785	4
9	Register Bank	13,858	32
10	Mult_signed	7,704	2
11	Mult_unsigned	7,652	2
12	Execution stage	7,623	33

4.2 Automatic Generation of Functional Stress Programs

Table 2 summarizes the processor division as required in the first step of the proposed methodology. This division was made by splitting the processor hierarchically. In addition, the table also reports the number of instructions used to stress every processor module.

It is interesting to note that functional modules such as the arithmetic adder, as well as the multipliers, involve a small set of independent instructions; on the contrary, control related parts (e.g., the forwarding unit) require a higher number of instructions that are shared with other modules. The set of instructions derived for every module will be used in the following generation steps in the form of constraint libraries.

The generation flow environment was developed using 4 Python scripts that support the implementation of the iteration loop proposed in Fig. 9. The environment allows to set the modules order taken to progressively generate stress programs, the satisfactory stress threshold value, as well as the different parameters required by the generation tool (in this case μGP). Taking into account the experimental results obtained in [12], and considering the stress quality equation provided before, we empirically determine for these experiments a Satisfactory Stress Threshold (SST) able to evaluate the module stress goodness given by the following factor:

$$SST = \frac{1}{3} \cdot \frac{F_{FAST}}{F_n} \tag{10}$$

where F_{FAST} is the overclocked frequency at which the program is executed, and F_n is the nominal frequency of operation.

Thus, for every targeted module, the ratio SA_{RATIO} between the average SA measured at the FAST execution frequency (SA_{FAST}) and the nominal one (SA_n) should comply with the following condition:

$$SA_{RATIO} = \frac{SA_{FAST}}{SA_n} \geq \frac{1}{3}\frac{F_{FAST}}{F_n}. \tag{11}$$

This value is used during the stress program generation process performed for every module, defining in this way a stop condition to be provided to the evolutionary optimizer.

In addition, in order to determine which gates in the module under consideration are successfully stressed, we define a value (called *GSS*) related to the gate SA and the average SA in the module, in the following way:

$$GSS = 0.5\left(SA_{pk} + SA_{avg}\right) \tag{12}$$

where SA_{pk} is the SA peak obtained in the considered module, whereas SA_{avg} is the SA average compute on the gates belonging to the module.

The evolution of stress programs was divided in three different steps to be performed in a row. In the first one the modules labeled in Table 2 from 1–8 were tackled, and then in the second run, the rest of them. At the end of the process, a third independent run was performed tackling the processor gates still considered as too low according to the gate successfully stress parameter. In order to automatically generate the stress programs, 10 different constraint libraries were devised to feed μGP with the appropriate information regarding the module under evaluation. Interestingly, only the

Table 3. Stress programs evolution (Level 1).

	L1							SYNC. 1
Processor module	SA_{avg}	freq [MHz]	SST	SA_{ratio}	SA_{peak}	GSS	ssg	Δssg
Adder	18.65	652.74	2.15	2.07	24.70	21.675	0	233
Decoder	88.17	612.37	2.02	3.16	92.40	90.285	1001	322
Bus Ctrl	128.28	684.46	2.26	4.75	136.2	132.25	695	416
Mem	110.12	676.59	2.23	4.53	119.1	114.64	756	64
Forward	53.34	651.04	2.14	2.29	74.22	63.780	650	386
Prefetch	34.08	512.30	1.69	6.31	49.83	41.955	410	82
Fetch	79.55	595.59	1.96	4.48	83.84	81.695	421	108
Sys-cop	24.46	623.44	2.05	1.75	32.51	28.485	0	570
Reg. Bank	–	–	–	–	–	–	–	–
Mult_sign.	–	–	–	–	–	–	–	–
Mult_unsig.	–	–	–	–	–	–	–	–
Execution stage	–	–	–	–	–	–	–	–

Table 4. Stress programs evolution (Level 2).

	L2							SYNC. 2
Processor module	SA_{avg}	freq [MHz]	SST	SA_{ratio}	SA_{peak}	GSS	ssg	Δssg
Adder	–	–	–	–	–	–	–	–
Decoder	–	–	–	–	–	–	–	–
Bus Ctrl	–	–	–	–	–	–	–	–
Mem	–	–	–	–	–	–	–	–
Forward	–	–	–	–	–	–	–	–
Prefetch	–	–	–	–	–	–	–	–
Fetch	–	–	–	–	–	–	–	–
Sys-cop	–	–	–	–	–	–	–	–
Reg. Bank	25.46	669.79	2.23	2.20	36.11	30.785	0	2,103
Mult_sign.	83.41	674.31	2.25	2.60	90.81	87.110	2,314	320
Mult_unsig.	85.08	677.97	2.26	2.60	91.03	88.055	2,433	287
Execution stage	99.55	677.97	2.26	2.51	107.05	103.300	1,987	1,208

Table 5. Stress programs evolution (Level 3 and total amount).

	SYNC. 3	L3	TOT
Processor module	Δssg	Δssg	ssg
Adder	0	12	**245**
Decoder	785	250	**2,358**
Bus Ctrl	20	3	**1,134**
Mem	3	0	**823**
Forward	12	53	**1,101**
Prefetch	0	9	**501**
Fetch	4	21	**554**
Sys-cop	0	2	**572**
Reg. Bank	5,601	1,007	**8,711**
Mult_sign.	302	208	**3,144**
Mult_unsig.	298	306	**3,324**
Execution stage	2,110	809	**6,114**

ones dedicated to the functional modules were independent, while the others were shared and composed of the most instructions in the processor ISA.

The evolution of the generation process is detailed in Tables 3, 4 and 5. The generation process is developed in 3 different levels. The table shows for every generation level, the average SA_{avg} for the modules under consideration, the FAST fre-

quency of the stress program generated, the SST value, SA_{RATIO} and SA_{peak} measured during the generation process, the *GSS* value for the gates in the module, and in addition, the number of gates successfully stressed (**ssg**) according to the previous parameters.

Furthermore, the table shows for each synchronization step (1 to 3), the additional gates (**Δssg**) successfully stressed by exploiting the positive effects of different programs in the targeted module. Finally, the last column of each Table (3, 4 and 5) shows the final number of gates successfully stressed for every module considering all the previous contributions.

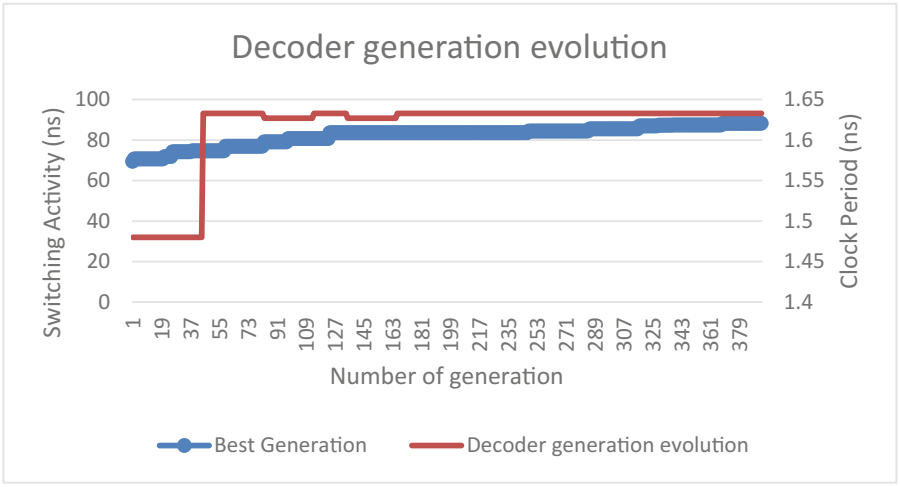
During the first generation level, the initial set of modules (the first 8 in the Table 2) is targeted. Then, the second set of modules. After every generation level, a synchronization step is performed. Then, the third synchronization step evaluates the different programs on all the other modules. The level 3 generation intends to stress the weak gates still present in the processor, and shows again the **Δssg** obtained in this phase. The reader may notice that there is a positive effect obtaining during the synchronization phases on the different modules. This positive effect reduces the generation time, allowing the test engineer to discard some gates for free.

It is interesting to note that in some cases, for example during the adder stress process, the SA_{RATIO} is lower than the *SST* parameter, then, since the condition is not satisfied no gates can be considered as thoroughly stressed. Then, in these cases, it is necessary to consider not only the stress programs developed targeting the module, but all the others. This is the reason why the gates successfully stressed in the adder appear only during the first synchronization step.

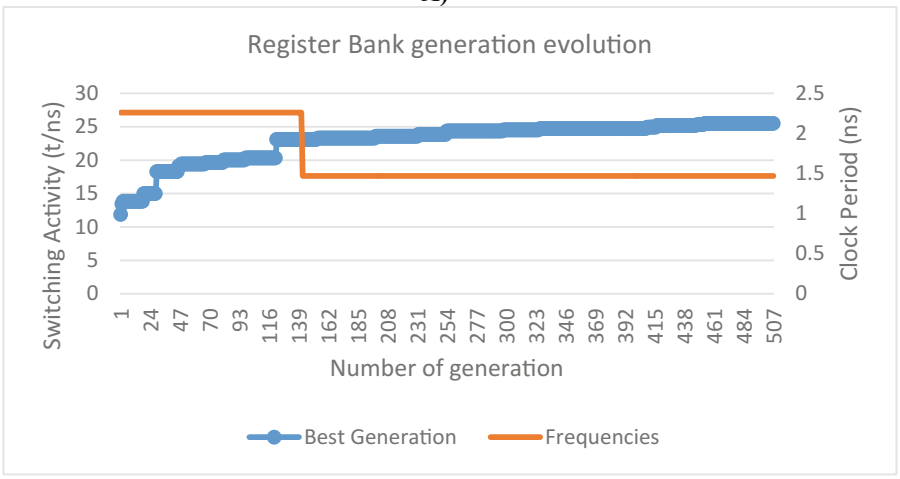
For the purpose of comparison, typical scan values may be considered. From our experience, we state that this structural solution for stress is strongly limited by the tester environment (such as along Burn-In testing). Additionally, scan procedures for stress need an accurate evaluation in terms of power consumption, as they are not functional. Thus, it is quite usual that a low power procedure is finally adopted, which are showing very low switching activity values.

The experiments described before were run in a workstation based on 2 Intel Xeon E5450 CPUs running at 3.00 GHz, equipped with 64 GB of RAM. Every stress generation process requires about 12 h. However, the process is highly parallelizable and it globally requires about 3 working days.

Checking some graphs gathered from the evolution of the stress programs, in particular, Fig. 11a depicts the evolution of the best stress programs for the decoder unit. The figure shows the evolution of the best individuals considering the maximum average SA (transition/ns) and the FAST execution frequency. It is interesting to see how at the very beginning of the evolution process there are some programs that run at



A)



B)

Fig. 11. Stress programs evolution for decoder unit (A) and register banks (B).

high frequencies (about 1.475 ns = 677.9 MHz) and very low SA average (70 transitions/ns), but later, the evolution improves the SA even reducing the FAST frequency.

On the contrary, Fig. 11b shows a different shape evolution where the frequency starts at a very low value (about 2.25 ns = 444.4 MHz), and later the frequency as well as the SA is increased.

5 Conclusions

In this contribution, we proposed a functional approach that can allow the target processor to work at faster-than-at-speed frequencies along with a method to identify which are the frequency ranges in which this technique is applicable. Furthermore, we proposed a novel technique able to automatically generate stress programs for pipelined processors at faster-than-at-speed frequencies exploiting the previous analysis.

Pushing the clock frequency to values higher the nominal is feasible thanks to processor architectural features such as pipeline stalls and memory wait states. The cause of blocking states is mainly attributable to the propagation of signals that violates the design timing constraints resulting in unknown values sampled by FFs. The enhanced switching activity achievable with this technique can be exploited to develop a much more reliable and effective electrical activity on a given device reaching high thermal stress.

For this reason, the analysis is followed by a possible exploitation of a program that can be executed at frequencies higher than the nominal one guaranteeing a high stress capacity. In fact, an automatic generation process for FAST functional programs is developed and provided, in order to reach a high grade of stress for pipelined processor cores. The proposed method splits the processor hierarchically and generates stress programs for every one of the processor modules. Then, the positive side effects of these programs are considered on the rest of the processor, reducing in this way the generation times. The method suitability was experimentally evaluated in a MIPS-like processor core obtaining a set of stress programs containing 13 programs running at frequencies about 6.4 higher than the nominal ones. In addition, from the performed experiments, it is possible to see that more than the 60% of the processor gates were labelled as successfully stressed gates.

The proposed method is a preliminary study carried out on a *MIPS-like* processor as reference architecture in order to verify its feasibility. The next step is to migrate this approach on an industrial device, identifying non-blocking regions in which it is possible to manage the activity of the processor and create a frequency-targeted stress. Moreover, the intrinsic dependency of internal delays to the real die temperature will be further and better investigated on silicon.

References

1. Rosinger, S., Metzdorf, M., Helms, D., Nebel, W.: Behavioral-level thermal - and aging-estimation flow. In: 2011 12th Latin American Test Workshop (LATW), Porto de Galinhas, pp. 1–6 (2011)
2. Kuo, W., Kuo, Y.: Facing the headaches of early failures: a state-of-the-art review of burn-in decisions. Proc. IEEE **71**(11), 1257–1266 (1983)
3. Appello, D., Bernardi, P., Cagliesi, R., Giancarlini, M., Grosso, M., Sanchez, E., Sonza Reorda, M.: Automatic functional stress pattern generation for SoC reliability characterization. In: 14th IEEE European Test Symposium (ETS), Sevilla, pp. 93–98 (2009)

4. Hellebrand, S., Indlekofer, T., Kampmann, M., Kochte, M.A., Liu, C., Wunderlich, H.J.: FAST-BIST: faster-than-at-Speed BIST targeting hidden delay defects. In: 2014 International Test Conference (ITC), Seattle, WA, pp. 1–8 (2014)
5. Yan, H., Singh, A.D.: Experiments in detecting delay faults using multiple higher frequency clocks and results from neighboring die. In: 2003 International Test Conference (ITC), pp. 105–111 (2003)
6. Tayade, R., Abraham, J.A.: On-chip programmable capture for accurate path delay test and characterization. In: 2008 IEEE International Test Conference (ITC), Santa Clara, CA, pp. 1–10 (2008)
7. Pei, S., Li, H., Li, X.: An on-chip clock generation scheme for faster-than-at-speed delay testing. In: 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, pp. 1353–1356 (2010)
8. Kampmann, M., Kochte, M.A., Schneider, E., Indlekofer, T., Hellebrand, S., Wunderlich, H.J.: Optimized selection of frequencies for faster-than-at-speed test. In: 2015 24th IEEE Asian Test Symposium, pp. 109–114 (2015)
9. Ahmed, N., Tehranipoor, M., Jayaram, V.: A novel framework for faster-than-at-speed delay test considering IR-drop effects. In: 2006 IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, pp. 198–203 (2006)
10. Amodeo, M., Cory, B.: Defining faster-than-at-speed delay tests. Cadence Nanometer Test Q. **2**(2), 1–3 (2005)
11. Psarakis, M., Gizopoulos, D., Sanchez, E., Sonza Reorda, M.: Microprocessor software-based self-testing. IEEE Des. Test Comput. **27**(3), 4–19 (2010)
12. Bernardi, P., Bosio, A., Di Natale, G., Guerriero, A., Venini, F.: Faster-than-at-speed execution of functional programs: an experimental analysis. In: IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Tallinn (2016)
13. miniMIPS. Opencores.org (2016). http://opencores.org/project,minimips
14. Primetime. synopsys.com (2016). https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html

Beyond Ideal DVFS Through Ultra-Fine Grain Vdd-Hopping

Valentino Peluso[1], Roberto G. Rizzo[1], Andrea Calimera[1(✉)], Enrico Macii[1], and Massimo Alioto[2]

[1] Department of Control and Computer Engineering, Politecnico di Torino, 10129 Torino, Italy
{valentino.peluso,andrea.calimera}@polito.it
[2] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583, Singapore

Abstract. DVFS is the de-facto standard for low energy Multi-Processor SoCs. It is based on the simple, yet efficient principle of lowering the supply voltage (Vdd) to the minimum threshold that satisfies the frequency constraint (f_{clk}) required by the actual workload. An ideal-DVFS deals with the availability of on-chip high resolution voltage regulators that can deliver the supply voltage with a fine step resolution, a design option that is too costly.

While previous research focused on alternative solutions that can achieve, or at least get close to, the efficiency of ideal-DVFS while using a discrete set of supply voltages, this work introduces *Ultra-Fine Grain Vdd-Hopping* (FINE-VH), a practical methodology that brings DVFS beyond its theoretical limit.

FINE-VH leverages the working principle of Vdd-Hopping applied within-the-core by means of a layout-assisted, level-shifter free, dynamic dual-Vdd control strategy in which leakage currents are minimized through an optimal timing-driven poly-bias assignment procedure. We propose a dedicated back-end flow that guarantees design convergence with minimum area/delay overhead for a cutting-edge industrial Fully-Depleted SOI (FDSOI) CMOS technology at 28 nm.

Experimental results demonstrate FINE-VH allows substantial power savings w.r.t. coarse-grain (i) ideal-DVFS, (ii) Vdd-Hopping, (iii) Vdd-Dithering, when applied on the design of a RISC-V architecture. A quantitative analysis provides an accurate assessment of both savings and overheads while exploring different design options and different voltage settings.

1 Introduction

It is well known that power consumption is the most stringent constraint for the growth of digital System-on-Chips (SoCs) [1]. As an answer to this issue, the

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 152–172, 2017.
DOI: 10.1007/978-3-319-67104-8_8

multi-core/many-core design paradigm was introduced as the only possible way out. Indeed, when high performance need to meet a low energy budget, the availability of multiple processing units that can be turned-ON/OFF, or just slowed down depending on the actual workload, represents an efficient solution. Within this context, Dynamic Voltage Frequency Scaling (DVFS) has been proven to be the most effective technique to get close to minimum energy consumption. DVFS is based on a straightforward working principle, that is, reduce the supply voltage (Vdd) down to the minimum threshold that satisfies the frequency constraint (f_{clk}) imposed by the workload.

Originally applied to "monolithic" SoCs [2], the degree of freedom made available by multi-processor SoCs (MP-SoCs) architectures enabled a more efficient core-based, i.e., fine-grained, DVFS implementation [3]. With a fine-grain DVFS, each core can be set working at a different operating point in the [f_{clk}, Vdd] space; this allows to run multiple tasks asynchronously and bring down the minimum-energy point of the whole SoC. An example of fine-grain DVFS on massively parallel platforms is given in [4], where 167 processors are orchestrated over a wide frequency range achieving minimum power consumptions, from 1.07 GHz–47.5 mW at 1.2 V to 66 MHz–608 µW at 0.675 V. As an add-on feature, fine-grain DVFS is a perfect knob to compensate and/or mitigate variations due to Process, Voltage and Temperature (PVT) fluctuations that affect different cores after fabrication and during the lifetime of the circuit [5].

A practical use of DVFS on MP-SoCs deals with the availability of programmable on-chip Vdd regulators that can deliver the supply voltage with fine resolution step and fast swing (Fig. 1-a). Unfortunately, the use of integrated DC/DC converters is made impractical due to high implementation costs. Indeed, on-chip

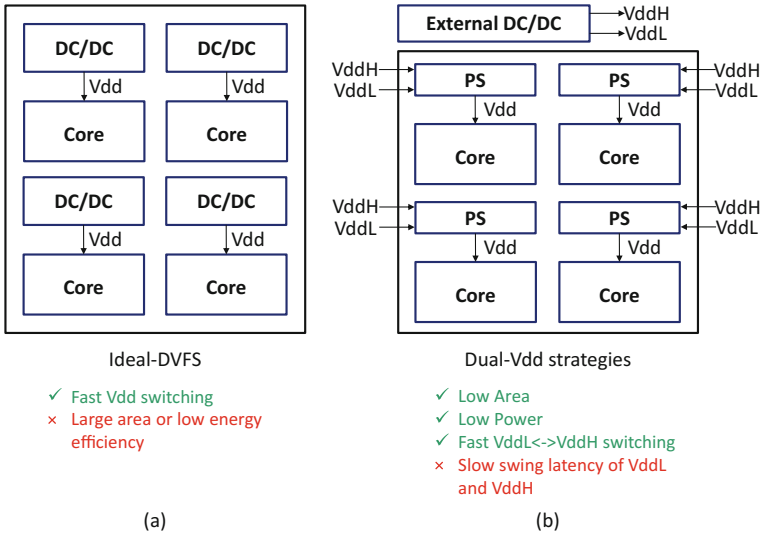


Fig. 1. Comparison between ideal-DVFS and dual-Vdd strategies

DC/DC converters fabricated with today's technologies may occupy a huge silicon area due to the low integration density of the components they contain, e.g., capacitors and inductors [4]. The picture gets even more complicated if one considers that each single core should be equipped with a dedicated converter.

The challenge faced by previous works is to achieve, or at least get close to, the efficiency of high-resolution DVFS, ideal-DVFS hereafter, with a discrete set of supply voltages. In their more general embodiment, discrete DVFS strategies use two Vdd levels (VddL and VddH) generated off-chip through external voltage regulators and evenly distributed across the die (Fig. 1-b). The absolute values of VddL and VddH are shifted up/down depending on the workload, while each core is fed with the proper Vdd by means of dedicated power switches (PS). Even though this design option offers a practical solution with low impact on area and power, it comes with a speed penalty due to high voltage swing latency of the external voltage regulators [6]. Nevertheless, this is an acceptable cost as the voltage scaling process typically applies at low rate.

The two most representative cases of discrete DVFS are the Vdd-Hopping [7] and the Vdd-Dithering [8]. The Vdd-Hopping is a basic scheme in which the supply voltage range is split into a discrete set of values, two ore more depending on the external voltage regulator; the proper Vdd is selected among the available values such that the frequency constraint is met. The Vdd-Dithering scheme is a more elaborated, yet precise scheme that implements a Vdd time-sharing strategy. Differently from Vdd-Hopping, the Vdd is made switching from low (VddL) to high (VddH), leading the core to an average frequency equals to the frequency constraint. As it will be shown later in the text, the power-frequency tradeoff obtained through Vdd-Dithering can be seen as a linear approximation of ideal-DVFS.

While the aforementioned techniques aim at pushing power consumption close to ideal-DVFS, this work proposes a solution that goes beyond that theoretical limit. Recalling the practical implementation described in [9], in this work we give a comprehensive parametric analysis of the main advantages brought by a Vdd-Hopping scheme applied at a ultra-fine granularity, i.e., within-the-core. Such a solution, called *FINE-VH*, represents a viable solution to achieve power consumption below ideal-DVFS by using a dual-Vdd scheme.

The implementation of a FINE-VH solution is not trivial nor even straightforward, especially when the goal is to devise a computer-aided design methodology and not a handcrafted design. Working with multiple voltages within the same functional block raises several concerns during the place & route stages, e.g., area overhead and timing closure due to layout fragmentation and standard cell displacement. Moreover, static power consumption increases due to leakage currents of logic gates driven by portions of the circuit powered at different Vdd. Considering a simple chain of two inverters, when the driven inverter is supplied at nominal Vdd, its static power increases up to 5.2x if the driver is powered at 90%Vdd, 22.1x at 80%Vdd. Notice that the use of voltage-level shifters is strictly forbidden at this level of granularity, as it would imply huge design overheads.

As an answer to these needs, we introduce a fully-integrated design flow that guarantees timing/power convergence through incremental re-synthesis stages. In particular, an optimal poly-bias assignment strategy is used to reduce intra-domain leakage power at zero area/delay penalties.

The core used as benchmark is the *RI5CY*, a RISC-V instruction set architecture embedded in the ultra-low power multi-processor platform *PULP* [10]. The RI5CY core has been mapped onto a cutting-edge Fully-Depleted SOI (FDSOI) technology at 28 nm. We give an accurate design space exploration which quantifies different figures of merit, like power, area and delay, at different granularity, i.e., using a layout partitioned into 9, 25 and 49 tiles, and different voltage set, i.e., multiple values of ΔVdd = VddH − VddL. As will be shown later in the experimental section, FINE-VH gives substantial power savings w.r.t. state-of-the-art DVFS solutions: ideal-DVFS, Vdd-Hopping and Vdd-Dithering.

2 Previous Works

2.1 Approaching Ideal-DVFS

The need to get close to an ideal-DVFS implementation may suggest the use of programmable on-chip DC/DC converters that guarantee a fine Vdd regulation. Unfortunately this design option would imply large overheads, both in terms of area and power. As alternative solutions, Vdd-Hopping [7] and Vdd-Dithering [8] are a preferred option as they enable a good approximation of ideal-DVFS at reasonable implementation costs.

- *Vdd-Hopping:* differently from ideal-DVFS (dashed line in Fig. 2) where the supply voltage can be adjusted with a very high resolution, this method employs a discrete set of supply voltages to control the local Vdd of a functional core. Figure 2-a gives a pictorial description of this principle. The whole supply voltage range is split into a specific set of intervals, three in the plots of Fig. 2; at a run-time, the proper supply voltage is selected in order to meet the target frequency (f_{clk}) imposed at the application level. Once f_{clk} is identified, the core is supplied by the Vdd at the right edge of the interval in which f_{clk} falls (Vdd_2 in Fig. 2-a). Within each interval, the Vdd is kept constant and the power consumptions decrease linearly with f_{clk}. When f_{clk} crosses a new interval, power scales accordingly with the new Vdd. Obviously, the power consumption obtained with Vdd-Hopping drifts from ideal-DVFS as f_{clk} approaches the left side of each interval.
- *Vdd-Dithering*: this scheme, graphically described in Fig. 2-b, implements a Vdd time-sharing scheme. Differently from Vdd-Hopping, the Vdd switches from low to high, i.e., from the left edge to the right edge of the reference interval (Vdd_1 and Vdd_2 in the example of Fig. 2-b). Given T_{high} as the time spent at high Vdd, hence high frequency f_{high}, and T_{low} as the time spent at low Vdd, hence low frequency f_{low}, the core operates at an average frequency (f_{avg}) which is proportional to the "switching ratio":

$$f_{\text{avg}} \propto \frac{(f_{\text{low}} \cdot T_{\text{low}}) + (f_{\text{high}} \cdot T_{\text{high}})}{T_{\text{low}} + T_{\text{high}}}. \tag{1}$$

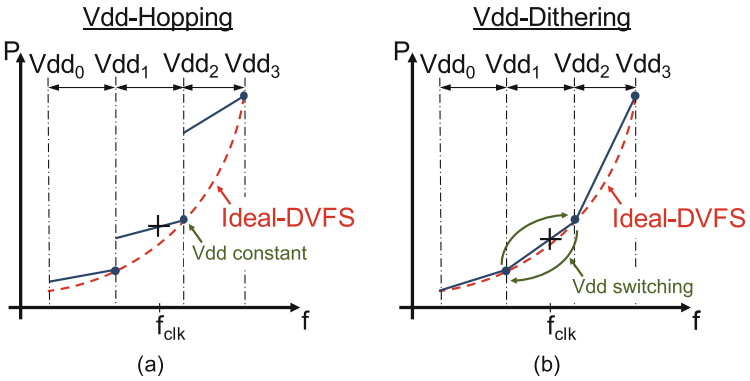


Fig. 2. Frequency-power tradeoff of existing DVFS strategies

Fixing the proper ratio between T_{high} and T_{low} allows the core to operate at an average frequency centered on the target frequency f_{clk}. Apart from some overhead introduced by the non-ideality of power switches, experimental results in [8] have shown physical implementations of Vdd-Dithering well fit the trend line depicted in Fig. 2-b.

2.2 Within the Core Power Management

The target of this work is to bring the Vdd-Hopping concept at a lower level of granularity, i.e., within the core. The basic concept behind this idea is not new as it follows the natural scaling other power-management strategies experienced in the last years, e.g., Multi-Vdd, Body-Biasing, Power-Gating [11–13]. The taxonomy tree shown in Fig. 3 gives a compact classification of the many granularity options.

The low-power techniques were originally applied at the architectural level where the grain was a single *functional block* (Fig. 3-a). Examples of this functional-based strategy are given in [14,15], where the power domains are defined as the boundaries of the micro-architectural units of a microprocessor. In [14] authors introduce a voltage interpolation scheme inspired by Vdd-Dithering where the pipeline stages are dynamically fed by VddH and VddL in a time-shared manner; combined with a variable latency mechanism, this technique offers a viable solution to compensate process-variations while achieving minimum energy consumption. As an extension, the authors of [15] improve the voltage interpolation mechanism by means of a dynamic local error detection & correction scheme for protection against timing violations.

While playing with functional blocks is an intuitive solution, the amount of power savings is limited by the coarse granularity at which the low-power knobs operate. It is well known that all the low-power techniques exploit a natural characteristic of digital circuits, that is, idleness; when a functional block is not used, or it can operate at a lower speed, its power can be reduced without

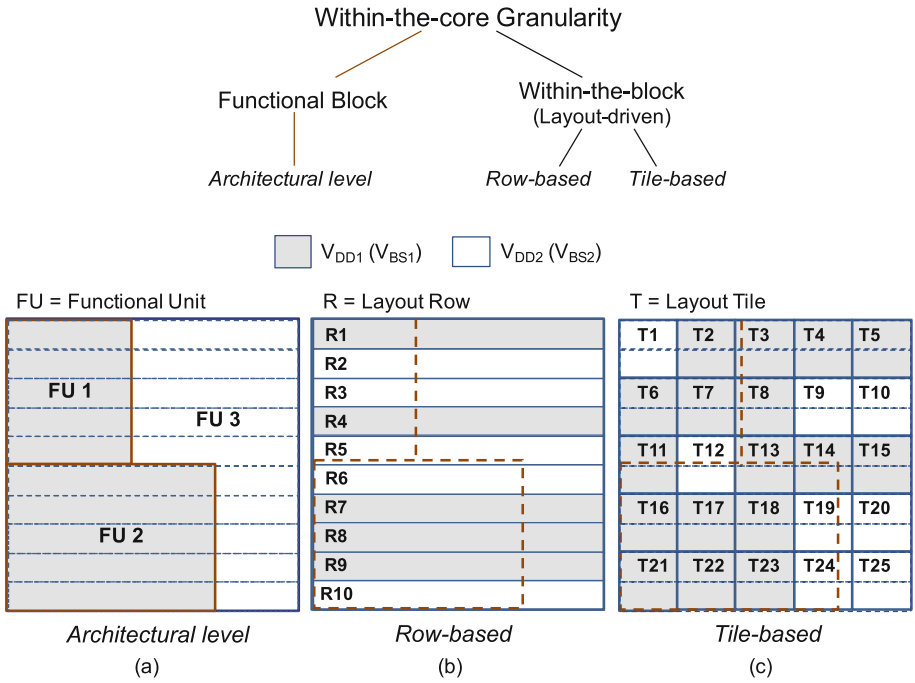


Fig. 3. Taxonomy of low-power knobs granularity. From coarse-grained architectural level (a), up to finer granularity as row-based (b) and tile-based (c) layout organization

affecting performance. This implies the identification of idle functional blocks and active ones. However, also within each single block large portion of the die are taken by non-critical components (e.g., standard cells belonging to fast logic paths) over which low-power knobs could effectively operate without inducing any performance penalty. Following this path, low-power techniques underwent a development process that brought them to work at a finer granularity, i.e., *within-the-block*.

Unfortunately, working at a finer granularity is not a free lunch as strict rules imposed by semi-custom row-based layouts might prevent the physical implementation. This imposes a higher level of awareness of the layout constraints and it forces hard constraints on the geometrical size of the minimum grain. The most common layout-driven solutions are *row-based* and *tile-based*.

As shown in Fig. 3-b, in row-based strategies the atomic element is a single layout row. In [16], authors describe a row-based dual-Vdd technique for PV compensation. The core of this technique is a timing-driven clustering procedure where timing critical rows are assigned to high Vdd and the remaining rows to low Vdd. A customized placement algorithm groups critical cells on adjacent rows such that leakage currents at the row interfaces are minimized. The main limitation of this work is that Vdd assignment is done at design time, i.e., statically. Within the same category, [17] introduces a row-based scheme for ultra-fine

grain body-biasing. Differently from [16], the layout is partitioned into equally sized bunches of rows. Such a structure is more flexible as it enables dynamic body-biasing scheduling for post-silicon process variation compensation.

In tile-based strategies, the atomic element consists of a regular section of the layout; Fig. 3-c reports a scheme where the layout of a logic circuit is arranged in 5×5 square mesh. Within this category, the most representative example is given in [18] where a PV-aware adaptive dual-Vdd strategy is applied on a DES core partitioned into 42 square tiles. The measurements obtained on a silicon test-chip demonstrate power savings are limited to 12% (w.r.t. monolithic DVFS) due to static power overheads induced by intra-tile leakage currents. The same tile-based architecture is adopted in [19], yet with a different goal; it assigns a high Vdd to those tiles containing standard cells whose electrical behavior requires a minimum operating voltage ($\mathrm{Vdd}_{\mathrm{min}}$) larger than the majority of the other cells. This allows fault-free, low power operation even if the overall circuit is powered at $\mathrm{Vdd} < \mathrm{Vdd}_{\mathrm{min}}$.

3 Implementing FINE-VH

3.1 Design and Optimization

As highlighted in the introduction, the main contribution of this work is the implementation of a novel design strategy, FINE-VH, that reshapes the Vdd-Hopping principle at a ultra-fine granularity. For this purpose, we devised a computer-aided design methodology that implements FINE-VH strategy by means of ultra-fine dual-Vdd. Working with multiple voltages within the same functional unit raises several concerns during the place & route stages, e.g., area overhead and timing closure, due to layout fragmentation and standard cell displacement. Moreover, static power consumption increases due to leakage currents of logic gates driven by parts of the circuit powered at different Vdd.

This section firstly introduces the layout organization and the physical design stages that implement the FINE-VH; secondly it describes an optimal poly-bias assignment technique that compensates the intra-tile leakage at no delay penalty; finally it proposes a Simulation/Emulation procedure for optimal Vdd selection.

Physical Design. The proposed FINE-VH strategy resorts to a tile-based structured layout, abstract view given in Fig. 4. The core is regularly partitioned into N × N square tiles ($N = 3$ in Fig. 4), each of them provided with dual-Vdd, i.e., low-Vdd (VddL) and high-Vdd (VddH), taken from around-the-core power-rings. The two power supply voltages are provided by external DC/DC converters and their value is fixed at the application level depending on the target frequency (referring to the example of Fig. 2, $\mathrm{VddL} = \mathrm{Vdd}_1$ and $\mathrm{VddH} = \mathrm{Vdd}_2$). Upper-metal horizontal/vertical stripes run over the core area forming five power-grids: VddH, VddL, Gnd, Vbn (n-bias), Vbp (p-bias). Notice that this scheme is compatible with adaptive back-biasing strategies (out of the scope of this work).

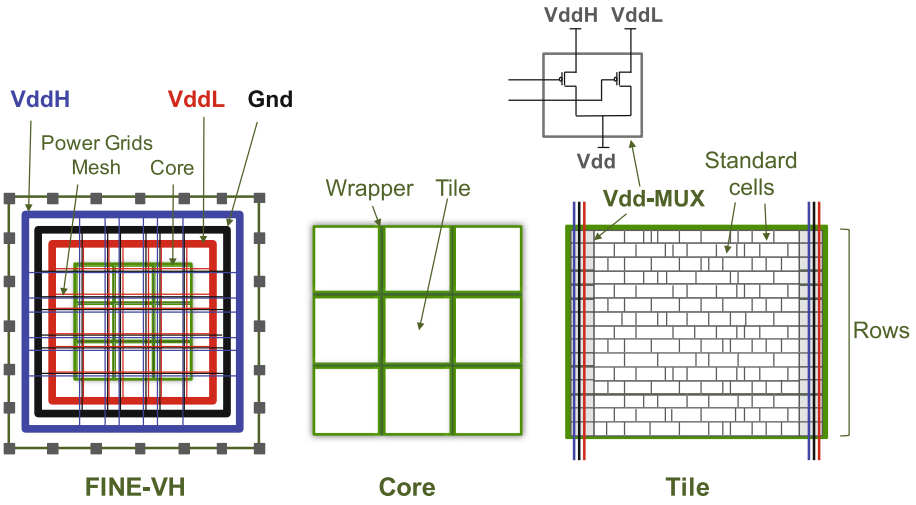


Fig. 4. Layout partitioning and tile organization

The layout rows are tied to the power-grids through p-type header power switches enclosed into dedicated cells, the *Vdd-MUX* cells. The power-management unit is in charge of driving those Vdd-MUXes by loading the Vdd configuration bit-stream into a dedicated flip-flop chain. The Vdd-MUXes are uniformly distributed within each tile following a row-based insertion scheme [20,21]. The power-grids are aligned with the Vdd-MUX columns, hence, VddL and VddH area easily brought to the Vdd-MUX cells using vertical vias.

Tiles are isolated each other by a void-space wrapper that creates discontinuity in the lower-metal power rails. That's mandatory as adjacent tiles might have a different Vdd. The wrapper width is defined by the minimum metal-to-metal distance for the technology in use.

It is worth noticing that the layout partitioning follows a "no-look" style, that is, once the grain size is defined through the parameter N, the tile partitioning is done at the floorplanning stage without considering how and where the sub-functional blocks of the core will be placed. On the one hand, this might seem an overhead as functional blocks are split across multiple tiles. However, that's what allows commercial place & route tools digesting the ultra-fine granularity so as to achieve (i) a regular power planning and (ii) faster timing closure.

From a practical viewpoint, the FINE-VH flow encompasses six different stages fully integrated into a commercial design platform by *Synopsys*® by mean of dedicated TCL procedures:

1. **Synthesis**: logic synthesis using technology libraries characterized at the maximum Vdd, e.g., 1.0 V for our 28 nm technology.
2. **Floorplanning**: estimation of the core area and creation of an empty layout; the latter is then automatically partitioned into N × N regular tiles using placement blockages.

3. **PG-Synthesis**: power-grids are synthesized following a regular mesh over the partitioned layout.
4. **Placement**: the Vdd-MUXes are placed at the boundaries of the tiles while standard cells are placed within the tiles so that timing constraints are satisfied.
5. **Post-Placement leakage optimization**: a re-synthesis stage performing optimal poly-bias assignment for those cells at the interface of the tiles (additional details in the next subsection).
6. **Routing**: a standard timing-driven routing for logic signals.

Poly-Bias Optimization. In a FINE-VH design, static power consumption may increase due to larger leakage currents in the "interface-cells", i.e., cells driven by signals coming from other tiles. When an interface-cell is fed with an input signal having a voltage lower than its Vdd, its internal pull-up network is partially turned-ON and leakage currents increase. This scenario is depicted in Fig. 5-a.

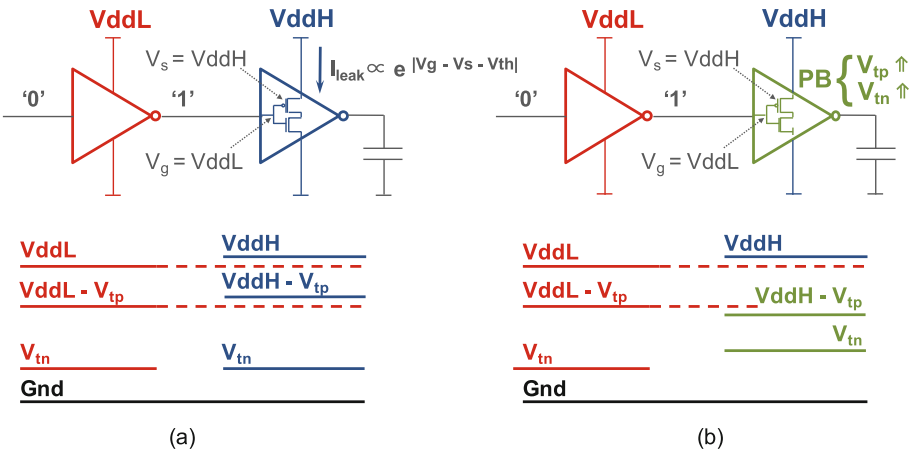


Fig. 5. Intra-tile leakage (a) and its reduction via poly-biasing (b)

This over-consumption effect can be mitigated increasing the p-MOS threshold voltage (V_{th}) of the interface-cells; the latter is typically done either through gate length modulation [22] or using high-V_{th} transistors [23]. The FDSOI CMOS process, target of this work, is provided with multi-V_{th} libraries obtained through the former technique, i.e., gate length modulation, also called *poly-biasing* (PB) and represented in Fig. 5-b. For each logic gate, four different versions are available: PB0 (the standard V_{th}), PB4, PB10 and PB16 (the highest V_{th}).

Since the Vdd assignment process is done at run-time, foreseeing those cells affected by intra-tile leakage is not feasible. At the same time, a conservative approach where all the interface-cells are swapped to high-V_{th} would imply excessive

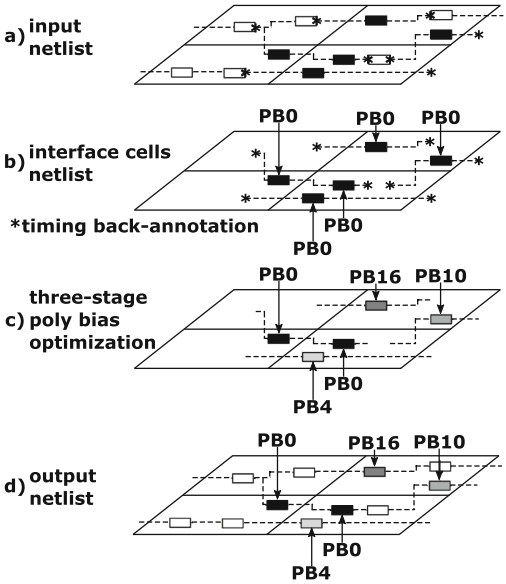


Fig. 6. PB assignment through local re-synthesis

delay overhead. As a compromise we introduce a timing-driven post-placement optimal poly-bias assignment which works as illustrated in Fig. 6. Starting from a placed netlist of standard V_{th} cells, i.e., PB0, the interface-cells are first identified (a) and then virtually isolated in a separated netlist with back-annotated delay information (b). Using the optimization engine embedded into the physical synthesizer, a timing-driven multi-PB assignment is run (c). The netlist returned by the multi-PB assignment step has the minimum leakage configuration, i.e., the largest set of high-V_{th} cells, that satisfies the delay constraints. Finally, the resulting PB assignment is annotated into the main netlist (d).

Step (c) is where the actual leakage optimization takes places. Since the goal is zero-delay penalty, only those interface cells crossed by timing paths with a timing slack greater than a certain threshold are taken into consideration. Moreover, in order to maximize the usage of cells with the highest V_{th} (PB16), we resorted to splitting the procedure into three incremental PB-assignment substages; at each substage, only a given class of PB cells is considered: at the first stage PB16, at the second stage PB10, at the third stage PB4. This procedure allows to eat the available slack first by PB16 cells (which guarantee the highest protection from intra-tile leakage), then by the PB10 cells, and finally by the PB4 cells (for which intra-tile leakage compensation is minimal).

The slack threshold used for the selection of the interface cells (i.e., those considered during PB assignment) may change depending on the actual substage, namely, depending on the class of PB cells. For instance, at the first substage (PB16 assignment), a larger slack threshold is needed as a replacement to PB16

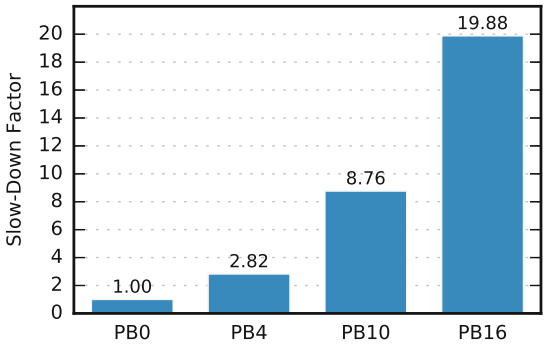


Fig. 7. Normalized average delay overhead for the different poly-bias options

would introduce a large delay penalty; at the third substage (PB4 assignment), the slack threshold can be relaxed. To this aim, a simple rule is followed. We first define a *slow-down factor* α_{sd} for each class of PB cells; this parameter is empirically extracted from the plot of Fig. 7 which reports the delay overhead introduced by poly-biasing: 2.82x for PB4, 8.72x for PB10, 19.88x for PB16. Then, depending on the current substage, the slack-threshold S_{th} is defined by (2):

$$S_{\text{th}}(PB) = d_{\text{avg}} \cdot \alpha_{\text{sd}}, \quad PB = [4, 10, 16]. \tag{2}$$

where d_{avg} represents the average delay of a cell belonging to the critical path, and α_{sd} the slow-down factor reported in Fig. 7. This enables a slack threshold that is proportional to the type of PB used at each substage.

As final remark, it is important to note that the implemented multi-stage re-synthesis is suited not only for the 28 nm FDSOI technology as it can be easily extended to all those technologies that offer multi-V_{th} libraries.

3.2 Simulation and Emulation

Commercial CAD tools lack static-analysis engines that can process level-shifter free multi-Vdd designs. Moreover, since FINE-VH does apply at run-time, the Vdd-selection policy implemented by the power-management unit needs to be emulated at design-time. For what concerns the first point, we opted for a static approach that uses off-line characterizations, while regarding the second issue, we implemented a simple, yet effective timing-driven Vdd-assignment.

Intra-tile Leakage Power Estimation. The key issue of static-analysis of multi-Vdd designs is to estimate the intra-tile leakage power avoiding heavy SPICE simulations of the whole core. We used a static methodology that adopts off-line characterizations. For each logic gate we compiled a look-up table containing the leakage power derating factors for all possible input patterns and all possible VddL/VddH voltage configurations. As for standard timing libraries,

Algorithm 1: Voltage Assignment Procedure

```
   Input: VddL, VddH, f_clk
   Output: Vdd assignment
 1 set_VddL([All_tiles]);
 2 while Worst_Slack < 0 do
 3 |   zero(Tile_Score[All_tiles]);
 4 |   Cell_List ← Cells ∈ Critical Path
 5 |   foreach Cell ∈ Cell_List do
 6 |   |   Cell_Delay ← propagation delay of Cell;
 7 |   |   Tile_ID ← tile hosting Cell;
 8 |   |   Tile_Score[Tile_ID] += Cell_Delay;
 9 |   end
10 |   Tile_Score ← Tile_Score[Tiles@VddL];
11 |   Tiles_Score ← sort(Tile_Score, decreasing);
12 |   Critical_Tile ← Tile_Score[0];
13 |   set_VddH(Critical_Tile);
14 end
```

the LUTs are obtained under different operating conditions. Having those LUTs, the static power of a single cell is estimated using the same model implemented into commercial tools:

$$P_{\text{leak}} = \sum_{i=1}^{2^n} P_i \cdot L_i \cdot k_i \tag{3}$$

where n is the number of input pins (the number of pins, for sequential cells), P_i is the input pattern probability, L_i is the nominal static power extracted from standard timing libraries, and k_i is a derating factor picked from the LUT. Notice that $k_i = 1$ if the driver cell is placed in a tile having the same Vdd of the logic cell under analysis.

Vdd Assignment. In order to emulate at design-time the Vdd-selection strategy we implemented a timing-driven Vdd-assignment whose pseudo-code is given in Algorithm 1. The algorithm applies a cell-based procedure during which tiles are sorted in terms of their timing criticality, that is, the tile containing the largest number of timing critical cells are assigned to VddH first. The procedure returns when all the cells show a positive slack.

The procedure starts by assigning all the tiles to VddL (line 1). For each tile a *criticality score* is initialized to zero (line 3). Once the most critical path is extracted, all the cells belonging to that path are stored in a dedicated list called *Cell_List* (line 4). For each cell in *Cell_List*, the propagation delay is extracted and added to the criticality score of the tile hosting the current cell (lines 5 to 8). Once all the cells in *Cell_List* have been processed, those tiles that are still at VddL are sorted according to their score (line 10, 11). The tile with the highest score (line 12) is assigned to VddH (line 13). The procedure is run until the core presents a positive path slack, that is, when the required f_{clk} is met (line 2).

4 Simulation Results

4.1 The RI5CY Benchmark

With no lack of generality, we validated the proposed FINE-VH flow on the RI5CY core, an open-source RISC-V instruction set architecture [24] used in the low-power parallel-processing platform PULP [10]. The core consists of the following units: prefetch buffer, instruction decoder, a 31 × 32 bit register file, integer ALU, single-cycle 32 × 32 integer multiplier, a control status register, hardware loop unit, debug unit, load and store unit. Figure 8 shows a layout of the die after tile partitioning.

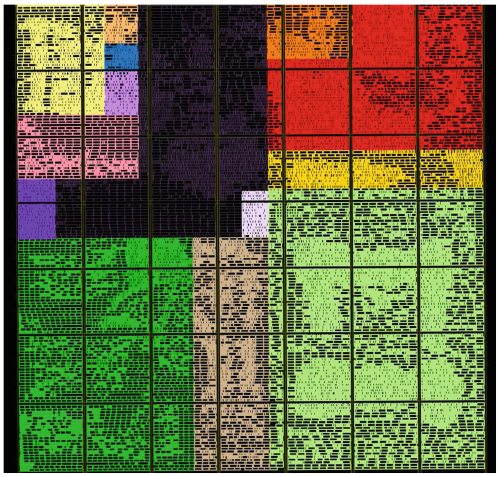


Fig. 8. A 49 tile RI5CY layout after standard-cell placement

4.2 Experimental Set-Up

Static Timing and Power analysis are performed with the STA tool by Synopsys (PrimeTime). We used technology libraries provided by the silicon vendor characterized for Vdd ranging from 0.60 V to 1.00 V (step of 50 mV) and worst-case corner (SS and 125 °C).

The four DVFS schemes used for the comparison have been set as follows.

– **Ideal-DVFS:** for each Vdd, the maximum frequency is extracted and set as the working frequency. The Vdd ranges from 0.60 V up to 1.00 V with a step of 25 mV, resulting in eight (f, Vdd) operating points; for those Vdd not available in the library set, we used a cross-library scaling feature embedded into the STA tool.

- **Vdd-Hopping:** the Vdd range [0.60 V–1.00 V] is split into a finite set of intervals having a fixed width: ΔVdd = 200 mV and ΔVdd = 100 mV; the resulting Vdd values are 0.60 V, 0.80 V, 1.00 V and 0.60 V, 0.70 V, 0.80 V, 0.90 V, 1.00 V respectively. The Vdd is chosen depending on the target frequency (please refer to Fig. 2).
- **Vdd-Dithering:** same Vdd ranges/intervals of the Vdd-Hopping scheme, but power consumption is a liner interpolation of points obtained through ideal-DVFS (please refer to Fig. 2).
- **FINE-VH:** the technique proposed in this work. As for the previous schemes, the Vdd ranges is [0.60 V–1.00 V]; two ΔVdd options are explored, i.e., 200 mV (Vdd range split into two intervals) and 100 mV (Vdd range split into four intervals).

For all the above design strategies the average power is extracted considering realistic switching activities of the primary inputs, i.e., annotating static probabilities and toggle rates extracted from functional simulations.

4.3 Results

Table 1 collects some key figures of the RI5CY core after FINE-VH is applied at different levels of granularity (#Tiles = 1 implies no FINE-VH). Both the core area and the number of layout rows increase with granularity due to wrapper insertion around the tiles. Such a void space is used for de-cap cells insertion and intensive routing. The area overhead ranges from 2.42% for 9 tiles to 5.95% for 49 tiles. The most interesting note is that the active cell area and the nominal delay@1.00V (i.e., delay on the longest path when all the tiles are supplied at 1.00 V) keep almost constant; this proves the convergence of the design flow, even at 49 tiles. As one can observe, the percentage of interface cells increases with the number of tiles, and so do the intra-tile interconnections. However, as will be shown later in the text, the intra-tile leakage overhead is controlled using the proposed poly-biasing optimization.

Table 1. Physical characteristics of the RI5CY with FINE-VH

# Tiles	1	9	25	49
Core area μm^2	36229	37105	37626	38386
# Rows	158	160	161	163
Cell area μm^2	25550	25397	25302	25698
Delay@1.00V ns	3.00	2.95	2.99	3.00
Interface cells	0.0%	38.44%	56.89%	62.65%

Concerning the V_{th} distribution, Fig. 9 shows that the PB assignment strategy makes extensive use of cells at the highest V_{th} (PB16), above 57% in all the

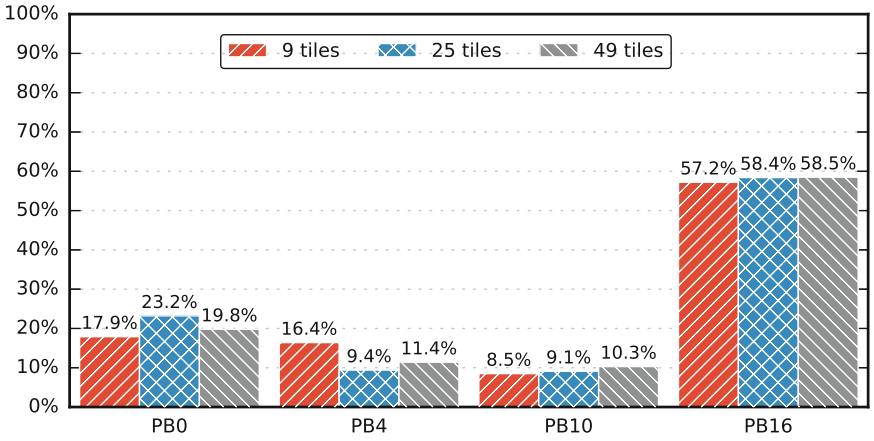


Fig. 9. Poly-bias distribution across the interface-cells

three configurations. This highlights how the leakage optimization engine embedded into commercial tools well fits FINE-VH purposes when properly instructed.

Figure 10 shows the power vs. frequency tradeoff curves for a 49-tile FINE-VH configuration and the three state-of-the-art DVFS schemes. Numbers are normalized w.r.t. an ideal-DVFS implementation (dashed line in the plot) supplied at minimum Vdd. As expected, Vdd-Hopping and Vdd-Dithering do approximate the behavior of ideal-DVFS, even though in a different way. Within each Vdd interval the Vdd-Hopping gets worse at lower frequencies, while Vdd-Dithering always runs close to ideal-DVFS. FINE-VH outperforms the competitors for all the operating points, both for ΔVdd = 200 mV and ΔVdd = 100 mV. When ΔVdd = 200 mV, average power reductions of 43.5% and 27.5% are obtained with respect to Vdd-Hopping and Vdd-Dithering respectively. Moreover, FINE-VH goes quite below ideal-DVFS achieving a 23.4% of power savings. The benefits of the proposed technique are even more empathized at ΔVdd = 100 mV, where the average power consumption is reduced to 35.3% w.r.t Vdd-Dithering and to 34.6% w.r.t. ideal-DVFS.

The power savings for each operating point are detailed through Fig. 11 (the plot does not show savings w.r.t. Vdd-Dithering as they are close to ideal-DVFS). For ΔVdd = 200 mV savings range from 8.5% to 30.6% w.r.t. ideal-DVFS and from 30.0% to 61.0% w.r.t. Vdd-Hopping. For ΔVdd = 100 mV power savings range from 32.0% to 38.2% w.r.t. ideal-DVFS and from 33.6% to 49.7% w.r.t. Vdd-Hopping. When considering a direct comparison to Vdd-Hopping, larger savings are achieved at lower operating frequencies (left side of each Vdd interval), where a finer granularity allows to supply more portions of the layout at the low voltage. Vdd-Hopping, instead, forces the core running at a Vdd that is quite far from the optimal one.

Working with ΔVdd = 100 mV brings larger average savings (w.r.t. ideal-DVFS) for two main reasons: *(i)* a finer voltage granularity allows a better

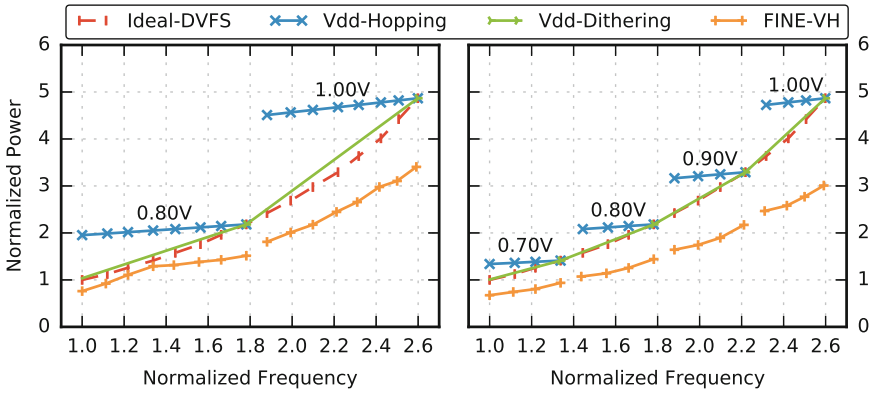


Fig. 10. Comparison of four DVFS techniques: (i) ideal-DVFS, (ii) Vdd-Hopping, (iii) Vdd-Dithering, (iv) FINE-VH (49 tiles); ΔVdd = 200 mV (left), ΔVdd = 100 mV (right)

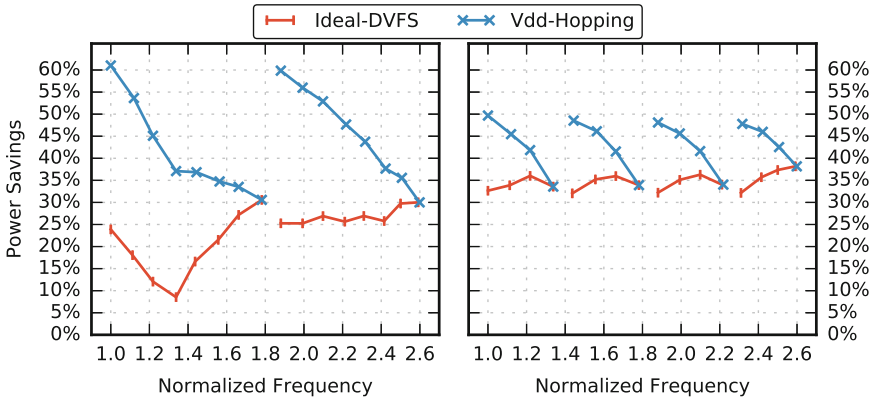


Fig. 11. Power savings of the proposed FINE-VH (49 tiles) w.r.t. ideal-DVFS and Vdd-Hopping; ΔVdd = 200 mV (left), ΔVdd = 100 mV (right)

selection of tiles that can be set at VddL, hence, it helps to get closer the global optimal; *(ii)* a smaller ΔVdd guarantees better noise margins while mitigating the effects of intra-tile leakage. This would suggest that a smaller ΔVdd is a better design option. That's true as long as power/delay overheads of external voltage regulators are neglected. Indeed, when the target frequency imposed at the application level is shifted outside the reference interval, VddL and VddH need to be set to different values (please refer to Fig. 2); this implies some extra power overheads and additional latencies due to off-chip DC/DC converters. Intuitively, a lower number of intervals, i.e., a larger ΔVdd, may allow to cover a larger set of target frequencies with the same pair of values for VddL and VddH. In other words, a larger ΔVdd reduces the probability of an external

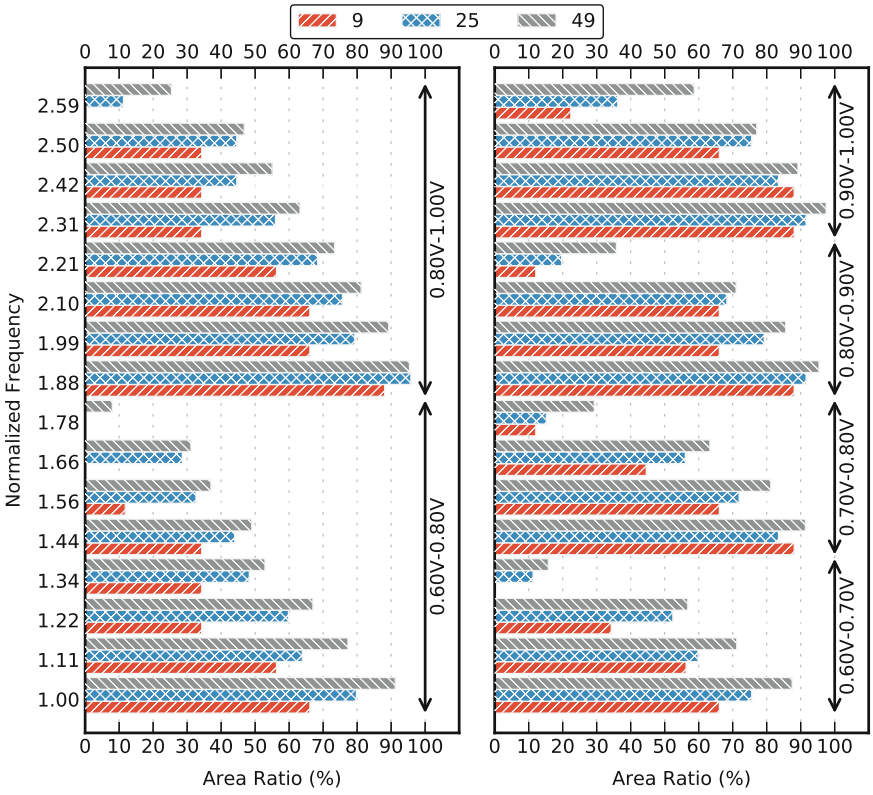


Fig. 12. Percentage of standard cell area @VddL for different number of tiles

voltage shift. The choice of an optimal ΔVdd is a tradeoff imposed by design specifications.

Figure 12 shows the percentage of the cell area supplied at low Vdd when FINE-VH is applied at different granularity, i.e., 9, 25 and 49 tiles. The plot clearly shows that 49 tiles give the best savings. For both the two ΔVdd options, working at higher frequencies decreases the amount of silicon area powered at low Vdd. For instance, at the maximum frequency, $f_{\text{clk}} = 2.59$, with 9 tiles, the percentage of area at low Vdd drastically reduces to zero for $\Delta\text{Vdd} = 200\,\text{mV}$, and to 2.4% for $\Delta\text{Vdd} = 100\,\text{mV}$. A lower granularity implies larger tiles that, most probably, will contain at least one timing critical logic path that forces the selection of a high Vdd. This issue is progressively mitigated as the granularity gets finer. As one can observe, at the maximum frequency $f_{\text{clk}} = 2.59$, with $\Delta\text{Vdd} = 200\,\text{mV}$, the percentage of area powered at low Vdd increases to 11.3% at 25 tiles and 25.4% at 49 tiles. Savings are even larger when $\Delta\text{Vdd} = 100\,\text{mV}$, 36.2% at 25 tiles and 58.7% for 49 tiles. The advantage of a finer granularity can be better appreciated considering the average over the whole frequency spectrum: for $\Delta\text{Vdd} = 200\,\text{mV}$, the average percentage increases to 38.5% (9 tiles), 52.0%

(25 tiles), 58.9% (49 tiles); for ΔVdd = 100 mV, the average percentage increases to 54.0% (9 tile), 60.7% (25 tiles), 69.2% (49 tiles). This confirms once again the rule of thumb: "the finer, the better".

Figure 13 shows the power savings (w.r.t. ideal-DVFS) achieved with the proposed PB optimization. For ΔVdd = 200 mV, do not using poly-biasing nullifies all the savings brought by FINE-VH, i.e., negative savings. On the contrary, the PB assignment helps recovering the overheads due to a level-shifter free strategy as multi-V_{th} cells substantially reduce the intra-tile leakage and are intrinsically less leaky. Our PB strategy achieves average savings of 23.4%. For ΔVdd = 100 mV, though the overhead imposed by intra-tile leakage currents is smaller, our PB optimization allows larger savings, 34.6% (PB) against 10.8% (no PB), without any performance penalty.

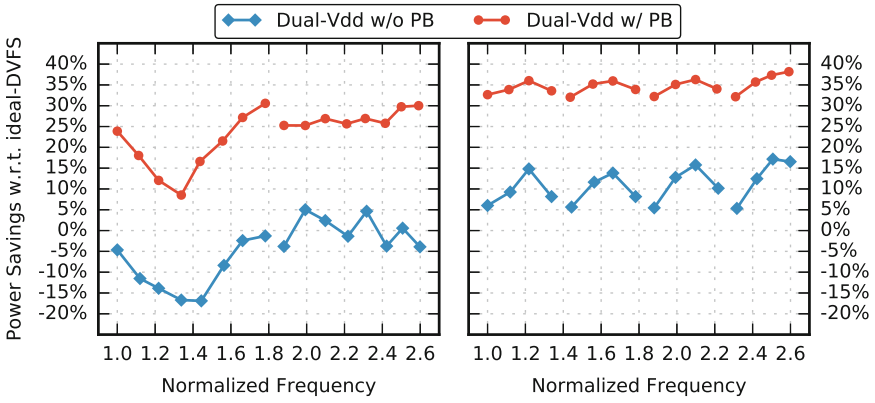


Fig. 13. Power savings with respect to ideal-DVFS before and after PB optimization for ΔVdd = 100 mV and ΔVdd = 200 mV (49 tiles)

As additional piece of information, Fig. 14 reports the Vdd configuration obtained by running Algorithm 1 for the case ΔVdd = 100 mV. The plot shows the Vdd assignment for each tile and each operating frequency; for the sake of space we only show the 25 tiles configuration. When the frequency increases, the number of tiles supplied at low Vdd decreases. However, some tiles, like the #4 and the #5, are more critical than others as they are constantly fed by VddH; other tiles are less critical, like tile #3, #6, #16, and, even at higher frequencies, they still keep running at low Vdd. The most critical functional blocks are the arithmetic units, for which at least one tile is always supplied at high Vdd. The proof that the Vdd-assignment algorithm detects the most timing-critical tiles can be inferred by observing the regularity of the Vdd distribution within each individual voltage interval: once a critical tile is assigned to VddH it never swaps to VddL. This holds for tile #7, #8, #9, #10, #12.

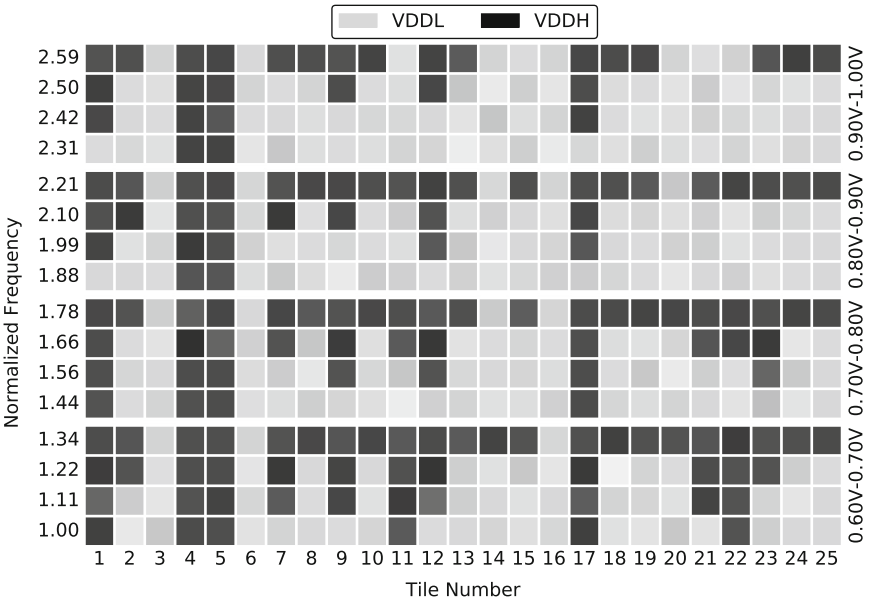


Fig. 14. Voltage assignment (25 tiles)

5 Conclusions and Final Remarks

Ultra-Fine Grain Vdd-Hopping (FINE-VH) improves the efficiency of DVFS schemes in MP-SoCs. We implemented a fully automated layout-assisted, level-shifter free flow which enables tile-based Vdd-Hopping at ultra-fine granularity with minimum design overheads. The proposed technique includes a timing-driven incremental re-synthesis stage for optimal poly-biasing assignment addressing intra-tile leakage waste. The FINE-VH strategy was experimented on a RISC-V core for MP-SoC applications mapped onto a commercial 28 nm FDSOI technology. In order to measure the benefits of the proposed technique, we devised an experimental framework capable of emulating the Vdd assignment at target clock frequency and estimating the intra-tile leakage power.

Experimental results shows that FINE-VH outperforms state-of-the-art dual-Vdd schemes and, most importantly, it goes beyond the theoretical limit imposed by ideal-DVFS. Indeed, when FINE-VH is compared with ideal-DVFS, average power savings range from 23.4% for ΔVdd = 200 mV up to 34.6% for ΔVdd = 100 mV.

An accurate parametric analysis clearly stated finer layout granularity enhances FINE-VH power efficiency; average layout area at low Vdd increases from 38.5% for 9 tiles, up to 58.9% for 49 tiles (when ΔVdd = 200 mV). Once again the rule of thumb "the finer, the better" is confirmed. Our quantitative analysis suggests FINE-VH can work for a wide range of ΔVdd values, from 100 mV (10%Vdd) up to 200 mV (20%Vdd). This represents an important degree

of freedom as a proper selection of ΔVdd may depend on the characteristics of the off-chip voltage regulators and the actual design specifications.

References

1. Venkatachalam, V., Franz, M.: Power reduction techniques for microprocessor systems. ACM Comput. Surv. **37**(3), 195–237 (2005)
2. Nowka, K.J., Carpenter, G.D., MacDonald, E.W., Ngo, H.C., Brock, B.C., Ishii, K.I., Nguyen, T.Y., Burns, J.L.: A 32-bit powerpc system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling. IEEE J. Solid State Circuits **37**(11), 1441–1447 (2002)
3. Kolpe, T., Zhai, A., Sapatnekar, S.S.: Enabling improved power management in multicore processors through clustered DVFS. In: Design, Automation & Test in Europe Conference & Exhibition (DATE 2011), pp. 1–6. IEEE, March 2011
4. Truong, D.N., Cheng, W.H., Mohsenin, T., Yu, Z., Jacobson, A.T., Landge, G., Meeuwsen, M.J., Watnik, C., Tran, A.T., Xiao, Z., Work, E.W., Webb, J.W., Mejia, P.V., Baas, B.M.: A 167-processor computational platform in 65 nm CMOS. IEEE J. Solid State Circuits **44**(4), 1130–1144 (2009)
5. Dighe, S., Vangal, S.R., Aseron, P.A., Kumar, S., Jacob, T., Bowman, K.A., Tschanz, J., Borkar, N., De, V., Howard, J., Erraguntla, V., Borkar, S.: Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. IEEE J. Solid State Circuits **46**(1), 184–193 (2010)
6. Park, J., Shin, D., Chang, N., Pedram, M.: Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors. In: International Symposium on Low-Power Electronics and Design (ISLPED 2010), pp. 419–424, August 2010
7. Miermont, S., Vivet, P., Renaudin, M.: A power supply selector for energy- and area-efficient local dynamic voltage scaling. In: Azémard, N., Svensson, L. (eds.) PATMOS 2007. LNCS, vol. 4644, pp. 556–565. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74442-9_54
8. Beigné, E., Clermidy, F., Lhermet, H., Miermont, S., Thonnart, Y., Tran, X.-T., Valentian, A., Varreau, D., Vivet, P., Popon, X., Lebreton, H.: An asynchronous power aware and adaptive noc based circuit. IEEE J. Solid State Circuits **44**(4), 1167–1177 (2009)
9. Peluso, V., Calimera, A., Macii, E., Alioto, M.: Ultra-fine grain vdd-hopping for energy-efficient multi-processor SoCs. In: International Conference on Very Large Scale Integration (VLSI-SoC 2016), pp. 1–6. IEEE, September 2016
10. Rossi, D., Pullini, A., Loi, I., Gautschi, M., Gürkaynak, F.K., Bartolini, A., Flatresse, P., Benini, L.: A 60 GOPS/W, 1.8 v to 0.9 v body bias ULP cluster in 28 nm UTBB FD-SOI technology. Solid State Electron. **117**, 170–184 (2016)
11. Bolzani, L., Calimera, A., Macii, A., Macii, E., Poncino, M.: Enabling concurrent clock and power gating in an industrial design flow. In: Design, Automation & Test in Europe Conference Exhibition (DATE 2009), pp. 334–339, April 2009
12. Calimera, A., Macii, A., Macii, E., Poncino, M.: Power-gating for leakage control and beyond. In: Reis, R., Cao, Y., Wirth, G. (eds.) Circuit Design for Reliability, pp. 175–205. Springer, New York (2015). doi:10.1007/978-1-4614-4078-9_9
13. Tenace, V., Miryala, S., Calimera, A., Macii, A., Macii, E., Poncino, M.: Row-based body-bias assignment for dynamic thermal clock-skew compensation. Microelectron. J. **45**(5), 530–538 (2014)

14. Liang, X., Wei, G.Y., Brooks, D.: Revival: a variation-tolerant architecture using voltage interpolation and variable latency. IEEE Micro **29**(1), 127–138 (2009)
15. Gupta, M.S., Rivers, J.A., Bose, P., Wei, G.Y., Brooks, D.: Tribeca: design for PVT variations with local recovery and fine-grained adaptation. In: International Symposium on Microarchitecture (MICRO 2009), pp. 435–446, December 2009
16. Kakoee, M.R., Benini, L.: Fine-grained power and body-bias control for near-threshold deep sub-micron CMOS circuits. IEEE J. Emerg. Sel. Topics Circuits Syst. **1**(2), 131–140 (2011)
17. Nakamura, Y., Levacq, D., Xiao, L., Minakawa, T., Niiyama, T., Takamiya, M., Sakurai, T.: 1/5 power reduction by global optimization based on fine-grained body biasing. In: Custom Integrated Circuits Conference (CICC 2008), pp. 547–550. IEEE, September 2008
18. Muramatsu, A., Yasufuku, T., Nomura, M., Takamiya, M., Shinohara, H., Sakurai, T.: 12% power reduction by within-functional-block fine-grained adaptive dual supply voltage control in logic circuits with 42 voltage domains. In: European Solid-State Circuits Conference (ESSCIRC 2011), pp. 191–194. IEEE, September 2011
19. Yasufuku, T., Hirairi, K., Pu, Y., Zheng, Y.F., Takahashi, R., Sasaki, M., Muramatsu, A., Nomura, M., Shinohara, H., Takamiya, M., Sakurai, T., Fuketa, H.: 24% power reduction by post-fabrication dual supply voltage control of 64 voltage domains in VDDmin limited ultra low voltage logic circuits. In: International Symposium on Quality Electronic Design (ISQED 2012), pp. 586–591. IEEE, March 2012
20. Babighian, P., Benini, L., Macii, A., Macii, E.: Post-layout leakage power minimization based on distributed sleep transistor insertion. In: International Symposium on Low Power Electronics and Design (ISLPED 2004), pp. 138–143. IEEE, August 2004
21. Calimera, A., Pullini, A., Sathanur, A.V., Benini, L., Macii, A., Macii, E., Poncino, M.: Design of a family of sleep transistor cells for a clustered power-gating flow in 65nm technology. In: Great Lakes symposium on VLSI (GLSVLSI 2007), pp. 501–504. ACM (2007)
22. Saha, D., Chatterjee, A., Chatterjee, S., Sarkar, C.K.: Row-based dual Vdd assignment, for a level converter free CSA design and its near-threshold operation. In: Advances in Electrical Engineering, pp. 1–6 (2014)
23. Diril, A.U., Dhillon, Y.S., Chatterjee, A., Singh, A.D.: Level-shifter free design of low power dual supply voltage CMOS circuits using dual threshold voltages. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **13**(9), 1103–1107 (2005)
24. Pulp: An open parallel ultra-low-power processing-platform. http://www.pulp-platform.org/

Earth Mover's Distance as a Comparison Metric for Analog Behavior

Alexander W. Rath(✉), Sebastian Simon, Volkan Esen, and Wolfgang Ecker

Infineon Technologies AG, Neubiberg, Germany
{alexander.w.rath,sebastian.simon,
volkan.esen,wolfgang.ecker}@infineon.com

Abstract. Evaluating the outcome of analog simulations is a common, mostly manually carried out task in the pre-silicon verification process of mixed-signal ICs. Its non-automated nature makes it an error-prone and time-consuming procedure. For this very reason, we introduce a novel approach for performing this evaluation automatically resulting in significantly reduced turnaround times as well as a considerably increased reliability of verification results. The presented concept is motivated by an algorithm that is used in optical pattern recognition and is called Earth Mover's Distance. Furthermore, we compare our approach with already existing algorithms, namely Fréchet Distance and Pearson Coefficient, in order to analyze its capability. Finally, we present a case study in which we prove the algorithm by applying it to the results of a mixed-signal simulation at chip-level demonstrating the efficiency of our approach.

Keywords: Mixed-signal verification · Analog behavior · Earth mover's distance · Pearson correlation coefficient · Fréchet distance

1 Introduction

Due to better technology scaling of digital blocks compared to analog blocks more and more parts of the analog implementation of modern IC designs are shifted to the digital domain. However, certain analog components are still indispensable, leading to mixed-signal designs. Hence, the demand for comprehensively verifying such designs is continuously growing and compared to the highly automated verification methodologies in the digital domain, pre-silicon verification in the analog domain usually implies a substantial amount of manual work and computational effort.

On analog module level, pre-silicon verification primarily targets the analysis of electrical characteristics as for instance gain, distortion or input resistance. Although determining electrical characteristics is also targeted on chip top and system level, confirming the functional correctness and the application fitness of the design by means of simulation is the main goal in this regard. The authors of [1], e.g., point out that it is still a common practice to manually inspect

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 173–191, 2017.
DOI: 10.1007/978-3-319-67104-8_9

waveforms when performing the latter. This effectively obstructs the usage of automated checking and regression-based simulations inevitably leading to an increased error-proneness as well as vast expenditures of time. Therefore, they advise to leverage automated solutions to overcome these problems. We were also able to identify these difficulties and we still see room for improvement in this field. For that reason we set ourselves the goal to offer an approach which enables an automated way of comparing analog behavior and allows mixed-signal system level functionality to be verified in a regression-based way.

The contribution is structured as follows. First, we give an overview of related work that covers approaches for comparing analog behavior. Their results form the basis for the motivation of our work. Following this, we explain the theoretical principles of our developed approach. In Sect. 4 we select two competitive algorithms and define a number of comparison criteria in order to show the strengths of our approach. Finally, a case study is outlined in which we point out the achieved benefits.

2 Related Work

If analog behavior of a design shall be checked, one of the most common approaches is to make use of analog assertions. There are different works [2,3] which successfully apply this technique for mixed-signal verification. However, assertions generally lack in abstraction since they have to be defined at signal level. Apart from that, they can usually be applied to check only one particular property of the behavior and are therefore highly application-specific.

Further works, which deal with this topic, are presented in [1,4,5]. They propose approaches which are based on application-specific checkers. These checkers continuously monitor and evaluate circuit characteristics like, for instance, open-loop gain or output noise. However, their solutions require a recurring effort for the development of checkers since they have to be elaborated and implemented for each design characteristic.

Due to the aforementioned deficiencies it is necessary to look for more generic approaches. For this reason we investigated several metrics that are capable of determining the similarity between discrete signals. One frequently used metric is, for instance, the Pearson correlation coefficient (PC). The authors of [6] demonstrate how to employ this measure for the purpose of mixed-signal verification. An overview of other potential metrics like quadratic mean error or Fréchet distance (FD) can be moreover found in [7]. Their results marked the starting point of our work, in which we strove for finding a better suited similarity measure. This newly developed approach is presented and assessed in the subsequent sections.

3 Earth Mover's Distance

The algorithm we propose is based on the so-called earth mover's distance (EMD). This measure is widely used in the field of content-based image retrieval

and is capable to determine the degree of similarity between two distributions. An essential part of this algorithm is the repetitive application of a ground distance. Therefore, we first show how to construct an appropriate ground distance before dwelling on the overall concept of our EMD approach.

3.1 Ground Distance

In order to quantify the similarity of analog signals it is necessary to operate on discrete-time signals. A discrete-time signal $x[i] \in \mathbb{R}$ is a series of data with the discrete time $i \in \mathbb{N}_0$ and $i < n$, where $n \in \mathbb{N}$ is the number of sample points, i. e. the length of the signal.

In this publication, however, we rely on an alternative definition that regards the signal $x[i]$ as a vector $\boldsymbol{x} \in \mathbb{R}^n$:

$$\begin{aligned} \boldsymbol{x} &= (x_0, \ldots, x_i, \ldots, x_{n-1}) \\ &= (x[0], \ldots, x[i], \ldots, x[n-1]) \end{aligned}$$

For our ground distance $\delta_{\mathrm{G}}(x_i, y_j)$, we claim the following fundamental requirement:

> The ground distance should be *bounded* to the interval $[0, 1]$, as it can be shown that a bounded ground distance is a prerequisite for the EMD to be bounded as well.

Several publications [8–10] utilize the Euclidean distance $|x_i - y_j|$ as ground distance. This measure is obviously not compatible with the requirement of boundedness. However, there are methods to subsequently bound such distances, e.g. one could use a division by a parameter that incorporates information about maximum and minimum signal values. Alternatively, one could apply an appropriate function like hyperbolic tangent to it. All these methods have the crucial drawback that the resulting distance functions still treat pairs (x_i, y_j) of big values differently than pairs of small values, although the ratios of the pairs are the same. For example consider

$$\begin{gathered} \frac{x_0 = 1000}{y_0 = 900} = \frac{x_1 = 10}{y_1 = 9}, \text{ but} \\ \tanh |x_0 - y_0| \approx 1 \neq \tanh |x_1 - y_1| \approx 0.1 \end{gathered}$$

This phenomenon effectively cancels out all sections of the involved signals where the signal's values have a relatively small magnitude. Thus, we claim another requirement:

> The magnitude of the ground distance's arguments should not affect the result, i. e. the ground distance has to be *scale-invariant*, such that the following property holds:

$$\delta_{\mathrm{G}}(x_i, y_j) = \delta_{\mathrm{G}}(a x_i, a y_j),\ a \in \mathbb{R}^* \tag{1}$$

In order to fulfill this requirement, we drop the Euclidean distance and use a scaled version of the Tanimoto distance [11,12] instead:

$$\delta_{\mathrm{T}}(x_i, y_j) = \begin{cases} \frac{-0.75 \cdot x_i \cdot y_j}{x_i^2 + y_j^2 - x_i \cdot y_j} + 0.75 & \text{if } (x_i, y_j) \neq (0,0) \\ 0 & \text{else} \end{cases} \tag{2}$$

A 3D-Plot of the function is shown in Fig. 1.

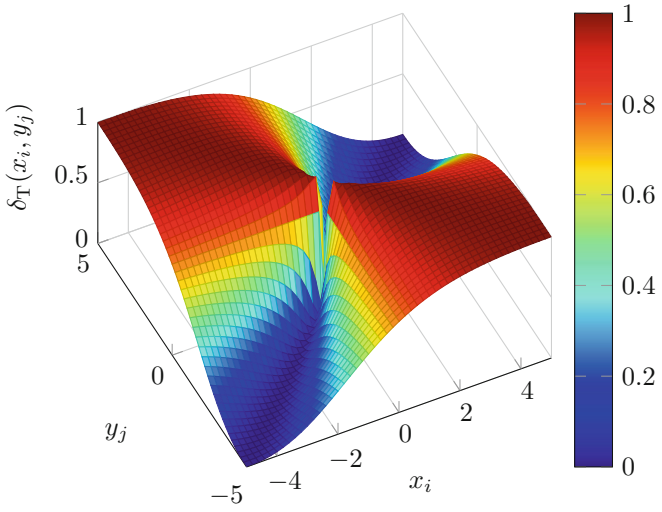


Fig. 1. Tanimoto distance

There are a number of important properties to be mentioned about the scaled Tanimoto distance: It is limited to the interval $[0, 1]$ and becomes 0, whenever $x_i = y_j$. Also, it becomes 1 if the arguments are equivalent in magnitude, but differ in sign (for $x_i \neq 0$): $\delta_{\mathrm{T}}(x_i, -x_i) = 1$.

Furthermore, it is continuous for $(x_i, y_j) \neq (0, 0)$ and it is scale-invariant, i. e. the distance does not depend on the order of magnitude of the arguments. Therefore it satisfies Eq. 1.

Since the scaled Tanimoto distance does not take the position of the elements x_i and y_j into account, we need to extend it. This extension is done by calculating the difference between the positions normalized to the biggest possible difference $n - 1$. The extension and the scaled Tanimoto distance are then combined using the hypot function. Thus, the ground distance is defined as follows:

$$\delta_{\mathrm{G}}(x_i, y_j) = \frac{1}{\sqrt{2}} \cdot \sqrt{\delta_{\mathrm{T}}(x_i, y_j)^2 + \frac{(i-j)^2}{(n-1)^2}} \tag{3}$$

The division by $\sqrt{2}$ is required to retain the limitedness to the interval $[0, 1]$. Note that this expression can only be applied for $n > 1$. For the very unusual case that $n = 1$ we set $\delta_{\mathrm{G}}(x_i, y_j) = \delta_{\mathrm{T}}(x_i, y_j)$.

3.2 Algorithmic Concept

The earth mover's distance is an approach for measuring the distance between two multi-dimensional distributions. Informally, if one must successively transport soil from one pile to another in order to equalize them, the earth mover's distance calculates the minimum cost for the total transport. The distributions are also called *signatures* and represent weighted feature vectors of both signals $\boldsymbol{x}$ and $\boldsymbol{y}$. In the following the concept of this approach, which is based on [9], shall be explained.

Given are two signatures $\mathcal{X}$ and $\mathcal{Y}$ where x_i, y_j denote elements from the respective feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Each signature consists of n *clusters*.

$$\mathcal{X} = \{(x_0, w_{x_0}), \ldots, (x_{n-1}, w_{x_{n-1}})\}$$
$$\mathcal{Y} = \{(y_0, w_{y_0}), \ldots, (y_{n-1}, w_{y_{n-1}})\}$$

For each element x_i, y_j a particular weight w_{x_i}, w_{y_j} can be assigned. In our case a constant weight W is sufficient since no element shall be prioritized:

$$W = w_{x_i} = w_{y_j} = \frac{1}{n} \qquad 0 \leq i \leq n-1, 0 \leq j \leq n-1 \tag{4}$$

Moreover the possibility for transports between both signatures is defined as *flow* $\boldsymbol{F} = [f_{ij}]$, with f_{ij} the flow between x_i and y_j.

Now the idea is to find a flow $\boldsymbol{F}$ that minimizes the costs C for the overall work:

$$C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \delta_{\mathrm{G}}(x_i, y_j) \cdot f_{ij} \tag{5}$$

For our ground distance we chose the combined distance from Eq. 3 since this distance can be used for scalars and is bounded above and below.

The optimization problem in Eq. 5 must furthermore satisfy the following constraints where constraint 7–9 can be simplified according to our assumption in Eq. 4:

$$f_{ij} \geq 0 \qquad 0 \leq i \leq n-1, 0 \leq j \leq n-1 \tag{6}$$

$$\sum_{j=0}^{n-1} f_{ij} \leq w_{x_i} = \frac{1}{n} \qquad 0 \leq i \leq n-1 \tag{7}$$

$$\sum_{i=0}^{n-1} f_{ij} \leq w_{y_j} = \frac{1}{n} \qquad 0 \leq j \leq n-1 \tag{8}$$

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} f_{ij} = \min \left[\sum_{i=0}^{n-1} w_{x_i}, \sum_{j=0}^{n-1} w_{y_j} \right] = 1 \tag{9}$$

Constraint 6 allows only unidirectional flows from $\mathcal{X}$ to $\mathcal{Y}$. Constraint 7 ensures that the weight in $\mathcal{Y}$ matched to x_i does not exceed n^{-1}, while constraint 8 ensures that also the weight in $\mathcal{X}$ matched to y_j does not exceed n^{-1}. Finally, constraint 9 forces the total amount of weight matched to be equal to the overall weight of each signature.

In summary this particular metric consists in solving the following transportation problem for a pair of distributions on condition that the above-stated constraints are satisfied:

$$d_{\mathrm{EM}}(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{F}=[f_{ij}]} C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}) \tag{10}$$

This expression represents a linear optimization problem. Hence, we can solve it by an appropriate algorithm like the simplex method. However, this method shows an exponential worst-case but polynomial average complexity [13], which in turn has a determining influence on the overall performance of this measure. The results for d_{EM} eventually yield values within the interval $[0, 1]$ where 1 symbolizes complete dissimilarity and 0 a full match.

4 Assessment

In order to assess the efficiency of EMD, we compare its behavior with two other algorithms mentioned in Sect. 2; the Pearson correlation coefficient and the Fréchet distance.

In this section we describe how we carried out the assessment and discuss its results. The assessment setup consists of a set of deviation algorithms that are used to generate distorted variants of a given source signal as well as implementations of the EMD, the PC and the FD that compare the source signal with its distorted variants. In this way, we can analyze the behavior of the different algorithms with respect to various types of signal deviations.

4.1 Deviations

A deviation $D(x_i, p)$ is defined as a function that maps the elements of a signal $\boldsymbol{x}$ to the elements of a signal $\boldsymbol{y}$ of the same length:

$$y_i = D(x_i, p), \tag{11}$$

where $p \in \mathbb{R}$ is the severity of the deviation.

In addition, we introduce the following symbols for the sake of readability: The amplitude $\hat{x}$ of a signal $\boldsymbol{x}$

$$\hat{x} = \max_i |x_i|, \tag{12}$$

and the swing Δx of a signal $\boldsymbol{x}$[1]

$$\Delta x = \max_i x_i - \min_i x_i \tag{13}$$

Offset. The first deviation to be defined here is the offset. An offset is a constant, global shift of the source signal along the signal axis. Parameter p_{offset} specifies the severity of the offset:

$$D_{\text{offset}}(x_i, p_{\text{offset}}) = x_i + p_{\text{offset}} \cdot \Delta x \tag{14}$$

In this definition, the actual offset is $p_{\text{offset}} \cdot \Delta x$. This causes p_{offset} to be relative to the swing of the signal. The advantage of this definition is that an offset with, say, $p_{\text{offset}} = 1$ is always significant. This allows for comparing different values of p_{offset} independent of the actual signal.

Figure 2 depicts the signal change in case of an offset deviation.

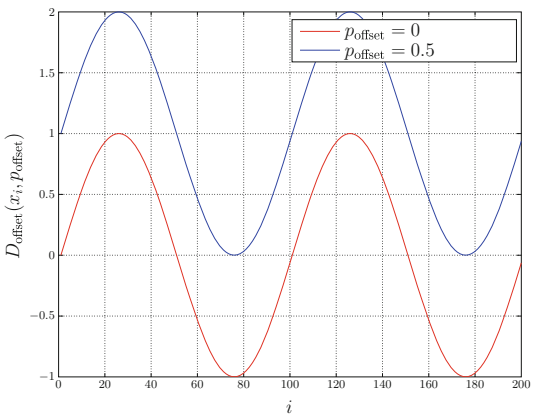


Fig. 2. Sine signal with deviations in offset.

Edge Steepness. This deviation describes a change in the edge steepness of the signal while retaining the signal's original characteristics:

$$D_{\text{edge}}(x_i, p_{\text{edge}}) = \begin{cases} \frac{\hat{x} \cdot \tanh \frac{p_{\text{edge}} \cdot x_i}{\hat{x}}}{\tanh p_{\text{edge}}} & \text{if } p_{\text{edge}} > 0 \\ \frac{\hat{x} \cdot \operatorname{artanh} \frac{p_{\text{edge}} \cdot x_i}{\hat{x}}}{\operatorname{artanh} p_{\text{edge}}} & \text{if } -1 < p_{\text{edge}} < 0 \\ x_i & \text{if } p_{\text{edge}} = 0 \end{cases} \tag{15}$$

[1] Please note that we assume that $\boldsymbol{x}$ is not a constant signal, i. e. $\Delta x \neq 0$ and hence $\hat{x} \neq 0$. We make this limitation as constant signals would unnecessarily complicate some of the deviation definitions that we give in the following subsections without providing any further insight. For example, consider Eq. 14, where $\Delta x = 0$ would cancel out the effect of p_{offset}.

The applied changes to the signal can be regarded as a three-step process: The values of $\boldsymbol{x}$ are scaled to the interval $[-1, 1]$. Then, the signal is compressed using tanh or artanh, respectively, and finally, it is rescaled to the original scale.

The first branch using tanh causes the signal's edges to become more convex, while the second branch using artanh causes the edges to become more concave. Neither of both branches is defined for $p_{\text{edge}} = 0$, but it can be shown that $\lim_{p_{\text{edge}} \to 0} D_{\text{edge}} = x_i$ for both branches. Hence, it makes sense to define a third branch $D_{\text{edge}} = x_i$ for $p_{\text{edge}} = 0$, leaving the signal as it is. Examples are illustrated in Fig. 3.

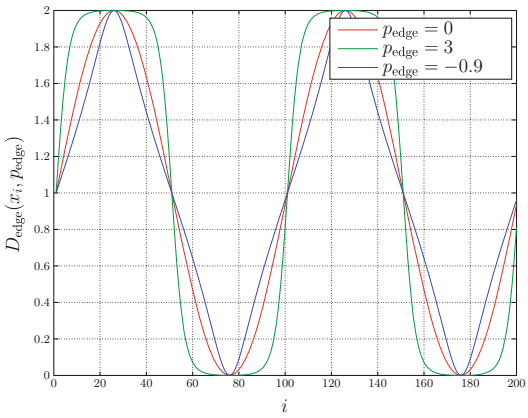


Fig. 3. Sine signal with deviations in edge steepness.

Delay. The delay is a constant, global shift of the signal along the time axis. When shifting a signal along the time axis, it is necessary to fill the emerging gap. Here we have chosen to fill the gap with the value x_0 leading to an acyclic delay. Thus, we define the delay for $0 \leq p_{\text{delay}} \leq 1$ as follows:

$$D_{\text{delay}}(x_i, p_{\text{delay}}) = \begin{cases} x_{i-n \cdot p_{\text{delay}}} & \text{if } i \geq n \cdot p_{\text{delay}} \\ x_0 & \text{else} \end{cases} \tag{16}$$

The severity p_{delay} is relative to the signal's length n.

An example for delay deviation can be found in Fig. 4.

Scaling. Scaling means that the signal is vertically stretched or compressed through multiplying its function by a constant factor p_{scale}:

$$D_{\text{scale}}(x_i, p_{\text{scale}}) = x_i \cdot p_{\text{scale}} \tag{17}$$

Figure 5 shows the effect of adding scale to a signal.

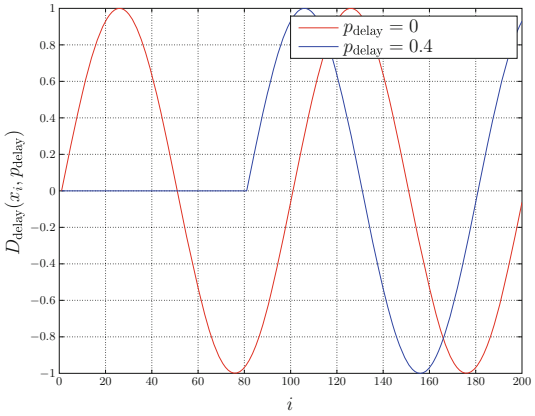


Fig. 4. Sine signal with deviations in delay.

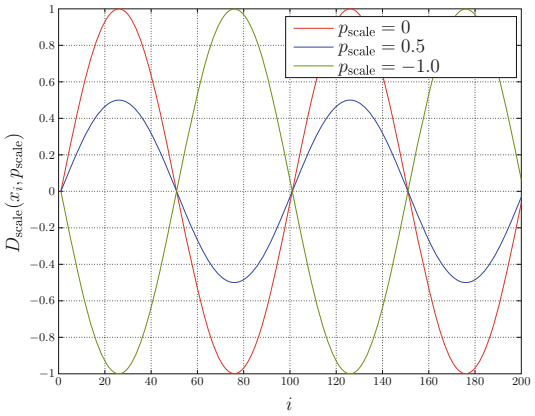


Fig. 5. Sine signal with deviations in scale.

Peak. In contrast to any other deviations presented here, a peak is only a local phenomenon. For n being odd, we define the peak as single spike in the middle of the signal:

$$D_{\text{peak}}(x_i, p_{\text{peak}}) = \begin{cases} x_i + p_{\text{peak}} \cdot \Delta x & \text{if } i = \lfloor \frac{n}{2} \rfloor \\ x_i & \text{else} \end{cases} \tag{18}$$

For n being even, we vary the two adjacent points in the middle of the signal:

$$D_{\text{peak}}(x_i, p_{\text{peak}}) = \begin{cases} x_i + p_{\text{peak}} \cdot \Delta x & \text{if } i \in \left\{ \frac{n}{2} - 1, \frac{n}{2} \right\} \\ x_i & \text{else} \end{cases} \tag{19}$$

Examples for peak deviations are illustrated in Fig. 6

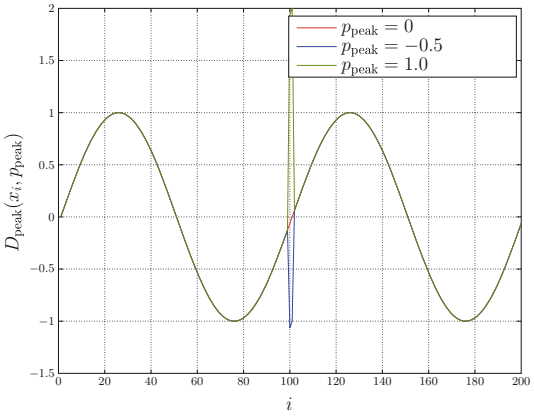


Fig. 6. Sine signal with deviations in peak.

Noise. In principal, adding noise to a signal is like adding offset, except that the offset is now random over time rather than constant:

$$D_{\text{noise}}(x_i, p_{\text{noise}}) = x_i + u_i, \tag{20}$$

where u_i is a uniformly distributed random number with

$$|u_i| \leq \frac{p_{\text{noise}} \cdot \Delta x}{2}. \tag{21}$$

An example for noise deviation is provided in Fig. 7.

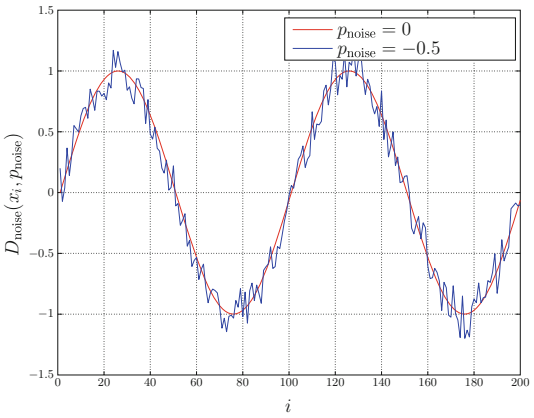


Fig. 7. Sine signal with noise deviation.

4.2 Comparable Algorithms

In this section, we present two additional algorithms that we use to evaluate our EMD approach.

Pearson Correlation Coefficient. The Pearson correlation coefficient is defined as

$$s_{\text{pea}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \bar{x})^2 \cdot \sum_{i=0}^{n-1}(y_i - \bar{y})^2}}, \tag{22}$$

where $\bar{x}$ and $\bar{y}$ represent the arithmetic mean:

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i \quad \text{and} \quad \bar{y} = \frac{1}{n}\sum_{j=0}^{n-1} y_j$$

The result of s_{pea} is a value in the interval $[-1, 1]$, where 1 indicates a full match, 0 a total mismatch and -1 a full match except the sign.

It can be moreover shown that there is a direct relation between Pearson correlation coefficient and normalized cross-correlation. The latter is given in [14] and defined as

$$(\boldsymbol{x} \star \boldsymbol{y})_{\text{norm}}(j) = \frac{\sum_{i=0}^{n-j-1}(x_i - \bar{x}) \cdot (y_{i+j} - \bar{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \bar{x})^2 \cdot \sum_{i=0}^{n-j-1}(y_{i+j} - \bar{y})^2}} \tag{23}$$

With the help of this equation it can be seen that Pearson correlation coefficient is equivalent to normalized cross-correlation at position $j = 0$:

$$s_{\text{pea}}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \star \boldsymbol{y})_{\text{norm}}(0)$$

In other words, an evaluation of the Pearson correlation coefficient additionally embraces an analysis of normalized cross-correlation, which in turn is a widely used measure of similarity in the field of signal processing.

Fréchet Distance. For discrete time signals, the Fréchet distance can be defined as

$$d_{\text{F}}(\boldsymbol{x}, \boldsymbol{y}) = \min_{\alpha,\beta} \max_{k} \delta_{\text{G}}(x_{\alpha(k)}, y_{\beta(k)}), \tag{24}$$

where $\alpha(k)$ and $\beta(k)$ are non-decreasing surjections $k \in \mathbb{N}_0 \mapsto [0, n-1]$. Values for d_{F} are in the same interval as for δ_{G}. Since we use the ground distance δ_{G} as given in Eq. 3, the interval is $[0, 1]$ where 0 indicates a full match and 1 a total mismatch.

The definition in Eq. 24 is not very intuitive, which is why a more intelligible explanation featuring a dog and its owner has become popular [10]: The Fréchet distance is the shortest length of a leash required for a dog and its owner to walk

on two routes—represented by our signals—from one endpoint to the other. Both are allowed to change their speed, but they are not allowed to walk back.

Furthermore, the definition does not indicate how to implement the Fréchet distance. Luckily, [15] provides a pseudo implementation for discrete-time signals.

4.3 Results

The results of the assessment are shown in Fig. 8. In order to make the metrics comparable, we mapped the results of the FD and the EMD to the interval of the PC by subtracting their results from 1. Thus, 1 indicates a full match, 0 a total mismatch and -1 full match except the sign. Please note that the interval $[-1, 0)$ is only applicable to the PC. The source signal $\boldsymbol{x}$ that was used to generate the results is a sine signal with an amplitude of 1 going over two full periods. Furthermore, n is set to 200.

From the results it can be seen that the comparison algorithms judge the delay deviation very differently. FD and PC react very pessimistic and, thus, suggest that source and deviated signal are hardly related to each other. In contrast, EMD detects the deviation but does not drop very low, indicating that there is still a relation between the two signals. The edge deviation is treated similarly by all algorithms, with FD being somewhat more pessimistic than the others. Noise is detected by all algorithms; again, FD being very pessimistic. In addition, FD is not able to differentiate between the noise severities. The offset deviation is detected well by EMD and FD. However, FD shows a peculiar behavior in the sense that it returns bigger values for severity 1 and 10. PC is not sensitive regarding offset. The peak is very well analyzed by PC, while not being detected at all by EMD. The FD detects the peak, however, it can't differentiate between different severities. Scaling is being detected well by EMD and FD, whilst PC is not sensitive, except on the sign.

In order to summarize the assessment, we compressed the results from Fig. 8 into Table 1. That table indicates, whether a metric is able to detect (D)

Table 1. Qualitative analysis of the assessment. The table shows whether an algorithm is able to detect (D) the deviation and whether it still correctly reports a similarity (S) between source and deviated signal.

	FD		PC		EMD	
	D	S	D	S	D	S
Delay	+	−	+	−	+	+
Edge	+	+	+	+	+	+
Noise	+	−	+	+	+	+
Offset	+	−	−	+	+	+
Peak	+	−	+	−	−	+
Scale	+	−	−	+	+	+

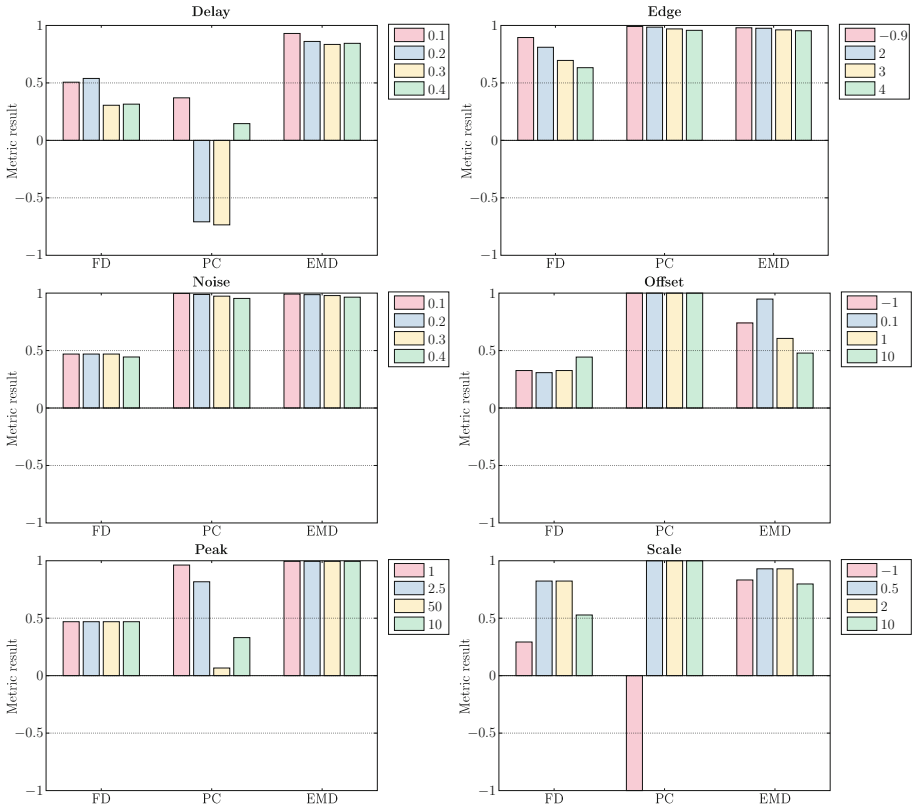


Fig. 8. Results of the assessment. Each diagram stands for a type of deviation. The horizontal axis shows the applied algorithms. The vertical axis shows the result of the comparison between the source signal and the deviated signal with the severity p of the deviation indicated by color. A value of 1 on the vertical axis stands for a full match, i. e. the metric states that the signals being compared are identical; 0 stand for a full mismatch, i. e. the metric states that the signals are completely dissimilar; -1 stands for a full match except the sign (note that only PC can produce such a result). For an interpretation of the results, see Table 1. (Color figure online)

a distortion and whether it can still see a similarity (S) between the original signal and its distorted version. We count a distortion as detected if the average of the absolute value of resulting metric M is significantly smaller than 1:

$$\mathrm{D} = \begin{cases} + & \text{if } |\bar{M}| < 0.99 \\ - & \text{else} \end{cases} \tag{25}$$

For example, PC can't detect a scale distortion, as all four result in Fig. 8 are ± 1 leading to an average of the absolute of $1 \geq 0.99$.

As a criterion for similarity being counted as detected, we define the average of the absolutes to be greater than 0.66:

$$\mathrm{S} = \begin{cases} + & \text{if } |\bar{M}| > 0.66 \\ - & \text{else} \end{cases} \tag{26}$$

It can be seen that the EMD can detect all kind of signal distortions, except peak. Referencing the explanation of EMD, where soil is transported from one pile to another, the peak is not detected because the amount of work that is needed to transport the peak is low. This is due to the peak being used in our assessment is very narrow, i. e. it does not contain much soil. However, the biggest advantage of EMD is that it is always able to correctly detect similarity between source and deviated signal.

5 Case Study

In order to test the reliability of the presented approach, we applied the EMD to a chip level verification of a real-world design. This case study is outlined in the following paragraphs.

5.1 Application

The design we applied the EMD to is micro-controller-based and features a LIN interface and power switches (see Fig. 9). Its application is focused on driving small electric motors—e.g. for window lifts or cooling fans—in cars. Being an

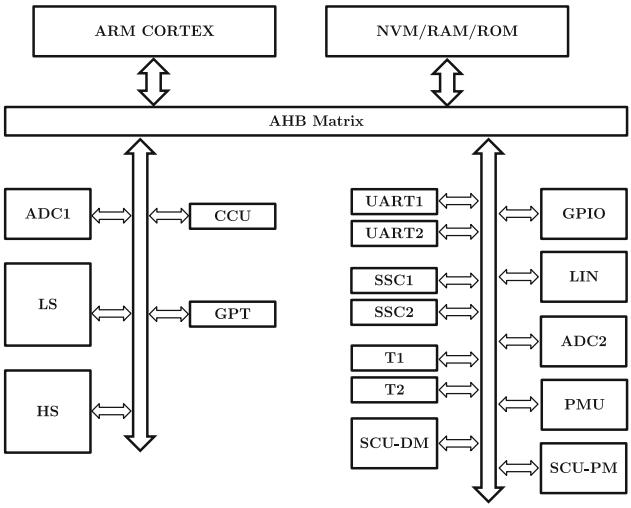


Fig. 9. Block diagram of the SoC used for our analyses.

automotive application, power consumption is of great concern for the design. In order to fulfill the requirements regarding power consumption, the design features a sleep mode as well as wake-up mechanisms. The verification of sleep and wake-up on chip level is crucial, since it affects three design domains—digital and analog hardware as well as software. The chip level verification was performed as a mixed-signal simulation in which the hand-shaking between the domains and the voltage curves of the power rails were checked for correctness. The checking was done in such a way that the results of the initial simulation were inspected manually and stored in the file system. The results of all subsequent simulations were then automatically compared against the results of the initial run using EMD.

In Fig. 10, we show the voltage curve of a rail in the power management unit gained during different simulations. The first curve is the one recorded during the initial simulation and serves as a reference for all subsequent simulations. The colored phases have been defined after the initial simulation, such that the automatic comparison of subsequent results with the ones gained during later simulations could be done phase-wise. Curves A to D represent simulation results after several modifications to the design.

5.2 Results

Tables 2, 3 and 4 show the results of the comparison between signal curves A to D and the reference using EMD, PC and FD, respectively. In order to detect a low similarity, we empirically assume a threshold value of 0.97 for each analysis. "Empirically" means that the threshold value, which serves as a pass/fail criterion, is application dependent and is therefore to be defined by verification engineers on a per-circuit basis. Results which fall below this threshold indicate an insufficient degree of similarity (i. e. fail) and are marked red in the tables.

First, it can be clearly seen that due to a bug the design does not wake up in curve A, resulting in an EMD value of approximately 0.5. This unintended functional behavior is also detected by the other two metrics.

A less obvious signal deviation can be found in curve B. The obtained results reveal a change of certain capacitors in the design leading to a slightly different discharging behavior in phase 3 (EMD $\approx$ 0.96 and FD $\approx$ 0.93). For this case PC already shows the first deficiencies since the metric is not able to detect the deviation (PC $\approx$ 0.99). The aforementioned change of particular external capacitances was actually not a bug, but a specification-compliant adjustment. Therefore, the reference curve had to be replaced by curve B.

In the third case (signal C), wake-up is triggered at a later point in time from the test bench. In this application, this has no impact on the functional behavior of the design. Therefore, the second active phase, which is consequently being altered by a delay, shall not be flagged as too dissimilar to the reference. However, Tables 2, 3 and 4 show that the results for PC and FD fall below the predefined threshold while EMD is able to compensate this deviation. This observation goes in line with the previously stated claim that the threshold has to be adjusted per application, i. e. per circuit.

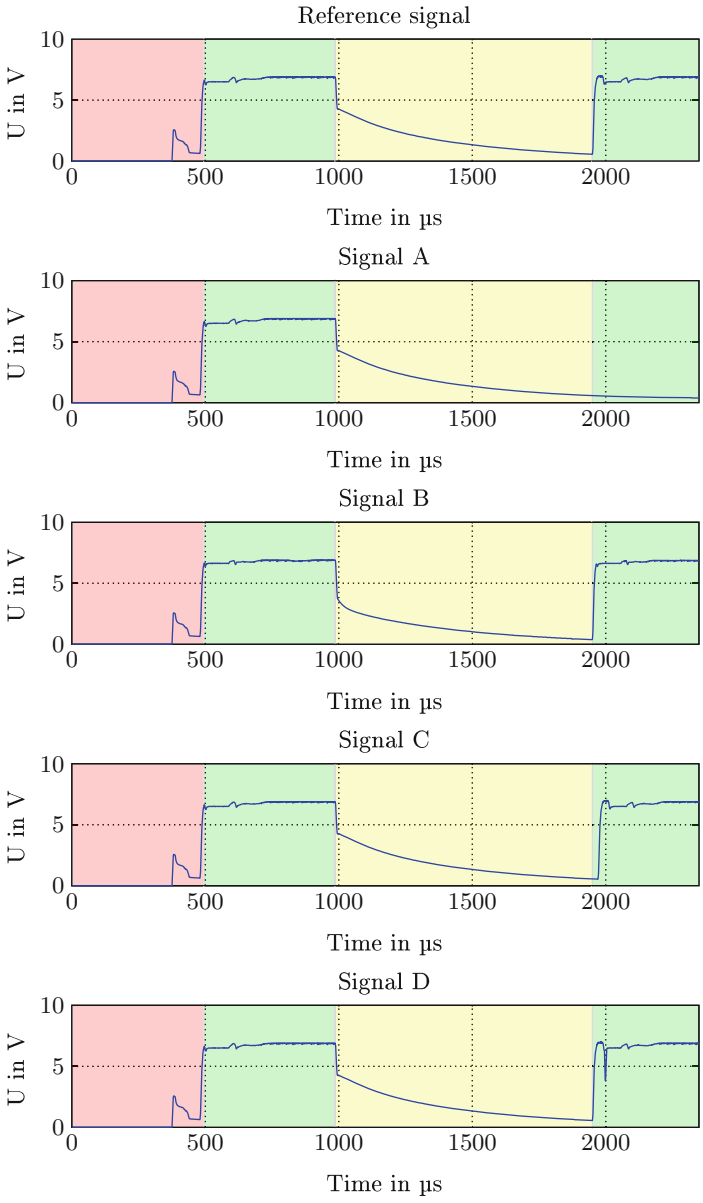


Fig. 10. Voltage curves of a power rail in an automotive design during different design states gained by chip level simulation. The red phase shows the power-up of the design, green stands for the active state and yellow for sleep mode. (Color figure online)

Table 2. Comparison of the curves from Fig. 10 using EMD.

Signals	Phase 1	Phase 2	Phase 3	Phase 4
Ref vs Ref	1.0000	1.0000	1.0000	1.0000
Ref vs A	1.0000	1.0000	0.9999	0.5118
Ref vs B	1.0000	0.9999	0.9577	0.9998
Ref vs C	1.0000	1.0000	1.0000	0.9741
Ref vs D	1.0000	1.0000	1.0000	0.9980

Table 3. Comparison of the curves from Fig. 10 using Pearson Correlation Coefficient.

Signals	Phase 1	Phase 2	Phase 3	Phase 4
Ref vs Ref	1.0000	1.0000	1.0000	1.0000
Ref vs A	1.0000	0.9920	1.0000	−0.4070
Ref vs B	1.0000	0.9369	0.9883	0.9998
Ref vs C	1.0000	1.0000	1.0000	0.5380
Ref vs D	1.0000	1.0000	1.0000	0.8599

Table 4. Comparison of the curves from Fig. 10 using Fréchet Distance.

Signals	Phase 1	Phase 2	Phase 3	Phase 4
Ref vs Ref	1.0000	1.0000	1.0000	1.0000
Ref vs A	0.9998	0.9998	0.9993	0.5009
Ref vs B	0.9998	0.9989	0.9250	0.9965
Ref vs C	1.0000	1.0000	1.0000	0.9413
Ref vs D	1.0000	1.0000	1.0000	0.8718

Switching on an external high load during wake-up leads to the signal shape illustrated by curve D. It is characterized by a peak in the fourth phase at 2000 μs, which can be traced back to a short-time voltage sag. Except for the peak, there is no significant difference between curve D and the reference signal. However, PC and FD are again not able to compensate this deviation and would flag a mismatch (PC ≈ 0.86 and FD ≈ 0.87) whereas EMD correctly detects the functionally unchanged wake-up behavior (EMD ≈ 1.0).

While the critical "sleep mode"-bug in the above-stated mixed-signal design was detected by all three metrics, EMD reveals its strengths in particular when deviations like delay or peak are distorting the signals. In these cases PC and FD are not able to identify correct functional behavior, which in turn confirms our analyses from Sect. 4. A concluding summary of our results can be found in Table 5. We furthermore observed the aforementioned deficiencies when analyzing

Table 5. Overview of results: EMD is the only metric that correctly detects, i. e. compensates all four deviation scenarios.

Signal	Deviation in phase	Reason for deviation	Supposed to be detected?	Detected by
A	4	Bug in the design	yes	EMD, FD, PC
B	3	Modified external capacitance	yes	EMD, FD
C	4	Delayed wake-up	no	FD, PC
D	4	Voltage sag due to temporary high load	no	FD, PC

several other metrics like Euclidean distance, cosine similarity and Hausdorff distance, inspiring us with confidence that our approach represents the most reliable one.

6 Conclusion and Outlook

In this contribution we introduced and described a novel technique for the automated comparison of analog behavior. The technique tackles the necessity of being able to automatically check the functional correctness of analog and mixed-signal ICs during design and verification. It moreover opens the possibility of reducing verification efforts by avoiding manual and error-prone investigations of waveforms since it can be easily leveraged for regression tests. Beyond that, applying the presented concept consequently leads to significantly reduced costs and speeds time to market.

Furthermore, we assessed the approach by comparing it with related techniques and presented a case study, in which we applied the approach to a real-word project confirming its usefulness and high reliability compared to other metrics.

Aside from the outlined benefits in the field of mixed-signal verification, the developed approach has also high impact on several other topics. It can, for instance, be leveraged to optimize objective functions, which in turn is helpful when performing model calibration. In this case an automated regression could be set up to successively fit model parameters of interest. Another field of application is the regression testing of SPICE-style simulators, which can be used to identify diverging simulation behavior caused by differing simulator versions.

Our future work will focus on the extension of the presented methodology regarding usability and flexibility. The long-term goal is to provide a building box that covers the need of verification engineers to simulate and verify mixed-signal designs. This building box shall include methods and techniques for driving, monitoring and checking analog signals, as well as for coverage collection and reference modeling.

References

1. Khan, N., Kashai, Y.: From spec to verification closure: a case study of applying UVM-MS for first pass success to a complex mixed-signal soc design. In: Design and Verification Conference, March 2012
2. Yang, X., Niu, X., Fan, J., Choi, C.: Mixed-signal system-on-a-chip (soc) verification based on systemverilog model. In: 45th Southeastern Symposium on System Theory (SSST), pp. 17–21 (2013)
3. Neumann, F., Sathyamurthy, M., Kotynia, L., Hennig, E., Sommer, R.: UVM-based verification of smart-sensor systems. In: International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pp. 21–24 (2012)
4. Sharma, V., Lakshmanan, G., Tare, S., Dhamankar, S.: Predicting the correlation between analog behavioral models and spice circuits for robust soc verification. In: Proceedings of the 2008 IEEE International Behavioral Modeling and Simulation Workshop, pp. 130–135 (2008)
5. Khan, N., Kashai, Y., Fang, H.: Metric driven verification of mixed-signal designs. In: Design and Verification Conference, March 2011
6. Rath, A.W., Esen, V., Ecker, W.: A transaction-oriented UVM-based library for verification of analog behavior. In: 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 806–811, January 2014
7. Ohlendorf, O., Steinhorst, S., Hartong, W., Hedrich, L.: Comparing two analog waveforms - a trivial task?. In: Zuverlässigkeit und Entwurf (ZuD 2008), Ingolstadt, pp. 153–154, September 2008
8. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**(2), 99–121 (2000)
9. Cohen, S.: Finding Color and Shape Patterns in Images. Ph.D. thesis, Stanford University, May 1999
10. Yoon, S., Yoo, H.M., Yang, S.H., Park, D.S.: Computation of discrete Fréchet distance using CNN. In: 2010 12th International Workshop on Cellular Nanoscale Networks and Their Applications (CNNA), pp. 1–6, February 2010
11. Theodoridis, S., Koutrumbas, K.: Pattern Recognition, 4th edn. Academic Press, Amsterdam (2008)
12. Xiang, H.: Similarity based virtual screening using frequency-based weighting-schemes: effect of the choice of similarity coefficient. In: Poster presentation at the Ninth International Conference on Chemical Structures, June 2011
13. Klee, V., Minty, G.J.: How good is the simplex algorithm?. In: Inequalities-III: Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, September 1–9, 1969, pp. 159–175. Academic Press (1972)
14. Yang, Z.: Fast template matching based on normalized cross correlation with centroid bounding. In: 2010 International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), vol. 2, pp. 224–227, March 2010
15. Eiter, T., Mannila, H.: Computing Discrete Fréchet Distance. Technical report, Technische Universität Wien, April 1994

Approximate Matrix Inversion for Linear Pre-coders in Massive MIMO

Syed Mohsin Abbas(✉) and Chi-Ying Tsui

VLSI Research Laboratory, Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong, China
smabbas@connect.ust.hk

Abstract. This work presents a high-throughput and low-latency matrix inversion design for linear pre-coders for massive multiple-input multiple-output (MIMO) systems. Because of the large number of base station (BS) antennas, as well as the multiple user terminals (UTs), served in a massive MIMO system, the channel matrix dimensions become large. Inversions of such large matrices using direct inversion methods, such as those used in linear pre-coders, would entail prohibitive complexity. To avoid such complexity, Neumann-series-based approximate inversion has been suggested for linear pre-coders in massive MIMO systems. However the performance, complexity and convergence speed of the Neumann series approach highly depends on the initial approximation of the inverse used as a starting point. In this work, we present a novel initial approximation for the Neumann series, which facilitates the parallel computation of the inverse and hence results in lower latency for inversion as well as better accuracy when compared to the previous approaches. A VLSI architecture of the proposed method is implemented for the inversion of a 16×16 matrix in TSMC 65-nm technology. A throughput of 0.54 M to 15 M matrix inversion per second is achieved at a clock frequency of 460 MHz with a 117 K gate count.

Keywords: Zero forcing (ZF) pre-coder · Regularized zero forcing (RZF) pre-coder · Neumann series · Massive MIMO · Matrix inversion

1 Introduction

Multiple-input and multiple-output (MIMO) systems have been adopted in modern wireless communication standards such as IEEE 802.11n, 4G, 3GPP Long Term Evolution and WiMAX. Due to the ever-growing demand for higher data rates without further increasing the communication bandwidth, novel transmission technologies are still an urgent need [1,3]. A potential technology for meeting this demand is large-scale multi-user MIMO, or 'massive MIMO', a form of multi-user multiple-antenna wireless technology which promises substantial improvements in spectral efficiency and energy efficiency [1,3,4,10].

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 192–212, 2017.
DOI: 10.1007/978-3-319-67104-8_10

In massive MIMO systems, a base station (BS) is equipped with a large number of antennas (M), as compared to conventional MIMO systems, while serving a relatively low number (K) of user terminals (UTs). This disparity in number of service antennas to active user terminals helps in focusing energy into ever smaller regions of space to bring huge improvements in throughput and radiated energy efficiency. Other benefits of massive MIMO include extensive use of inexpensive low-power components, reduced latency, simplification of the MAC layer, and robustness against intentional jamming [2].

However, to enjoy the benefits of massive MIMO, an efficient linear pre-coding scheme at the transmitter side is of paramount importance. Pre-coding is a pre-processing technique that exploits channel state information (CSI) at the transmitter to match the transmission to the instantaneous channel conditions [13–17]. In particular, linear pre-coding is a simple and efficient method that can reduce the complexity of a MIMO receiver.

Linear pre-coding includes zero-forcing (ZF), matched filter (MF) and regularized zero forcing (RZF). It has been shown that when $M >> K$, the simplest linear pre-coders are optimal and thermal noise, interference and channel estimation errors vanish [3]. For such cases, simple linear pre-coders like ZF perform exceptionally well, and sum rates of up to 98% of the (optimal) dirty paper coding (DPC) are reported to have been achieved [4].

However, linear pre-coders like ZF and RZF involve channel inversion using the pseudo-inverse of the channel, which involves inverting a $K \times K$ gram matrix Z ($Z = GG^H \; for \; ZF \; and \; Z = (G^H G + \alpha I) \; for \; RZF$), where K is the number of UTs and G is the propagation matrix. Exact matrix inversion methods, such as LU decomposition, Cholesky decomposition and QR decomposition, require cubic order complexity O (K^3) [7]. Hence exact inversion methods cannot be applied due to larger matrix dimensions in massive MIMO systems.

To reduce the matrix inversion complexity, the Neumann-series-based approximate inverse [8,9] has been proposed for large matrices in massive MIMO systems. As the ratio of the number of antennas at the BS to the number of UTs, $\beta = M/K$, increases, Z becomes diagonally dominant [1]. Wu et al. [8] exploited this diagonal dominance property and suggested using the diagonal elements of matrix Z as an initial approximation (seed) for Neumann series expansion. This method is termed the diagonal Neumann series (DNS) in the rest of this chapter for easy reference.

However, for highly correlated channels or low β values, matrix Z may not be very strongly diagonally dominant or perhaps not dominant at all. For such cases, using only the diagonal elements as a seed will result in slow convergence of the Neumann series. Since inversion accuracy is an important parameter, which defines the suppression of the multi-user interference (MUI) in the downlink [9], the DNS will require a greater number of Neumann series terms to achieve a certain accuracy.

For scenarios with low β values or highly correlated channels, Prabhu et al. [9] proposed to include some off-diagonal elements in addition to the diagonal elements to form the initial approximation for the Neumann series. In particular,

they used a tri-diagonal matrix as the seed. We call this method the tri-diagonal Neumann series (TNS) for easy reference. Using the tri-diagonal matrix as a seed in the Neumann series results in better convergence and superior performance as compared to the DNS [8].

For the approximate inverse calculation of either the DNS or TNS, the inverse of the seed matrix has to be calculated, and hence it should be easily invertible. However using the TNS as seed matrix results in an increase in complexity for calculating the inverse for the tri-diagonal matrix as compared to the DNS. Prabhu et al. [9] proposed to use a modified Gauss-elimination-based algorithm to obtain the inverse of the tri-diagonal matrix. However due to the sequential nature of the algorithm, the latency is proportional to K and cannot be reduced with parallel hardware. Thus it may not be desirable for a system that has a very low latency requirement.

In this work, we present a low-latency and high-throughput matrix inversion method, also based on the Neumann series. In particular we propose to use a new seed matrix [18], for which the inverse can be easily calculated in a parallel fashion with lower complexity, hence leading to lower latency and higher throughput.

2 System Model

For this work, we consider single-cell large-scale multi-user MIMO (MU-MIMO) with one BS and K UTs. The BS is equipped with M antennas and the UTs have a single antenna, such that M antennas at the BS communicate with K single-antenna UTs ($M > K > 1$). The system model considered in this work is in line with the corresponding system model described in [1].

The reverse link propagation matrix, G with a dimension of $M \times K$ is the product of the $M \times K$ matrix H, which accounts for the small-scale fading, and a $K \times K$ diagonal matrix $D_\beta^{1/2}$, which accounts for the large-scale fading [1]. Hence $G = HD_\beta^{1/2}$, where the K^{th} column vector of H describes the small-scale fading between the K^{th} UT and the M antennas, while the K^{th} diagonal element of $D_\beta^{1/2}$ is the large-scale fading coefficient. For the forward link, the BS transmits an $M \times 1$ vector, x, through its M antennas and the K UTs collectively receive a $K \times 1$ vector:

$$y = \sqrt{\rho} G^H x + w, \tag{1}$$

where w is the $K \times 1$ vector of the receiver noise whose components are independent and distributed as $CN(0, 1)$. The total transmitted power is normalized to satisfy $E[x^H x] = 1$. Hence, $\rho_T > 0$ denotes the total downlink power.

2.1 Linear Pre-coding at the Transmitter

The $M \times 1$ transmitted signal vector x is given as $x = Ws$, where W is an $M \times K$ pre-coding matrix and s is a $K \times 1$ vector containing data symbols intended for

K users. For MU-MIMO with large arrays of antennas at the BS, the columns of the propagation matrix are asymptotically orthogonal under favorable conditions [1], and

$$(\frac{G^H G}{M})_{M \gg K} \approx D_\beta. \tag{2}$$

Due to this property, simple linear pre-coders, like MF, ZF and RZF, can be deployed on the transmitter side for close to optimal performance. The MF pre-coder, despite being simple, requires more BS antennas to achieve close to optimal performance [1]. Hence we focus on a ZF and RZF pre-coding scheme in our system model.

Zero Forcing (ZF) Pre-coding: The ZF pre-coding schemes [13] have been extensively studied on multiuser systems as ZF decouples the multiuser channel into independent single-user channels and has been shown to achieve a large proportion of dirty paper coding capacity [12].

The ZF [11] expression is given by

$$W_{ZF} = G(G^H G)^{-1}. \tag{3}$$

Regularized Zero Forcing (RZF) Pre-coding: The regularized ZF pre-coder, as the name implies, introduces a regularization parameter in the channel inversion. One way to regularize an inverse is to add a multiple of the identity matrix before inversion, such as

$$W_{RZF} = G(G^H G + \alpha I)^{-1}. \tag{4}$$

The amount of interference depends on $\alpha > 0$. If $\alpha = 0$, then it essentially becomes a ZF pre-coder(Eq. 3). The amount of interference increases with α , and the optimum value of α is given as [11]

$$\alpha = \frac{K}{\rho} \; where \; K \; is \; number \; of \; users \; and \; \rho \; is \; SNR. \tag{5}$$

Both ZF and RZF pre-coding schemes involve inversion of a $K \times K$ gram matrix Z ($Z = GG^H \; for \; ZF \; and \; Z = (G^H G + \alpha I) \; for \; RZF$), which is an expensive and critical operation. In the next section, we will present a low-latency approximate matrix inversion suitable for massive MIMO.

3 Approximate Matrix Inversion

An exact matrix inversion operation for a $K \times K$ gram matrix Z requires cubic order complexity $\mathrm{O}(K^3)$ [7]. Due to the increasing channel matrix size in massive MIMO systems, such direct inversion methods may not be a very efficient solution. To reduce the computational complexity, the Neumann series has been proposed as an alternative to exact inversion methods [8,9].

If Z is very "close to" an invertible matrix X in the sense that

$$lim_{n\to\infty}(I - X^{-1}Z)^n = 0,$$

then Z is nonsingular and its inverse is given by [8]

$$Z^{-1} = \sum_{n=0}^{\infty}(I - X^{-1}Z)^n X^{-1}. \tag{6}$$

And if the $L-1$ terms of the Neumann series are used for approximation, then Eq. 6 can be modified as

$$Z^{-1} = \sum_{n=0}^{L}(I - X^{-1}Z)^n X^{-1}. \tag{7}$$

If the first two terms of the Neumann series are used ($L = 1$), and Eq. 7 becomes

$$Z^{-1} = X^{-1} + (I - X^{-1}Z)X^{-1}. \tag{8}$$

Similarly, if the first four terms of the Neumann series are used ($L = 3$), Eq. 7 becomes

$$Z^{-1} = X^{-1} + (I - X^{-1}Z)X^{-1} + (I - X^{-1}Z)^2X^{-1} + (I - X^{-1}Z)^3X^{-1}. \tag{9}$$

The matrix X, termed as the 'seed', is an initial approximation of Z^{-1}, which should be easily invertible for lower latency and higher throughput.

Diagonal Neumann Series. As mentioned, Wu et al. [8] suggested using the diagonal elements of matrix Z as an initial approximation (seed) for Neumann series expansion, which we refer to as the DNS.

Inversion of a diagonal 'seed' matrix is very simple, and it requires only K inversion operations. Thus for $L = 2$, the complexity of the inversion operation for a $K \times K$ gram matrix Z using Eq. 8 is $\mathcal{O}(K^2)$ [8] as opposed to cubic order complexity $\mathcal{O}(K^3)$ of the exact inversion methods [7].

Tri-Diagonal Neumann Series. For highly correlated channels or low β values, using only the diagonal elements as a seed will result in slow convergence of the Neumann series. Hence more Neumann series terms will be required to achieve a certain accuracy, which will result in higher complexity as each additional term for a Neuman series involves computing power of the matrix $(I - X^{-1}Z)^n$, as shown in Eq. 9.

As mentioned, Prabhu et al. [9] proposed that by including some off-diagonal elements with the diagonal elements as a seed, the convergence of the Neumann series could be improved. Hence they proposed using a tri-diagonal matrix as a seed, which we refer to as the TNS. This method results in better convergence

and superior performance as compared to the DNS [8]. However the complexity of obtaining the inverse of the tri-diagonal seed matrix also increases as compared to the DNS.

4 Proposed Method

Here we propose a new seed matrix, which has similar performance to the TNS, but is much easier to invert, hence resulting in lower overall latency and resulting in higher throughput. In addition to the diagonal elements of Z, we also keep the first column to form the seed matrix for the Neumann series. The following shows an example of the seed matrix (X) of a 4×4 gram matrix Z:

$$X = \begin{bmatrix} Z_{00} & 0 & 0 & 0 \\ Z_{10} & Z_{11} & 0 & 0 \\ Z_{20} & 0 & Z_{22} & 0 \\ Z_{30} & 0 & 0 & Z_{33} \end{bmatrix}. \tag{10}$$

The reason for using X as a seed matrix is that it is much easier to invert when compared to using a tri-diagonal matrix as a seed. The inverse of X can be obtained using the following procedure. X can be decomposed as $X = DA$ as follows:

$$X = \begin{bmatrix} Z_{00} & 0 & 0 & 0 \\ 0 & Z_{11} & 0 & 0 \\ 0 & 0 & Z_{22} & 0 \\ 0 & 0 & 0 & Z_{33} \end{bmatrix} \begin{bmatrix} 1 & 0\,0\,0 \\ Z_{10}Z_{11}^{-1} & 1\,0\,0 \\ Z_{20}Z_{22}^{-1} & 0\,1\,0 \\ Z_{30}Z_{33}^{-1} & 0\,0\,1 \end{bmatrix}. \tag{11}$$

Matrix D is a diagonal matrix of which the inverse is easy to obtain by simply calculating the reciprocal of the diagonal elements. Similarly, matrix A is called an atomic triangular matrix [7], of which the inverse can be obtained easily by the following:

$$\begin{bmatrix} 1\,0\,0\,0 \\ a\,1\,0\,0 \\ b\,0\,1\,0 \\ c\,0\,0\,1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0\,0\,0 \\ -a & 1\,0\,0 \\ -b & 0\,1\,0 \\ -c & 0\,0\,1 \end{bmatrix}. \tag{12}$$

Hence the inverse of the proposed seed matrix is calculated as $X^{-1} = A^{-1}D^{-1}$, and

$$\begin{bmatrix} 1 & 0\,0\,0 \\ -Z_{10}Z_{11}^{-1} & 1\,0\,0 \\ -Z_{20}Z_{22}^{-1} & 0\,1\,0 \\ -Z_{30}Z_{33}^{-1} & 0\,0\,1 \end{bmatrix} \begin{bmatrix} Z_{00}^{-1} & 0 & 0 & 0 \\ 0 & Z_{11}^{-1} & 0 & 0 \\ 0 & 0 & Z_{22}^{-1} & 0 \\ 0 & 0 & 0 & Z_{33}^{-1} \end{bmatrix} = \begin{bmatrix} Z_{00}^{-1} & 0 & 0 & 0 \\ -Z_{10}Z_{00}^{-1}Z_{11}^{-1} & Z_{11}^{-1} & 0 & 0 \\ -Z_{20}Z_{00}^{-1}Z_{22}^{-1} & 0 & Z_{22}^{-1} & 0 \\ -Z_{30}Z_{00}^{-1}Z_{33}^{-1} & 0 & 0 & Z_{33}^{-1} \end{bmatrix}. \tag{13}$$

From the above equation, it can be seen that X^{-1} can be calculated in constant time as long as we have enough parallel hardware to calculate the reciprocal and the multiplication.

For the inversion of the tri-diagonal matrix, a modified Gaussian elimination method was proposed in [9]. The dependency of the Gaussian elimination algorithm makes it difficult to parallelize the computation, and hence the latency for computing the inverse of the tri-diagonal matrix is in the order of $O(K)$ [9], even when a large number of hardware resources are available.

5 Performance Analysis

We simulate an un-coded large-scale MU-MIMO downlink system, employing 4-QAM modulation and linear pre-coding at the transmitter side and MF detection at the receiver. For computing the inverse of the gram matrix Z, the Neumann series is employed with different seed matrices. Figures 1, 2, 3, 4, 5 and 6 compare the bit error rate (BER) performance of the Neumann-series-based inverse, using different seed matrices, with that using exact matrix inversion under the conditions of $\beta = 5$ and with different numbers of UTs ($K = 8$).

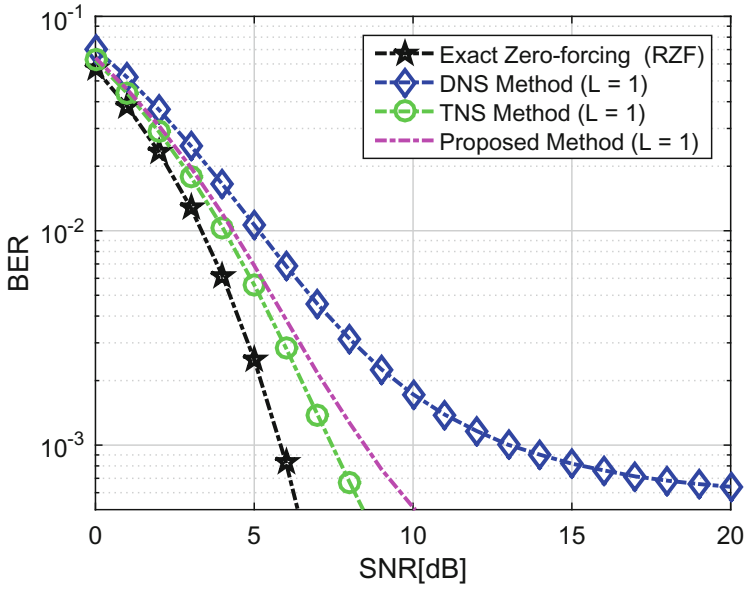
Figure 1 shows that for $K = 8$, when only the first two terms of the Neumann series are included (i.e., $L = 1$), the performance of the proposed seed matrix and TNS is much better than that of the DNS. Figure 1 (a) depicts the experimental results for ZF pre-coding and Fig. 1 (b) shows the result for RZF pre-coding at the transmitter.

Figure 2 shows that when we increase the number of Neumann series terms (i.e., when $L = 3$), the BER performance of the proposed seed matrix and TNS becomes closer to that of the exact inversion, which shows that the proposed seed matrix and TNS require fewer Neumann series terms as compared to the DNS. The same BER performance trend can be observed for different numbers of UTs (K), as shown in Figs. 1, 2, 3, 4, 5 and 6.

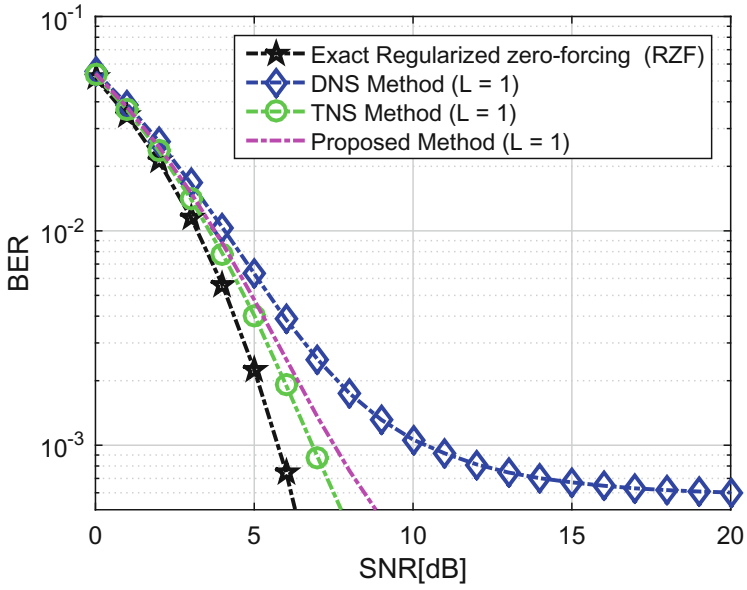
For the experimental results presented in Figs. 1, 2, 3, 4, 5 and 6, a fixed value of $\beta = 5$ is used. For evaluating the performance of the proposed scheme under different values of β, the proposed scheme is compared with the DNS [8] and TNS [9] methods by evaluating the SNR-loss compared to the exact inversion to achieve a BER of 10^{-3}. The results are summarized in Figs. 7, 8 and 9, which show the performance comparisons for different numbers of UTs ($K = 8, 12$ *and* 16), different pre-coding methods (ZF and RZF) and different numbers of Neumann series terms (for $L = 1$ and $L = 3$), respectively.

Experimental results (Figs. 7, 8 and 9) show that for low values of β ($\beta < 10$), the TNS and the proposed method give better performance than the DNS, whereas with higher values of β (>10), the performance loss is almost zero. This means that using only the first two terms is sufficient for inverse calculation ($L = 1$). However for low β values, a larger number of Neumann series terms is required to achieve a certain accuracy ($L = 3$).

In general, it can be deduced from the experimental results that using the proposed seed matrix gives a better performance than using the DNS, and is almost the same as using the TNS. However the complexity and latency of the proposed seed matrix is less than that of the TNS method.

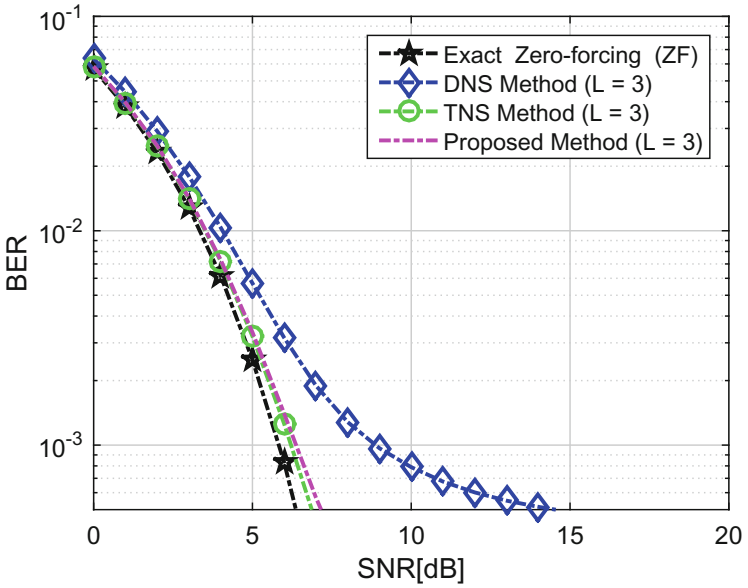


(a) Zero forcing pre-coding

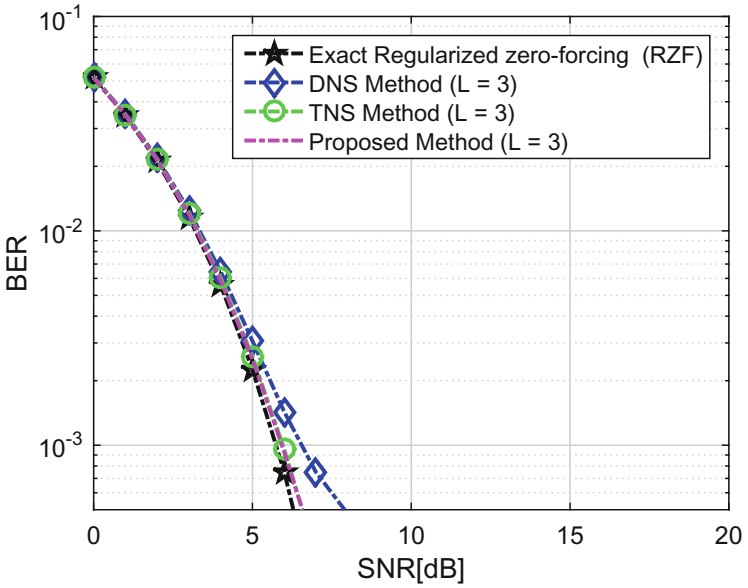


(b) Regularized zero forcing pre-coding

Fig. 1. Performance comparison $\beta = 5$, number of UTs ($K = 8$), $L = 1$

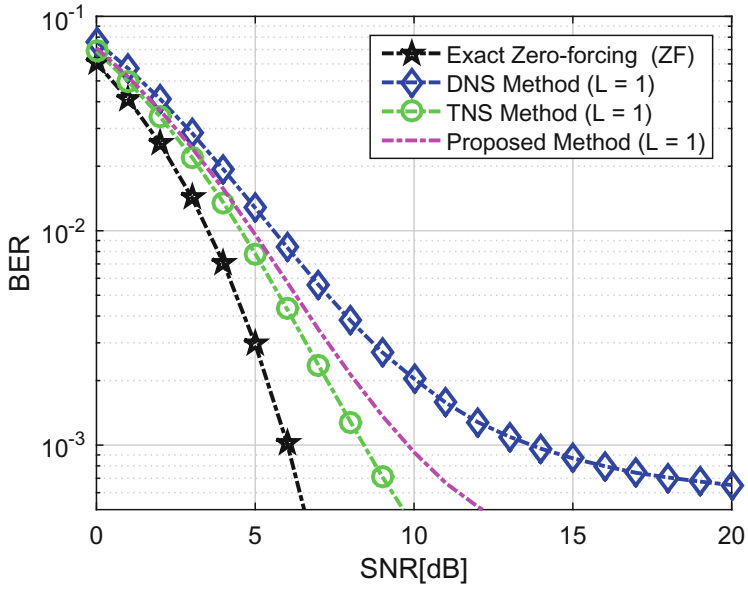


(a) Zero forcing pre-coding

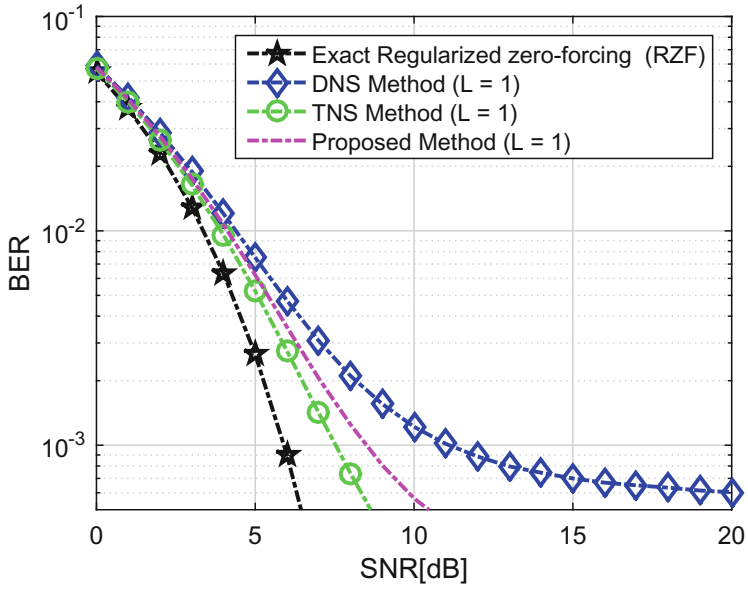


(b) Regularized zero forcing pre-coding

Fig. 2. Performance comparison $\beta = 5$, number of UTs ($K = 8$), $L = 3$

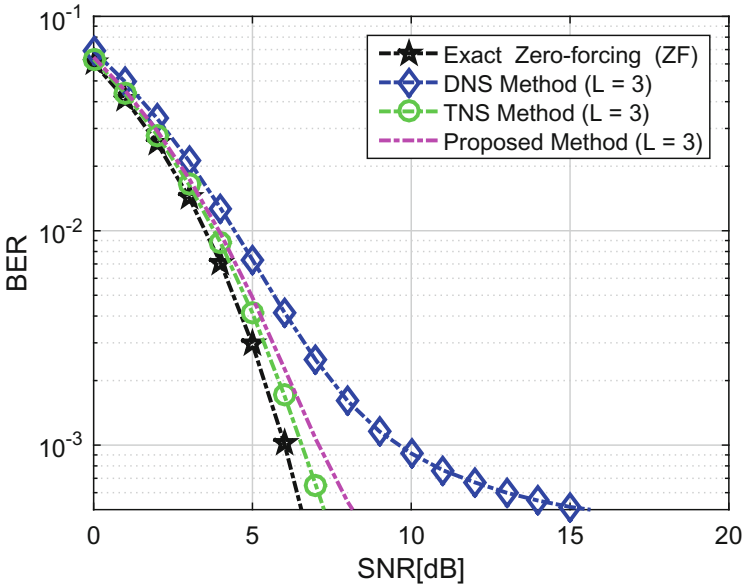


(a) Zero forcing pre-coding

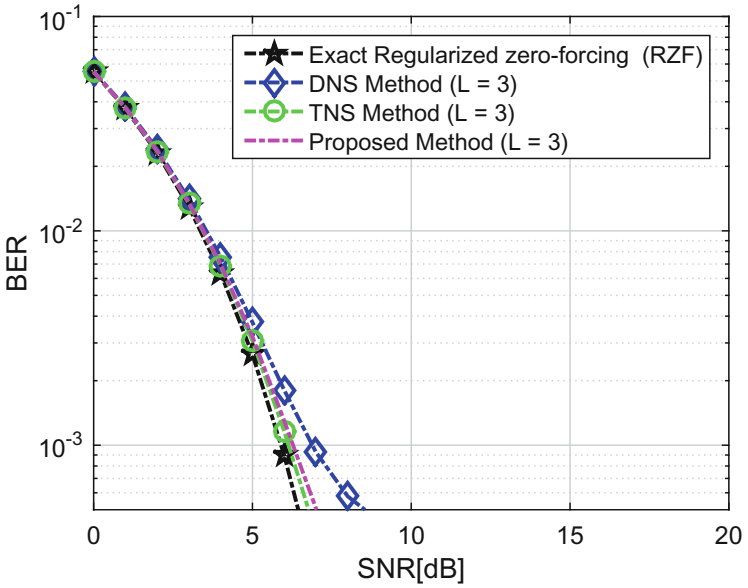


(b) Regularized zero forcing pre-coding

Fig. 3. Performance comparison $\beta = 5$, number of UTs ($K = 12$), $L = 1$

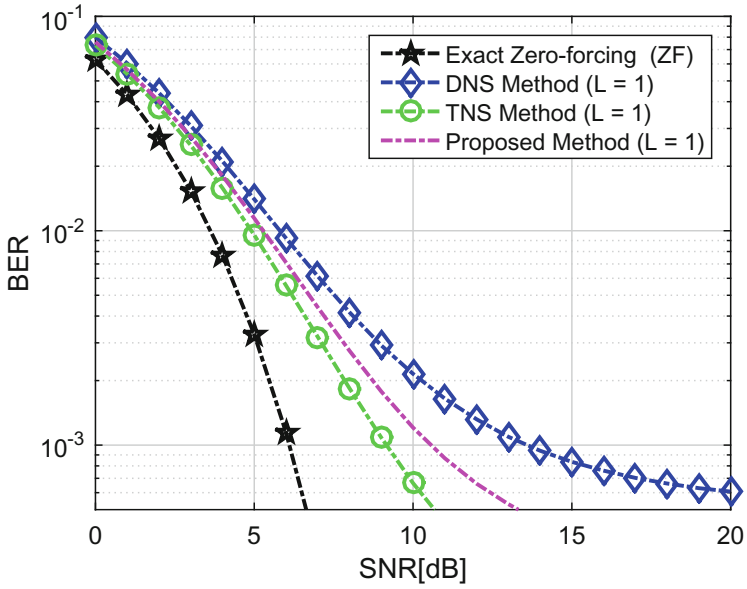


(a) Zero forcing pre-coding

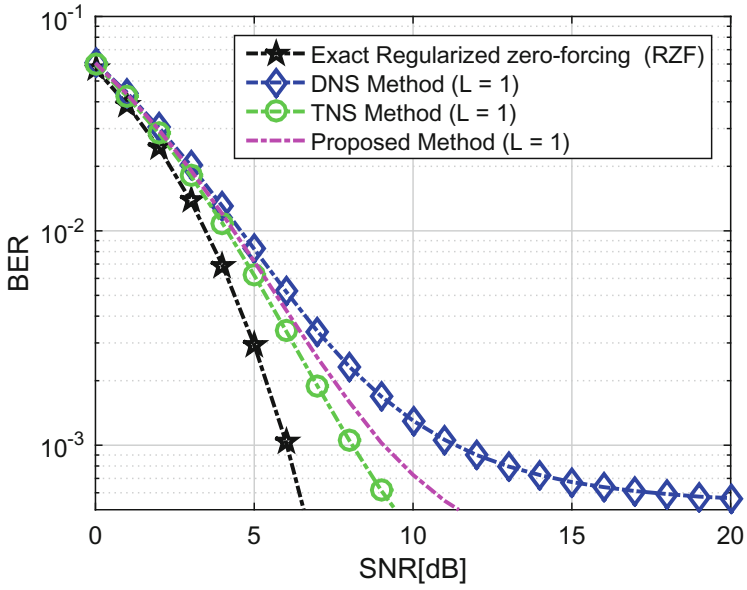


(b) Regularized zero forcing pre-coding

Fig. 4. Performance comparison $\beta = 5$, number of UTs ($K = 12$), $L = 3$

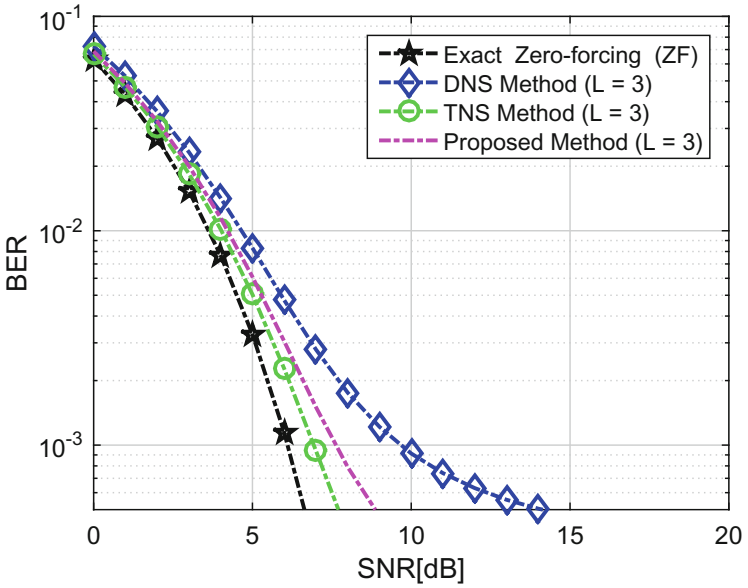


(a) Zero forcing pre-coding

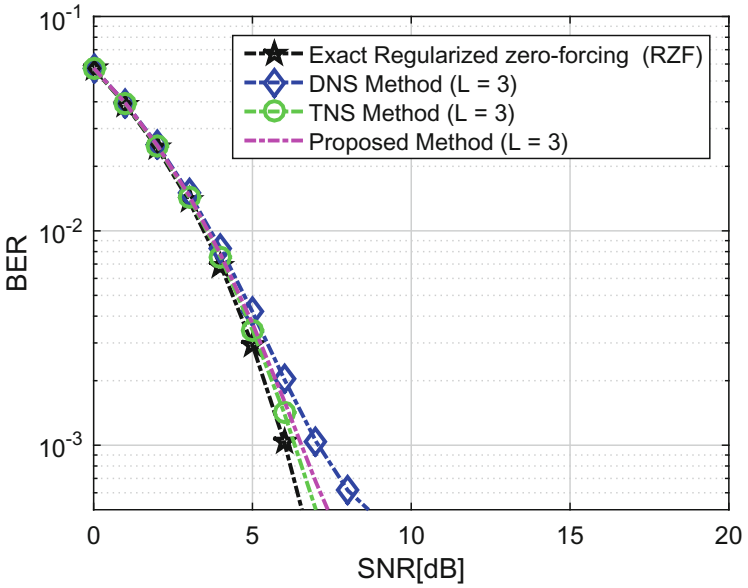


(b) Regularized zero forcing pre-coding

Fig. 5. Performance comparison $\beta = 5$, number of UTs ($K = 16$), $L = 1$

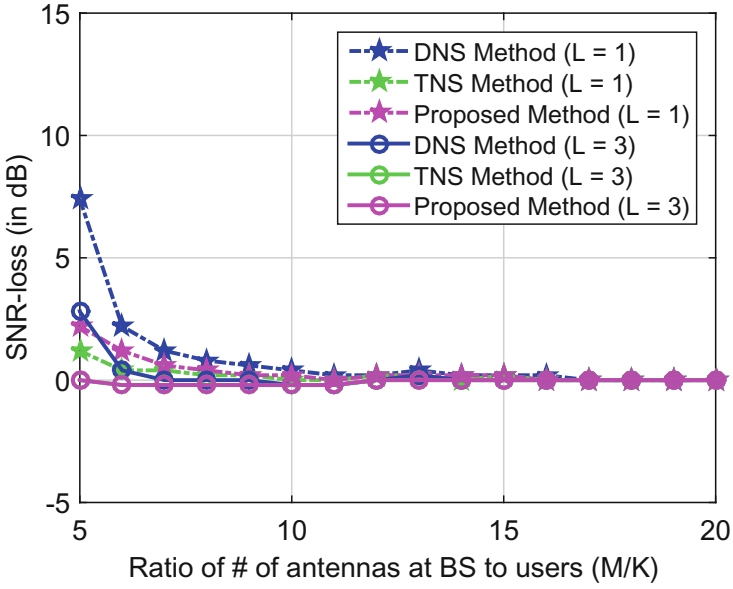


(a) Zero forcing pre-coding

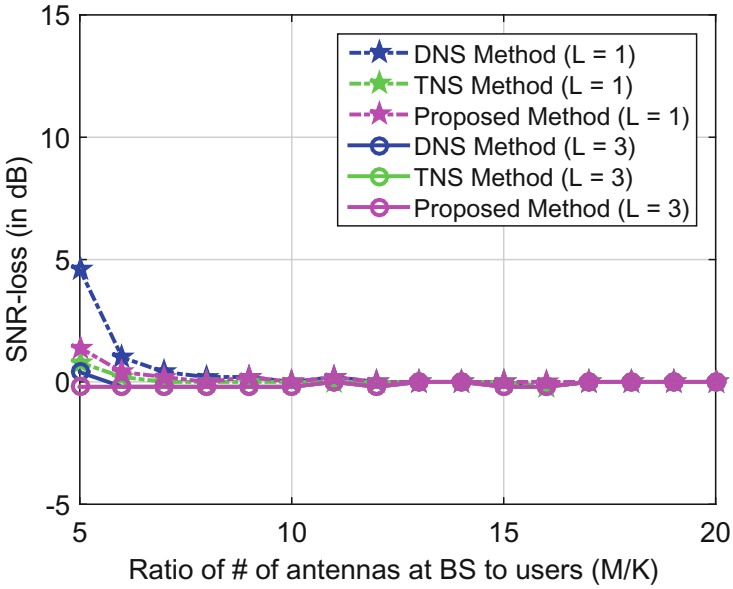


(b) Regularized zero forcing pre-coding

Fig. 6. Performance comparision $\beta = 5$, number of UTs ($K = 16$), $L = 3$

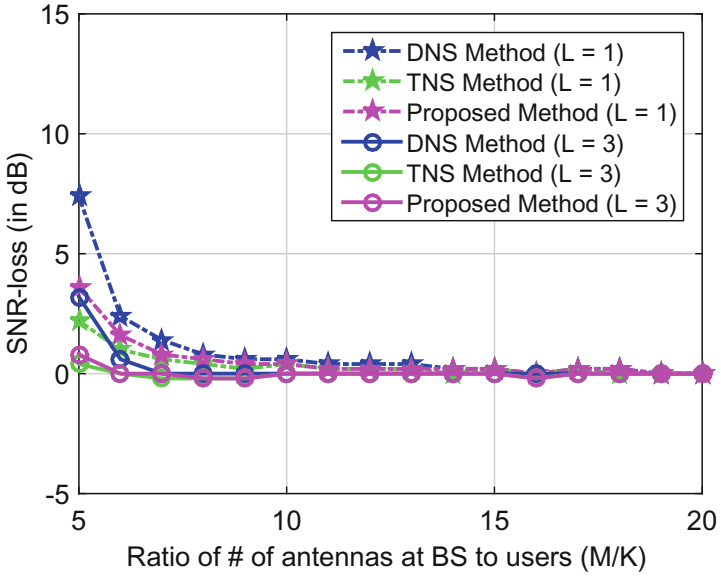


(a) SNR loss relative to exact ZF pre-coding to achieve BER of 10^3

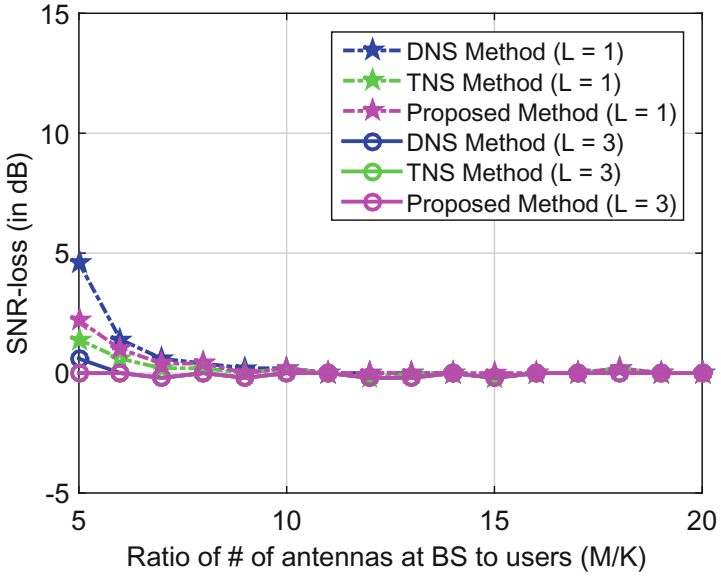


(b) SNR loss relative to exact RZF pre-coding to achieve BER of 10^3

Fig. 7. Number of UTs ($K = 8$)

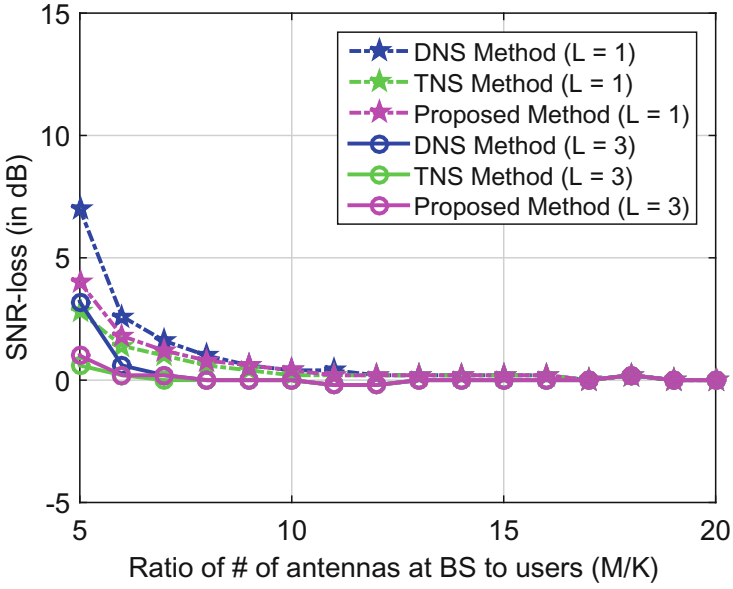


(a) SNR loss relative to exact ZF pre-coding to achieve BER of 10^3

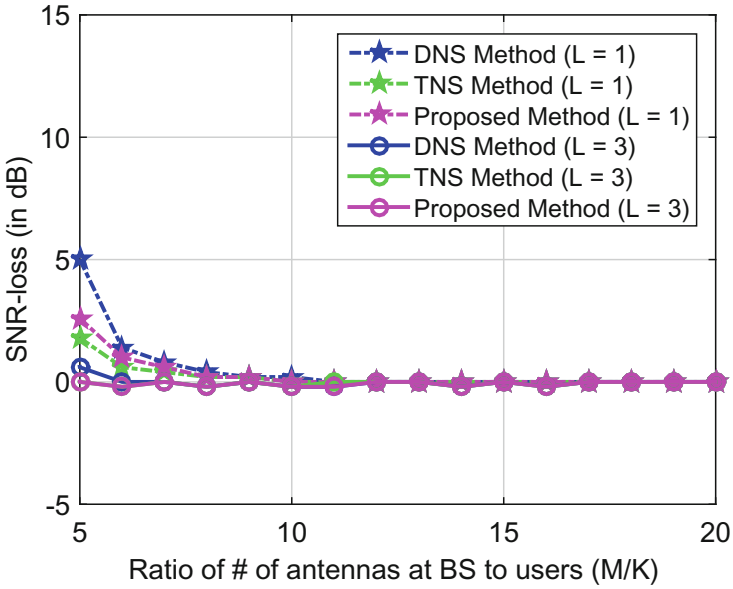


(b) SNR loss relative to exact Regularized Zero Forcing (RZF) pre-coding to achieve BER of 10^3

Fig. 8. Number of UTs ($K = 12$)



(a) SNR loss relative to exact ZF pre-coding to achieve BER of 10^3



(b) SNR loss relative to exact RZF pre-coding to achieve BER of 10^3

Fig. 9. Number of UTs ($K = 16$)

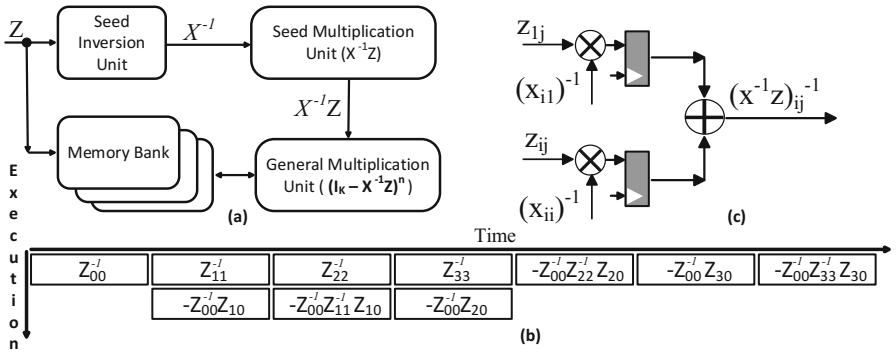


Fig. 10. proposed VLSI architecture (a) Block diagram of proposed inversion method (b) Seed multiplication unit $(X^{-1}Z)$ (c) Scheduling diagram for seed inversion (X^{-1})unit

6 Proposed Architecture

In this section, a VLSI architecture which performs the proposed matrix inversion with low latency and high throughput is presented. Figure 10(a) shows the top level block diagram of the architecture. After extracting the seed matrix (X) from the gram matrix Z, the seed is inverted in a seed inversion block. The inverted seed matrix (X^{-1}) is then multiplied with the gram matrix Z, to obtain the product $(X^{-1}Z)$, and a generic multiplication unit computes the powers of the product $(I_K - X^{-1}Z)^n$. For the best case scenarios with higher β values, only the first two terms of the Neumann series are used for approximation, i.e., $Z^{-1} = X^{-1} + (I_K - X^{-1}Z)X^{-1}$. For this case, the generic matrix multiplication unit is not required and can be bypassed, resulting in much lower latency and higher throughput.

6.1 Seed Inversion Unit (X^{-1})

Fig. 10(b) shows the scheduling diagram of the seed inversion (X^{-1}) for a 4×4 matrix, example presented in Sect. 4, using a single reciprocal unit and a multiplier. The overall latency for this inversion is $2(K–1)$, which is one third of the latency for inverting a tri-diagonal matrix $(6K)$ using a modified Gaussian elimination algorithm [9]. For the reciprocal calculation (Z_{ii}^{-1}), a standard unrolled single Newton Raphson iteration, similar to that used in [8,9], is employed. As discussed in the previous section, the latency of the proposed seed inversion can be further reduced by deploying a greater number of multipliers and reciprocal units, since there is no dependency, whereas the modified Gaussian elimination algorithm [9], due to the dependencies, may not be suitable for parallel implementation.

6.2 Seed Multiplication Unit ($X^{-1}Z$)

Figure 10 (c) shows a simple circuit that consists of two multipliers and one adder for the multiplication of $X^{-1}$and Z. z_{ij} is multiplied by x_{ii} and then added to the product of z_{1j} and x_{1i} to obtain the value of y_{ij} ($Y = X^{-1}Z$), where z_{ij}, x_{ij}, and y_{ij} are the values of the elements at the i^{th} row and the j^{th} column of the matrices Z, X^{-1}, and Y, respectively. This multiplication unit has a latency of two cycles.

6.3 Generic Multiplication Unit ($I_K - X^{-1}Z)^n$

For lower values of β, more terms of the Neumann series are required for better accuracy of inversion. Hence we need to compute higher powers of the product $(I_K - X^{-1}A)^n$. Generic MAC (multiply and accumulate) banks are used to compute the higher powers of $(I_K - X^{-1}A)^n$. The matrix multiplication is of cubic order complexity, $O(K^3)$, where K is the size of the matrix. As discussed in [9], parallel hardware can be used to speed up this calculation. Let α be a parallelization factor,. Then the total latency of the multiplication unit is reduced to $\frac{(L-1)K^3}{\alpha}$.

Table 1. Latency Comparision

	TNS [9]	Proposed method
Seed Inversion (X^-1)	6K	2(K-1)
Generic Multiplication $(I - X^{-1}Z)^{L-2}(\alpha = 10)$	$\frac{(L-1)K^3}{\alpha}$	$\frac{(L-1)K^3}{\alpha}$
Latency with Higher β ($L = 1$)	6K	2(K-1)
Latency with Lower β ($L = 3$)	$6K + \frac{(L-1)K^3}{\alpha}$	$2(K-1) + \frac{(L-1)K^3}{\alpha}$

7 Timing Analysis

Table 1 summarizes the latency comparison with the TNS [9] for high and low β values under the same hardware complexity and α value (parallelization factor). For higher values of β (e.g. $\beta > 10$), using only the first two terms of the Neumann series is sufficient ($L = 1$), and hence the generic multiplication module can be bypassed and no matrix multiplications $(I_K - X^{-1}A)^{L-1}$ are required. Therefore the total latency for both the TNS and the proposed method is comprised of seed inversion latencies. It can be seen that the latency of the proposed seed matrix is three times smaller than that of the TNS. In fact, if more hardware is available, the latency reduction is even higher.

For lower values of β, more Neumann series terms are required, and $\frac{(L-1)K^3}{\alpha}$ cycles are added to the final latency. If α (parallelization factor) is small, the latency is dominated by that of the generic multiplication unit, and hence total latency for both the TNS and proposed method will be similar.

Table 2. ASIC implementation of matrix inversion

	Direct inverse		Approximate inverse			
	DMI [5]	BMI [6]	TNS [9]		Proposed method	
			Higher β (L=1)	Lower β (L=3)	Higher β (L=1)	Lower β (L=3)
Order	4 X 4	8 X 8	16 X 16		16 X 16	
Technology	0.25 μm	90 nm	65 nm			
Gate Count	73K	90K	104K		117K	
Max. Freq. (MHz)	170	500	420		*448*	
Throughput	1.72M	0.65M	4.37M	0.51M	15M	0.54M
Normalized Throughput[a]	0.016	0.016	1.04	0.12	3.35	0.12
Normalized Hardware Efficiency[b]	0.22	0.18	10	1.15	28.6	1.02

[a] Normalized Throughput for K = 16 at 100 MHz = $\frac{Throughput \times Order^3 \times 100}{Frequency \times K^3}$

[b] Normalized Hardware Efficiency = $\frac{Normalized\ Throughput}{Gate\ Count}$

8 Implementation Results

The VLSI architecture for inverting a 16×16 matrix using the proposed seed matrix was designed and synthesized in TSMC 65-nm technology. 16-bit internal precision is used for the datapath of the architecture. The implementation is compared with the architecture using direct matrix inversion [5,6] and the TNS [9]. The comparison results are summarized in Table 2. It can be seen that the proposed architecture can sustain a maximum clock frequency of 448 MHz, with an area cost of 117 K gates.

For a MIMO system with $K = 16,\ \beta = 5,\ \alpha = 10$ and $L = 3$ the latencies for the proposed method and TNS are 850 cycles and 916 cycles, respectively. For such cases with lower values of β, as mentioned in Sect. 7, the latency is dominated by that of the generic multiplication unit. Due to overlapped scheduling, the latency for the seed inversion unit can be absorbed into the generic multiplication unit, and hence the generic multiplication unit is the main source of latency for both the TNS and proposed method at lower values of β. Therefore a throughput of 0.54M inversions per second is achieved for the proposed method operating at 448 MHz, while the TNS method operating at 420 MHz will have a throughput of 0.51M inversions per second.

Although it seems that for lower values of β, the TNS and proposed scheme have similar throughput and hardware efficiency. However we would like to mention that this is mainly due to lower values of the parellization factor α used in the generic multiplication unit, due to which the overall latency is dominated by that of the generic multiplication unit.

However if a higher paralleziation factor of $\alpha > 100$ is used, the latency will be dominated by the seed multiplication unit instead of the generic multiplication unit. Since the proposed seed inversion has lower latency than the TNS, even for lower values of β, the proposed scheme will result in higher throughput than the TNS and better inversion accuracy than the DNS.

For MIMO systems with higher β values, the generic multiplication unit can be bypassed, and the latency for the proposed method and TNS are reduced to 30 cycles and 96 cycles, respectively, resulting in a throughput of 15M and 4.37M inversions per second, respectively.

9 Conclusion

In this work, we have developed a low-latency and high-throughput matrix inversion method for the linear pre-coder in massive MIMO. The inversion method is based on Neumann series expansion, and a new seed matrix is used as an initial approximation for the Neumann series, which gives better performance, a lower complexity matrix inverse, and lower latency. Detailed latency and throughput analysis is presented for high and low values of β. A high-throughput VLSI architecture for inverting a 16×16 matrix using the proposed method is also presented.

References

1. Rusek, F., Persson, D., Lau, B.K., Larsson, E.G., Marzetta, T.L., Edfors, O., Tufvesson, F.: Scaling up MIMO: opportunities and challenges with very large arrays. IEEE Signal Process. Mag. **30**(1), 40–60 (2013)
2. Larsson, E.G., Edfors, O., Tufvesson, F., Marzetta, T.L.: Massive MIMO for next generation wireless systems. IEEE Commun. Mag. **52**(2), 186–195 (2014)
3. Hoydis, J., ten Brink, S., Debbah, M.: Massive MIMO in the UL/DL of cellular networks: how many antennas do we need? IEEE J. Sel. Areas Commun. **31**(2), 160–171 (2013)
4. Gao, X., Edfors, O., Rusek, F., Tufvesson, F.: Linear pre-coding performance in measured very-large MIMO channels. In: Proceedings of IEEE Vehicular Technology Conference (VTC Fall), Sept 2011
5. Burg, A., Haene, S., Perels, D., Luethi, P., Felber, N., Fichtner, W.: Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2006), May 2006
6. Wu, D., Eilert, J., Liu, D., Wang, D., Al-Dhahir, N., Minn, H.: Fast complex valued matrix inversion for multi-user STBC-MIMO decoding. In: Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2007), Porto Alegre (2007)
7. Stewart, G.: Matrix Algorithms, Basic decompositions (1998)
8. Wu, M., Yin, B., Vosoughi, A., Studer, C., Cavallaro, J.R., Dick, C.: Approximate matrix inversion for high-throughput data detection in the large-scale MIMO uplink. In: Proceedings of IEEE International Symposium Circuits and Systems (ISCAS 2013), May 2013
9. Prabhu, H., Edfors, O., Rodrigues, J., Liu, L., Rusek, F.: Hardware efficient approximative matrix inversion for linear pre-coding in massive MIMO. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2014), June 2014
10. Marzetta, T.L.: Noncooperative cellular wireless with unlimited numbers of base station antennas. IEEE Trans. Wireless Commun. **9**(11), 3590–3600 (2010)

11. Peel, C.B., Hochwald, B.M., Swindlehurst, A.L.: A vector-perturbation technique for near-capacity multiantenna multiuser communication-part I: channel inversion and regularization. IEEE Trans. Commun. **53**(1), 195–202 (2005)
12. Yoo, T., Goldsmith, A.: On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming. IEEE J. Sel. Areas Commun. **24**(3), 528–541 (2006)
13. Mehana, A.H., Nosratinia, A.: Diversity of MIMO linear precoding. IEEE Trans. Inf. Theory **60**(2), 1019–1038 (2014)
14. Biglieri, E., Proakis, J., Shamai, S.: Fading channels: Information-theoretic and communications aspects. IEEE Trans. Inf. Theory **44**(6), 2619–2692 (1998)
15. Scaglione, A., Stoica, P., Barbarossa, S., Giannakis, G., Sampath, H.: Optimal designs for space-time linear precoders and decoders. IEEE Trans. Signal Process. **50**(5), 1051–1064 (2002)
16. Jayaweera, S., Poor, H.: Capacity of multiple-antenna systems with both receiver and transmitter channel state information. IEEE Trans. Inf. Theory **49**(10), 2697–2709 (2003)
17. Joham, M., Utschick, W., Nossek, J.: Linear transmit processing in MIMO communications systems. IEEE Trans. Signal Process. **53**(8), 2700–2712 (2005)
18. Abbas., S.M., Tsui., C.Y.: Low-latency approximate matrix inversion for high-throughput linear pre-coders in massive MIMO. In: 2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Tallinn, pp. 1–5 (2016)

A Novel Hardware-Oriented Stereo Matching Algorithm and Its Architecture Design in FPGA

Yanzhe Li[1(✉)], Kai Huang[1], and Luc Claesen[2]

[1] Institute of VLSI Design, Zhejiang University, Hangzhou, China
{liyz,huangk}@vlsi.zju.edu.cn
[2] Engineering Technology - Electronics-ICT Department, Hasselt University, 3590 Diepenbeek, Belgium
luc.claesen@uhasselt.be

Abstract. Stereo matching is a crucial step to extract depth information from stereo images. However, it is still challenging to achieve good performance in both speed and accuracy for various stereo vision applications. In this contribution, a hardware-compatible stereo matching algorithm is proposed and its associated hardware implementation is also presented. The proposed algorithm can produce high-quality disparity maps with the combined use of the mini-census transform, segmentation-based adaptive support weight and effective refinement. Moreover, the proposed architecture is optimized as a fully pipelined and scalable hardware system. Implemented on an Altera Stratix-IV FPGA board, it can achieve 65 frames per second (fps) for 1024×768 stereo images and a 64 pixel disparity range. The proposed system is evaluated on the Middlebury benchmark and the average error rate is 6.56%. The experimental results indicate that the accuracy is competitive with some state-of-the-art software implementations.

Keywords: Stereo matching · Hardware implementation · Real-time · High-quality

1 Introduction

Stereo vision is one of the most active research topics in computer vision and it is widely used in many applications. Just recently, three dimensional television (3DTV) and virtual reality gaming have become popular. They provide the audience with a greater sense of presence in a computer-generated environment. However, the requirement to wear additional eyeglasses is usually perceived uncomfortable. In order to overcome the problem, autostereoscopic displays are utilized to support glasses-free 3D depth perception. In this way, multiple views are shown simultaneously so that the audience always sees a stereo pair from predefined viewpoints regardless of his position. These multiple views need to be generated by using depth-image based rendering (DIBR) from the original views and their corresponding depth maps. Each depth map gives information about

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 213–232, 2017.
DOI: 10.1007/978-3-319-67104-8_11

the distance between the camera and the objects in the 3D scene. Here the depth maps can be extracted in stereo vision systems. Real-time depth image generation is also important in future advanced driver assistance systems (ADAS) as well as self driving cars. Two or more cameras can assist in the distance calculation of other traffic, vehicles and objects while driving. In comparison to "Time of Flight" cameras, stereo cameras can be used over much larger distances, and can be used under various intensities of the environment (e.g. sun light).

Stereo matching, which is treated as the key operation in a stereo vision system, takes a pair of rectified images, estimates the movement of each pixel between two images and displays the associated movement in a disparity map. The depth of a pixel is inversely proportional to the disparity of this pixel. As a result, stereo matching is a complicated and time-consuming procedure. Considering that many applications often require high performance and real-time processing speed, it is difficult for software implementations of stereo matching algorithms on a CPU to meet these constraints. In this condition, hardware acceleration of stereo matching algorithms is inevitable and it has been done extensively using DSPs, GPUs and dedicated hardware. However, DSPs are limited by the computational ability and fail to support real-time processing; while GPUs always result in excessive power consumption for embedded applications. In contrary, the dedicated hardware approaches using FPGAs and ASICs can provide a balance between computational power and energy efficiency.

In the presented research work, the mini-census and segmentation-based ADSW algorithms are combined to achieve a high matching accuracy in depth discontinuity regions. Different from many other hardware designs that lack refinement, a disparity refinement with segmentation information is presented. This refinement step can significantly improve the quality of initial disparity maps in textureless and occluded regions. Moreover, a fully pipelined and scalable architecture is implemented based on the proposed algorithm. In order to make a tradeoff between accuracy and speed, some techniques such as a simplified weight function and an adaptive window size are applied. A prototype of the proposed hardware system is built on an Altera FPGA board, which achieves 65 fps for 1024×768 stereo images and a 64 pixel disparity range. The system is evaluated on the Middlebury benchmark and the visual satisfactory results are derived. The experimental results indicate that the proposed system has the top-performing processing ability and its accuracy is competitive with state-of-the-art software implementations.

In the rest of this contribution, Sect. 2 reviews the background of stereo matching algorithms and some related work. Section 3 presents the proposed algorithm. In Sect. 4, the hardware implementation based on the proposed algorithm is described. Section 5 presents experimental results and compares them with previous methods. Finally, Sect. 6 concludes the contribution.

2 Background and Related Work

2.1 Stereo Matching Background

Stereo matching algorithms aim to establish correspondence between a pair of images. This requires a pixel-by-pixel search through the whole image, consuming a large amount of computation power. To solve the problem, camera calibration and image rectification are used as preprocessing steps for most stereo matching algorithms. The preprocessing steps project each image to a common image plane and align each epipolar line to a common axis. In this way, stereo matching is reduced to a 1D search problem along the same horizontal scanline of the image pair.

Given two calibrated and rectified images, stereo matching can be addressed by searching for the corresponding pixel in the right image for each pixel in the left. To make the results more reliable, a support region is built for each pixel and the matching process is carried out over these regions instead of pixel by pixel. For a pixel $P(x, y)$ in the left image, its corresponding pixel $P'(x + d, y)$ is searched on the same horizontal line in the right image, where $0 \leq d < D_{max}$, D_{max} is the largest search distance and d is called the disparity. The matching costs are calculated for each pixel pair in the support regions and then aggregated. The smaller the aggregated matching cost is, the more similar the support regions are. Thus, the corresponding pixel is defined as the anchor pixel in the support region with the minimal aggregated matching cost.

2.2 Related Work

Nowadays, stereo matching algorithms can be divided into two groups: local approaches and global approaches [1]. Since local approaches only utilize local information, the accuracy is usually not sufficient in textureless and occluded regions. On the other hand, while global approaches can show better results, they are not yet suitable for real-time implementations due to their high computation complexity [2].

Global approaches usually compute disparities based on a global cost optimization. Dynamic programming (DP) [3] is a technique that optimizes disparity maps on a scanline in an efficient manner. Belief propagation (BP) [4] is a global approach that has attracted much attention. It gathers information from neighboring pixels and incorporates the information to update a smoothness term, then iteratively optimizes the smoothness term to achieve global energy minimization. Another popular technique explored by global approaches is graph cut [5]. Its energy function presents three terms: the data term that represents the difference between two corresponding pixels, the smoothness term that makes neighboring pixels tend to have similar disparities and the occlusion term, which imposes a penalty for making a pixel occluded. Although global approaches provide impressive accuracy results, the real-time implementations for high resolution images are challenging due to their computational complexity.

Another type of stereo matching algorithms is the class of local approaches, which compute disparities at a given point within a finite window. Early works on local approaches evaluate the impact of different similarity measures [2]. Common window-based matching costs include the sum of absolute or squared differences (SAD/SSD), normalized cross correlation (NCC), census transforms and mutual information [6]. Another important research topic that has been studied is that of the support regions. The early conventional approach is to use fixed-size square windows, which is easy to implement but suffers from severe artifacts. To remedy this, variable window size [7] is developed and it can improve the disparity quality in textureless regions. A recent development with promising results is to adapt the support weights in fixed-size windows. An adaptive support weight (ADSW) algorithm [8] is proposed, which assigns different weights to the pixels in a support window based on the proximity and color distances to the center pixel. A segment support algorithm [9] assigns fixed weights to the pixels in the same segment as the center pixel is in, and assigns weights to the pixels outside the center pixels segment according to the color similarity between the outside pixel and the center pixel. The disparity quality of [8,9] is comparable to some of the complex global algorithms.

Compared to global approaches, local approaches are more suitable for dedicated hardware implementation such as FPGAs and ASICs because of their low computation complexity and storage requirement. A segmentation-based design with adaptive support weight (ADSW) has been implemented on FPGAs [10]. Their proposed design can achieve 30 fps for 640 × 480 images using a disparity range of 64 pixels. This design is inspired by the algorithm in [9], which used to be the best local method on the Middlebury benchmark [11]. However, the performance of the design is restricted by the small fixed window size. In [12], a hardware solution provides high-quality disparity results in ASICs based on the mini-census adaptive support weight (MCADSW) method. But this solution only targets low resolution images. Its performance drops to 6 fps for 1024 × 768 images and a 64 pixel disparity range. In [13], an algorithm is proposed to achieve high accuracy based on mini-census and variable-cross methods and a fully pipelined architecture is presented for real-time processing. The design can process 1024 × 768 images with a disparity range of 64 pixels in 60 fps. A. Akin proposes a hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time hardware implementation [14]. It can handle 60 fps at a 1024 × 768 resolution for a 128 pixel disparity range. Although the results in [13,14] are outstanding among hardware implementations, the accuracy is not comparable to state-of-the-art software implementations.

3 Stereo Matching Algorithm

3.1 Algorithm Overview

In local stereo matching algorithms, cost calculation, cost aggregation, disparity selection and disparity refinement are four well-defined steps [1]. Since the proposed stereo matching algorithm belongs to local approaches, the mini-census

transform is used in the cost calculation step; the segmentation-based ADSW algorithm is used in the cost aggregation step; and a tree-structure winner-takes-all (WTA) method is used in the disparity selection step. The last step, disparity refinement, consists of three stages: consistency check, disparity voting and invalid disparity inpainting. The flow of the proposed algorithm is illustrated in Fig. 1. Two images are operated and the corresponding disparity maps are generated simultaneously.

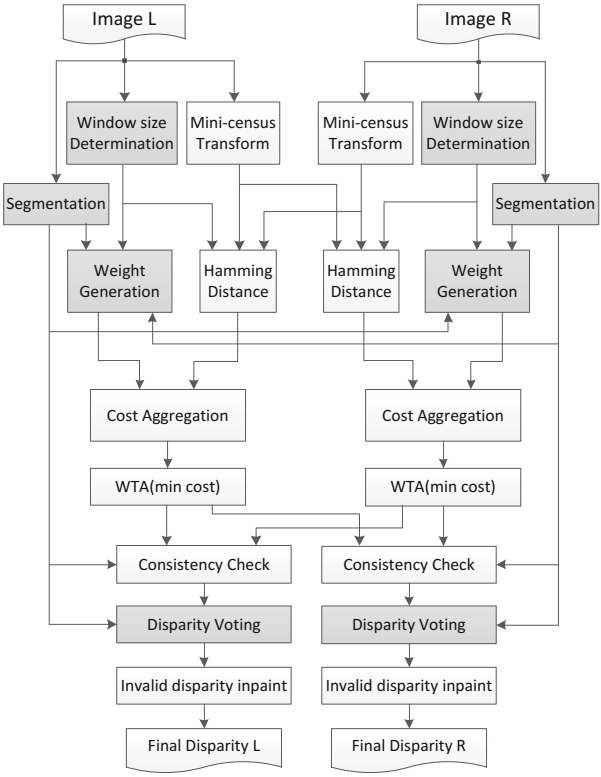


Fig. 1. Overview of the proposed algorithm.

In the cost calculation step, the mini-census transform is a hardware-friendly census transform, which makes the matching cost robust to brightness bias and exposure gain. It extracts a 6-pixel neighborhood information of the center pixel within a support window and encodes the information into a vector. If a pixels luminance is larger than the center pixels, a label 0 will be assigned to the pixel. Otherwise, a label 1 will be assigned [12]. In this way, each pixel can be represented by only 6 bits, which results in a reduction of the memory utilization due to fewer storage bits. Then the matching cost is defined as the Hamming distance between output vectors.

In the cost aggregation step, the segmentation-based ADSW algorithm employs the segmentation information within the weight cost function to increase the robustness of the matching process. Rather than only relying on colour and proximity, the use of segmentation takes the relationship between pixels and the shape of the segments into account. It assumes that each pixel on the same segment of the center pixel has a similar disparity value, and its weight is equal to the maximum value of the range [9]. The weight coefficients w_r and w_l are defined as

$$w_{r,l} = \begin{cases} 1.0 & p_i \in S_c \\ \exp(-\frac{d_c(I_{r,l}(p_i), I_{r,l}(p_c))}{\gamma_c}) & \text{otherwise} \end{cases} \tag{1}$$

where S_c is the segment of the central point, d_c is the Euclidean distance between two triplets in the CIELAB color space, and γ_c is a fixed parameter in the algorithm. The final aggregated cost is calculated by summing up all the weighted matching costs in the support windows W_r and W_l, and then normalized with the sum of weight coefficients

$$C(p_c, q_c) = \frac{\sum\limits_{p_i \in W_r, q_i \in W_l} w_r(p_i, p_c) w_l(q_i, q_c) MC(p_i, q_i)}{\sum\limits_{p_i \in W_r, q_i \in W_l} w_r(p_i, p_c) w_l(q_i, q_c)} \tag{2}$$

where p_c and q_c are the central points of W_r and W_l, respectively. The cost aggregation step is executed for all disparity levels and a number of aggregated costs are produced.

In the disparity selection step, a tree-structure WTA method is used to pick the disparity with the minimum aggregated cost, as depicted in Fig. 2. The aggregated costs for the whole disparity range are arranged into groups. Here, the disparity range and the size of a group are defined as 64 and 4, respectively. For each group, the smallest value and the corresponding position are selected and stored. Then after several times iteration, the minimum value among groups is finally detected; its position is selected as the disparity result.

The advantage of using a tree-structure method is not only to reduce the complexity of the search operation. It also fits the dataflow within FPGAs very well. Thus, it can be highly pipelined and the throughput increased up to one disparity range per clock cycle.

In the disparity refinement step, the initial disparity maps are operated. The consistency check is used to check whether disparity maps are valid or not. With the help of segmentation information, a left-to-right consistency check is expressed as

$$V_p = (d_p(x, y) == d'_p(x - d_p(x, y), y)) \ \& \ (S_p == S'_p). \tag{3}$$

In the process of left-to-right consistency check, for each pixel p in the left image, the corresponding pixel p' in the right image is determined by the disparity d_p. Then the right image's disparity d'_p and segmentation S'_p will be compared to d_p and S_p of the left image, respectively. If the expression is calculated as false, the

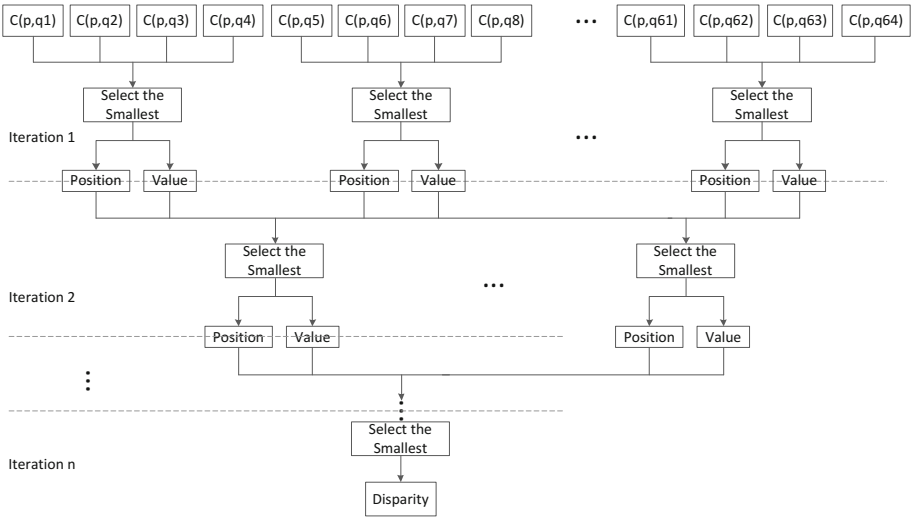


Fig. 2. Tree-structure WTA method.

left-to-right consistency check will fail and the disparity will be marked as invalid. It is noted that this process can also be utilized for a right-to-left consistency check.

After the consistency check, the disparity voting will update the center disparity based on the most frequent valid disparity in its local support window. This is because adjacent pixels that belong to the same object in an image should share the same disparity. Although the disparity voting helps to remove many invalid disparities, it will fail if the window does not contain any valid disparities. In order to address this problem, the disparity inpainting will replace the invalid disparity with the closest valid disparity on its scanline so as to get the final disparity maps. It is worthy to note that the median filtering is not used in the design because of its complicated hardware implementation but limited quality improvement.

3.2 Hardware-Oriented Optimization

To reduce the computation complexity and improve the hardware compatibility of the algorithm, some optimizations are proposed in this subsection and will be applied to the shaded blocks in Fig. 1. These hardware-oriented optimizations affect the accuracy of the final disparity maps slightly.

In the window size determination block, a method called AWDE [14] is introduced to make a tradeoff between accuracy and speed. It uses three different window sizes for different textures on the image; the window size is determined by the mean absolute deviation (MAD) of the pixel in the center of a 7×7 block, which is expressed as

$$MAD(c) = \frac{\sum_{q \in N_c} |I_t(q) - I_t(c)|}{48}. \tag{4}$$

A high MAD value indicates a high texture content, while a low MAD value is a sign of a low texture content. As expressed in (5), a 7×7 window is used if the MAD of the center pixel is high, and a 25×25 window is used if the MAD is very low.

$$\text{window size} = \begin{cases} 7 \times 7 & MAD(c) > th_7 \\ 13 \times 13 & th_{13} < MAD(c) \leq th_7 \\ 25 \times 25 & MAD(c) \leq th_{13} \end{cases} \tag{5}$$

As a general rule, increasing the window size increases the hardware complexity. In order to provide constant hardware complexity over the three different window sizes, a total of 49 pixels are constantly sampled with different intervals for different window sizes. In this way, a low computation cost is required for large support window sizes.

In the segmentation block, the segmentation-based ADSW algorithm [9] uses mean shift segmentation. However, the computational complexity and memory requirements make it unsuitable for real-time applications. In our algorithm, the image is divided into segments using thresholding [10]; this method is simple and can be implemented in hardware efficiently.

In the weight generation block, in order to replace signed floating-point numbers with unsigned integers, the YUV color representation is adopted instead of the CIELAB color representation. In addition, only the luminance channel (Y) is used in the design to reduce the potential bandwidth and storage requirements. Rather than Euclidean distance, Manhattan distance is used to avoid square and square root computations. Furthermore, for the pixel whose luminance is similar to the center pixel in the support window, it should be allowed to have more influence on the final matching cost. Therefore, a scale-and-truncate approximation of the weight function is proposed, and the curve is shown in Fig. 3. As a result, the multiplication of the weight coefficients is reduced to a left shift operation.

In the disparity voting block, a local support window is applied to achieve a reliable result. A vertical-horizontal approach is used to efficiently determine the most frequent valid disparity in the window, as shown in Fig. 4. Here the numbered shaded squares indicate valid disparities, which are used to update the center disparity. First, the approach searches for the majority disparity vertically in each column. Then it searches for the majority disparity horizontally and finally selects it as the center disparity. To further reduce the computation complexity, the approach also reduces the internal bandwidth.

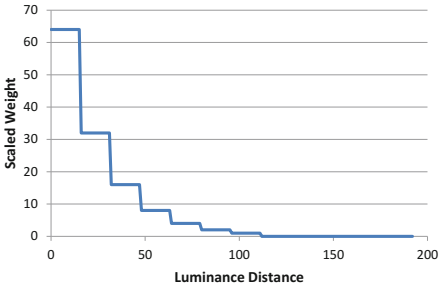


Fig. 3. Weight function.

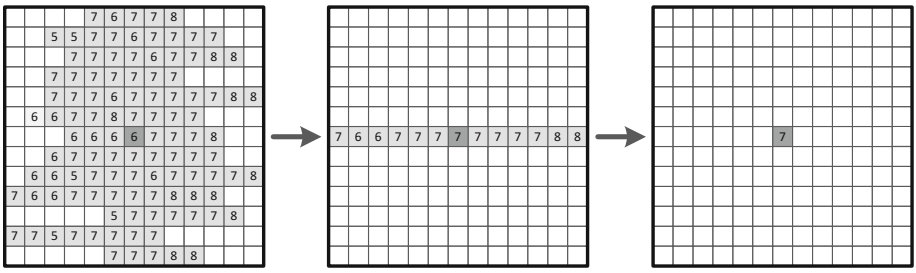


Fig. 4. The vertical-horizontal approach for disparity voting.

4 Hardware Implementation

4.1 Architecture Overview

In this section, a hardware architecture is designed based on the proposed algorithm. The whole system consists of three stages: pre-processing, stereo-matching and post-processing. First, in the pre-processing stage, pixel-based operations are performed on each pixel and the temporary results are stored into the line buffers. Then these temporary results are operated in the stereo-matching stage in order to calculate initial disparity maps. When the initial disparity maps are available, they will be refined in the post-processing stage. The top-level block diagram of the proposed hardware architecture is shown in Fig. 5; the implementation details of the three stages will be discussed in the following subsections.

One key feature of the proposed system is high processing ability. To achieve this goal, the architecture is designed to be fully pipelined without external memory limitation. All of the three stages are fully pipelined, i.e. source image pixels are fetched and operated in scanline order; initial disparity maps are generated in pipeline using a parallelism scheme; finally the disparity maps are refined in scanline order after a certain pipeline latency. External memory bandwidth is also an important limitation to the processing ability. In order to solve the problem in our design, each pixel is read only once from the external memory during

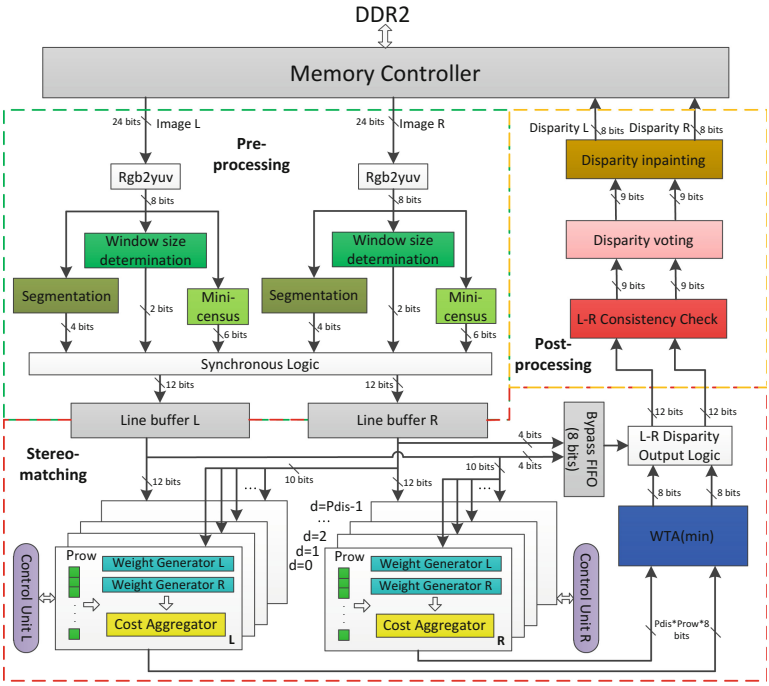


Fig. 5. Block diagram of the proposed architecture. (Color figure online)

the whole processing flow. In this case, dual-port BRAMs are used as buffers to store original pixel data and temporary intermediate results.

Another key feature of the proposed system is scalability. The design can be scaled with image resolution, disparity range and parallelism degree to achieve maximum flexibility. The image resolution is related to the user demand, while the disparity range is configured depending on the expected distance to the objects. Configuring the hardware for low image resolution and disparity range increases the processing speed. In contrast, high image resolution and disparity range lead to a high accuracy. The parallelism degree is used to indicate how many disparities are calculated in parallel, which can make a tradeoff between resource utilization and processing speed.

4.2 Pre-processing Stage

In the preprocessing stage, 24-bit source pixels of RGB will be fetched from the DDR2 memory once the system is started. The RGB pixels are treated as the input to the color space converters (rgb2yuv). Hereafter 8-bit Y values are generated from the converters and will be utilized to produce 12-bit temporary results in the three submodules.

As displayed in the green dotted box in Fig. 5, 4 bits of the 12-bit temporary result are generated in the segmentation module. In this module, the number of

segments k is given as input. A segmentation label is calculated using a simple method that multiplies the Y value by the value of $k/256$. Note that k is always defined as 16 in our system. In this way, the shift operation can be exploited instead of multiplication, and the label can be expressed within 4 bits. Another 6 bits come from the mini-census transform module. Here the center pixel is compared with its surrounding 6 pixels, and a 6-bit mini-census vector is obtained as the comparison result. The last 2 bits are produced in the window size determination module. The MAD of the center pixel in a 7×7 block is calculated, and then the 2-bit window size is assigned based on the MAD value. Since all the operations in the submodules are window-based, a register matrix is employed to provide pipelined processing, as shown in Fig. 6. The whole register matrix of 7×7 is used for window size determination; the 6 green registers are used for the mini-census transform; the red one in the center is used for segmentation. The Y values will be shifted from the left to the right in the register matrix per clock cycle. Meanwhile, a total of 12 bits will be written into the line buffer as the temporary result for each pixel.

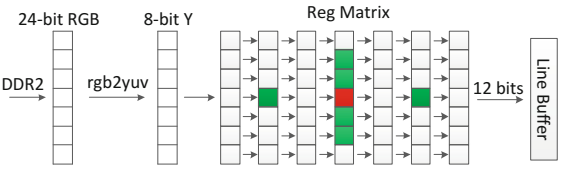


Fig. 6. Pipeline architecture in the pre-processing stage. (Color figure online)

4.3 Stereo-Matching Stage

In the stereo-matching stage, it is challenging to develop an efficient parallelism scheme for cost aggregation due to the requirement for real-time processing speed. Many hardware systems calculate all the disparities in parallel and process pixel by pixel. It is simple to implement but inefficient. In our system, a hybrid parallelism scheme [15] is adopted. It combines the row-level parallelism P_{row} with the disparity-level parallelism P_{dis}. The row-level parallelism means that P_{row} pixels in neighboring rows are processed in parallel, and the disparity-level parallelism means that P_{dis} disparities are processed for each pixel. Thus, the parallelism degree is $P_{row} \times P_{dis}$ in the proposed system. As shown in the red dotted box in Fig. 5, a total of $P_{dis} \times 2$ aggregation modules are generated to deal with different disparities for both images, while P_{row} pixels are processed in each aggregation module. Here the generate statement in Verilog is used for these two parameters to make the hardware architecture scalable.

To satisfy the row-level parallelism, P_{row} pixels along the column direction are processed in parallel. Therefore the line buffer is composed of $(P_{row}+6)$ dual port BRAMs to build a wide throughout, and the size of the register matrix in the pre-processing stage is extended to $(P_{row}+6)\times 7$. Source data and temporary

results in each column of the matrix can be reused to reduce the computational requirements, because a column is usually a part of multiple horizontally overlapping windows.

In each aggregation module, the weight generators are utilized to generate weight coefficients using the 4-bit segmentation labels in the temporary results. The circuit of the weight generator is shown in Fig. 7; the look-up table (LUT) is a straightforward solution that consumes a low amount of hardware resources. Meanwhile in the cost aggregator, 49 Hamming distances are generated as matching costs between corresponding pixels in the support windows W_r and W_l. These matching costs are shifted by the corresponding weight coefficients w_r and w_l. The final aggregated cost is calculated by summing the weighted costs using a tree adder, then dividing it by the sum of weight coefficients. The architecture of the cost aggregator is illustrated in Fig. 8. Then, the disparity with the minimum aggregated cost is selected in the WTA module. In addition, the Bypass FIFO is used to store the segmentation label of each pixel for the next stage. The output of the stereo-matching stage is a data stream that consists of the initial disparity maps and their corresponding segmentation labels.

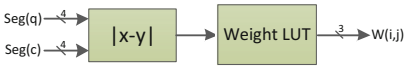


Fig. 7. Circuit of the weight generator.

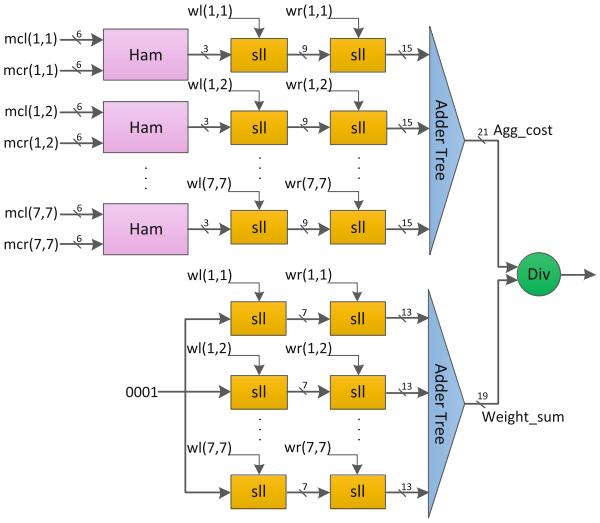


Fig. 8. Architecture of the cost aggregator.

4.4 Post-precessing Stage

In the post-precessing stage, three submodules work in pipeline to generate the final disparity maps, as shown in the yellow dotted box in Fig. 5. First, the initial disparity maps of both images are used to check the consistency of every pixel with the help of the segmentation information in the consistency check module. The module generates one more bit to label whether each disparity is valid or not.

Then in the disparity voting module, the disparities are updated in a 25×25 support window using the vertical-horizontal method. To enable a fully pipelined implementation, a bitwise fast voting technique [16] is applied to handle the most frequent valid disparity value. For each column, it drives each bit of the most frequent disparity independently from the other bits. In this way, the hardware cost depends on the number of the disparity bits in binary. It is noted that when counting bit votes, the valid information of each disparity must be taken into account. The architecture of bitwise fast voting for one column is shown in Fig. 9. Since the support window size is 25×25, the 25 most frequent disparities for the 25 columns are derived. The same technique is applied to the 25 derived disparities and finally the most frequent valid disparity in the support window is picked as the center disparity.

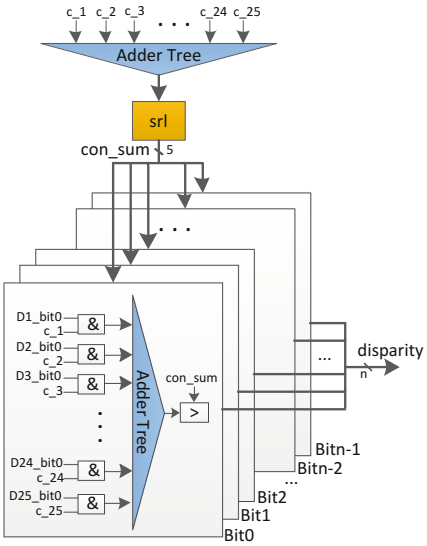


Fig. 9. Architecture of bitwise fast voting.

After the disparity voting, most invalid disparities will be updated to valid ones only by their valid neighbours. In the disparity inpainting module, the remaining invalid disparities are replaced with the closest valid ones. At the end, the refined disparity maps are written out in scanline order.

5 Experimental Results

A prototype of the proposed system has been implemented on an Altera EP4SGX-230 FPGA board. It is evaluated using rectified synthetic stereo images, initially stored in the DDR2 memory. In addition, the system is designed to be scaled with image resolution, disparity range and parallelism degree. Unless stated otherwise, the image resolution is defined as 1024 × 768, the disparity range as 64, the row-level parallelism P_{row} as 4, and the disparity-level parallelism P_{dis} as 8 in our design. The evaluation results of three important aspects — resource utilization, processing speed and quality evaluation — are elaborated in the following subsections.

5.1 Resource Utilization

Table 1 lists the detailed resource utilization of the FPGA prototype. The proposed system occupies 80% of the ALUTs, 58% of the registers, and 16% of the memory bits on the FPGA board, and can operate at 120 MHz. As shown in Table 1, the majority of the ALUTs and registers are consumed in the stereo-matching stage, which is mainly composed of the weight generators and the cost aggregators. So the resource utilization is mainly determined by the parallelism degree in the proposed system. The two parameters, the disparity-level parallelism P_{dis} and the row-level parallelism P_{row}, can be scaled to make the system more flexible. To make a tradeoff between resource utilization and processing speed, P_{dis} is 8 and P_{row} is 4 in the current system. On the other hand, the memory bits are mostly used as line buffers to store original pixel data and temporary results.

Table 1. Resource utilization report

Altera EP4SGX230	ALUTs total: 228000	Registers total: 228000	Memory bits total: 17133000
Pre-processing	4170	2286	1015808
Stereo-matching	158496	120292	1638400
Post-processing	18852	9684	28672
Whole system	181518	132262	2682880

5.2 Processing Speed

The processing speed of stereo matching is given by million disparity estimations per second (MDE/s), which is calculated by (image resolution × disparity range × frame rate). Table 2 presents a comparison between some exiting implementations and the proposed system. It is shown that CPU and GPU based implementations can hardly achieve real-time speed with high resolution images.

Table 2. Processing speed comparison

Design	Platform	Image size	Disparity range	FPS	MDE/s
Shan et al. [15]	FPGA	1280 × 1024	256	46	15437
MCADSR [17]	FPGA	1024 × 768	128	129	13076
AWDE-IR [14]	FPGA	1024 × 768	128	60	6040
Zhang et al. [13]	FPGA	1024 × 768	64	60	3019
Wang et al. [18]	FPGA	1024 × 768	96	31.8	2400
Ttofis et al. [10]	FPGA	640 × 480	64	30	589
MCADSW [12]	ASIC	352 × 288	64	42	272
AD-Census [19]	GPU	450 × 375	60	10.6	107
Yang et al. [20]	GPU	640 × 360	20	10	46
VariableCross [21]	CPU	450 × 375	60	0.63	13
SemiGlobal [22]	CPU	450 × 375	64	0.55	6
Proposed	FPGA	1024 × 768	64	65	6543

For FPGA based implementations, the achievable processing speed is usually limited by the available hardware resources, such as on-chip memories. The proposed system is able to achieve 65 fps for 1024 × 768 images with a disparity range of 64 pixels. Although the design in [15] has the highest processing speed, it is based on a simple SAD matching method that leads to low accuracy. Likewise, the system in [17] improves the accuracy with variable support regions, but its disparity quality is still worse than that of the proposed system.

5.3 Quality Evaluation

To discuss the quality of the proposed system, the disparity maps are evaluated based on the Middlebury benchmark using the percentage of bad pixels on different regions, a commonly accepted metric [1]. Table 3 lists the accuracy comparison with some state-of-the-art implementations. The average error rate of the final disparity maps in the proposed system is 6.56%. The first row shows the results of the AD-Census algorithm implemented on GPUs. However, it is challenging to realize it into an FPGA because of its multi disparity enhancement functions. The design in [18] utilizes cross-based regions and semi-global optimization on an FPGA, but its high accuracy is achieved at the expense of the decreased processing speed. The SegSupport algorithm outperforms the proposed design slightly but fails to reach real-time performance. The algorithms in [21,22], which are also software implementations, have a higher error rate than the proposed algorithm. To summarize, the comparison shows that the accuracy of the disparity maps is not only among the best in hardware accelerated stereo systems, but also competitive with state-of-the-art software implementations.

Table 3. Accuracy comparison on the middlebury benchmark

Data set	Tsukuba			Venus			Teddy			Cones			Average error rate
Evaluation	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
AD-Census [19]	1.07	1.48	5.73	0.09	0.25	1.15	4.10	6.22	10.9	2.42	7.25	6.95	3.97
Wang et al. [18]	1.93	2.95	7.90	0.61	1.43	2.87	6.44	13.8	16.0	2.37	11.1	6.70	6.17
SegSupport [9]	1.25	1.62	6.68	0.25	0.64	2.59	8.43	14.2	18.2	3.77	9.87	9.77	6.44
MCADSW [12]	–	2.80	–	–	0.64	–	–	13.7	–	–	10.1	–	all = 6.81
SemiGlobal [22]	3.26	3.96	12.8	1.00	1.57	11.3	6.02	12.2	16.3	3.06	9.75	8.90	7.50
VariableCross [21]	1.99	2.65	6.77	0.62	0.96	3.20	9.75	15.1	18.2	6.28	12.7	12.9	7.60
MCADSR [17]	3.62	4.15	14.0	0.48	0.87	2.79	7.54	14.7	19.4	3.51	11.1	9.64	7.65
Zhang et al. [13]	3.84	4.34	14.2	1.20	1.68	5.62	7.17	12.6	17.4	5.41	11.0	13.9	8.20
Ttofis et al. [10]	4.48	6.04	12.7	6.01	7.47	18.2	21.5	28.1	28.8	17.1	25.9	25.8	16.8
Result1[1]	9.86	11.3	19.3	5.44	7.64	17.9	10.3	19.3	22.4	4.88	15.3	12.3	13.0
Result2[2]	3.50	3.98	11.7	0.44	0.71	5.66	4.60	9.25	16.1	3.34	8.64	10.8	6.56

[1]The error rate of the initial disparity maps before the disparity refinement step.
[2]The error rate of the final disparity maps in the proposed system.

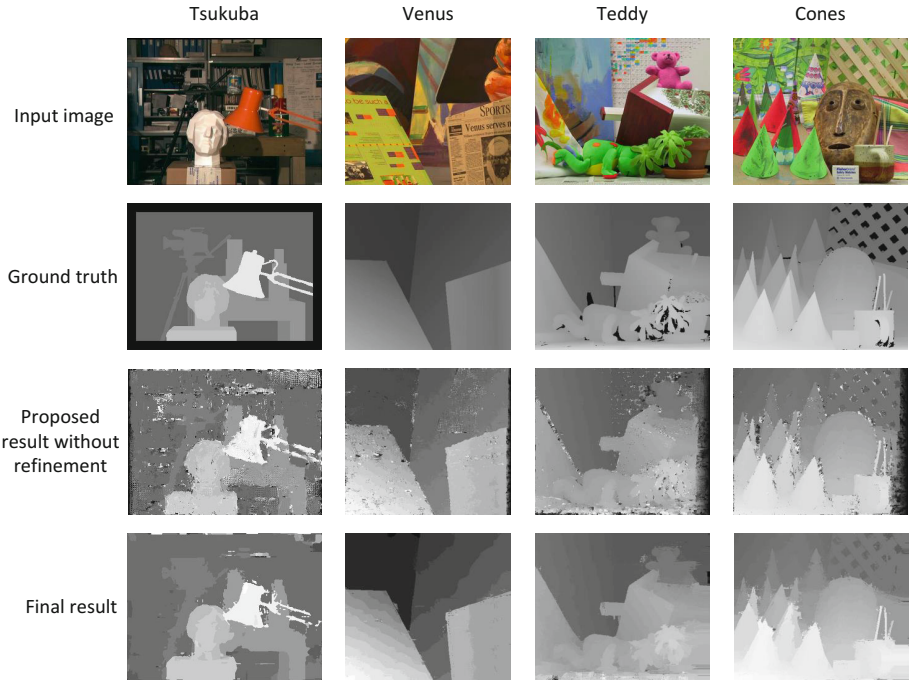


Fig. 10. True disparity maps and experimental results.

The disparity maps of the four data sets *Tsukuba, Venus, Teddy* and *Cones* are displayed in Fig. 10, and the final disparity maps are compared with the initial disparity maps to demonstrate the effect of the disparity refinement. Some of them are generated from the left images, and the others are generated from the

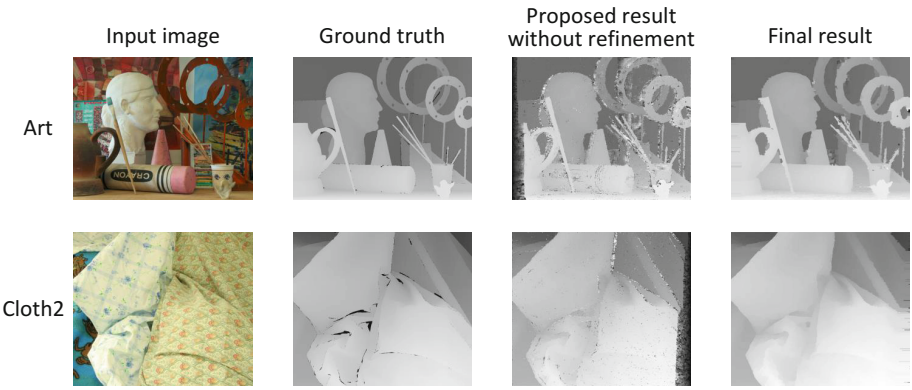


Fig. 11. Evaluation results on high-definition images.

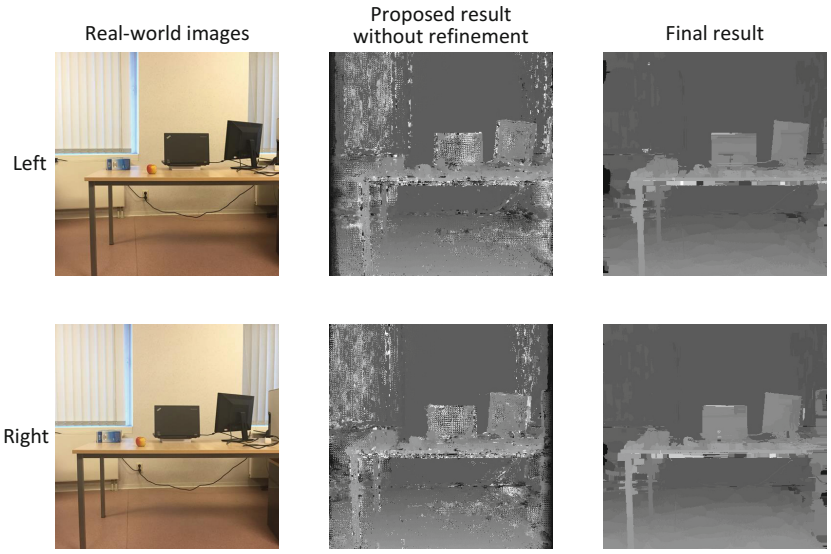


Fig. 12. Evaluation results on real-world images.

right images. In this way, the proposed system is comprehensively evaluated. It is observed that the refinement step contributes significantly to the final disparity maps and many visual improvements are obvious, including the elimination of speckle noise, fewer errors at the image borders and sharply delineated edges. The quantitative results in Table 3 also verify it.

The resolutions of the data sets *Tsukuba, Venus, Teddy* and *Cones* are all smaller than that of VGA. To further evaluate the proposed design, some high-definition images in the benchmark are used with a disparity range of 128 pixels. The results of the data sets *Art* and *Cloth2* captured at different viewpoints are

shown in Fig. 11. The overall error rates are 12.85% and 4.67%, respectively. The proposed system provides quite clear and smooth disparity maps and the accuracy is comparable to the low-definition results.

The reference images in the benchmark are all well captured and rectified so that the results are quite accurate. But the quality of the disparity maps for real-world images may decrease due to some undesirable factors, such as luminance differences and rectification errors. The proposed system is further evaluated by real-world images to prove its robustness. The images are captured by two adjacent cameras in a office, and then rectified by the toolbox in Matlab 2016b. Here the image resolution is 460×460, and the disparity range for the real-world images is defined as 45. It is noted that our system still provides high-quality disparity maps for the real-world images, as shown in Fig. 12.

6 Conclusion

This contribution has proposed a stereo matching algorithm based on the mini-census transform and the segmentation-based ADSW. The disparity refinement step with segmentation information has been presented and the quality of disparity maps has been improved significantly. Furthermore, a fully pipelined and scalable hardware architecture is designed with hardware-oriented optimizations. A prototype of the hardware system has been built on an Altera Stratix-IV FPGA board. The design is evaluated on the Middlebury benchmark and the average error rate is 6.56%. The experimental results have shown that our hardware system has the top-performing processing ability and its accuracy is competitive with state-of-the-art software implementations. In the future, we will introduce a part of global matching algorithms to achieve higher accuracy of disparity maps.

Acknowledgment. The research in this contribution was sponsored in part by the Belgian FWO (Flemish Research Council) and the Chinese MOST (Ministry of Science and Technology) bilateral cooperation project number G.0524.13.

References

1. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**, 7–42 (2002)
2. Hirschmuller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE Press, Minneapolis (2007)
3. Veksler, O.: Stereo correspondence by dynamic programming on a tree. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 384–390. IEEE Press, San Diego (2005)
4. Klaus, A., Sormann, M., Karner, K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: 18th International Conference on Pattern Recognition (ICPR 2006), pp. 15–18. IEEE Press, Hang Kong (2006)

5. Kang, S.B., Szeliski, R., Chai, J.: Handling occlusions in dense multi-view stereo. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 103–110. IEEE Press, Kauai (2001)
6. Hirschmuller, H.: Accurate and efficient stereo processing by semiglobal matching and mutual information. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 807–814, San Diego (2005)
7. Kanade, T., Okutomi, M.: A stereo matching algorithm with an adaptive window: theory and experiment. IEEE Trans. Pattern Anal. Mach. Intell. **16**, 920–932 (1994)
8. Yoon, K.-J., Kweon, I.-S.: Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 650–656 (2006)
9. Tombari, F., Mattoccia, S., Stefano, L.: Segmentation-based adaptive support for accurate stereo correspondence. In: Mery, D., Rueda, L. (eds.) PSIVT 2007. LNCS, vol. 4872, pp. 427–438. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77129-6_38
10. Ttofis, C., Theocharides, T.: Towards accurate hardware stereo correspondence: a real-time FPGA implementation of a segmentation-based adaptive support weight algorithm. In: Proceedings of the Conference on Design, Automation & Test in Europe, Conference and Exhibition (DATE), pp. 703–708. IEEE Press, Germany (2012)
11. Middlebury benchmark. http://vision.middlebury.edu/stereo/
12. Chang, N.Y.-C., Tsai, T.-H., Hsu, B.-H., Chen, Y.-C., Chang, T.-S.: Algorithm and architecture of disparity estimation with mini-census adaptive support weight. IEEE Trans. Circ. Syst. Video Technol. **20**, 792–805 (2010)
13. Zhang, L., Zhang, K., Chang, T.S., Lafruit, G., Kuzmanov, G.K., Verkest, D.: Real-time high-definition stereo matching on FPGA. In: 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 55–64. ACM Press, Monterey (2011)
14. Akin, A., Baz, I., Schmid, A., Leblebici, Y.: Dynamically adaptive real-time disparity estimation hardware using iterative refinement. Integr. VLSI J. **47**, 365–376 (2014)
15. Shan, Y., Wang, Z., Hao, Y., Wang, Y., Tsoi, K., Luk, W., Yang, H.: FPGA based memory efficient high resolution stereo vision system for video tolling. In: International Conference on Field-Programmable Technology (FPT), pp. 29–32. IEEE Press, Seoul (2012)
16. Zhang, K., Lu, J., Lafruit, G., Lauwereins, R., Gool, L.V.: Real-time accurate stereo with bitwise fast voting on CUDA. In: 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 794–800. IEEE Press, Kyoto (2009)
17. Shan, Y., Hao, Y., Wang, W., Wang, Y., Chen, X., Yang, H., Luk, W.: Hardware acceleration for an accurate stereo vision system using mini-census adaptive support region. ACM Trans. Embed. Comput. Syst. **13**, 1–24 (2014)
18. Wang, W., Yan, J., Xu, N., Wang, Y., Hsu, F.H.: Real-time high-quality stereo vision system in FPGA. In: International Conference on Field-Programmable Technology (FPT), pp. 358–361. IEEE Press, Kyoto (2013)
19. Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., Zhang, X.: On building an accurate stereo matching system on graphics hardware. In: 14th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 467–474. IEEE Press, Barcelona (2011)

20. Yang, Q., Li, D., Wang, L., Zhang, M.: Fast local stereo matching using two-level adaptive cost filtering. In: International Conference on Acoustics, Speech and Signal Processing, pp. 1986–1990. IEEE Press, Vancouver (2013)
21. Zhang, K., Lu, J., Lafruit, G.: Cross-based local stereo matching using orthogonal integral images. IEEE Trans. Circ. Syst. Video Technol. **19**, 1073–1079 (2009)
22. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 328–341 (2008)

Author Index