Handbook of Hardware/Software Codesign

Soonhoi Ha • Jürgen Teich
Editors

Handbook of Hardware/Software Codesign

With 575 Figures and 56 Tables

Springer

Editors
Soonhoi Ha
Department of Computer
Science and Engineering
Seoul National University
Seoul, Korea

Jürgen Teich
Department of Computer Science
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)
Erlangen, Germany

ISBN 978-94-017-7266-2 ISBN 978-94-017-7267-9 (eBook)
ISBN 978-94-017-7268-6 (print and electronic bundle)
https://doi.org/10.1007/978-94-017-7267-9

Library of Congress Control Number: 2017947685

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer Science+Business Media B.V.
The registered company address is: Van Godewijckstraat 30, 3311 GX Dordrecht, The Netherlands

Foreword

Hardware/software codesign means to achieve system-level design objectives by leveraging the synergy between hardware and software through their concurrent design. Codesign has been practiced in various ways since the inception of digital systems. The specification of *instruction-set architectures* enabled the concurrent development of hardware and software as well as the creation of high-level languages and compilers. Grace Hopper was indeed a pioneer of codesign in the early 1950s with the design of portable languages (i.e., machine-independent), which led to the development of COBOL and of modern programming languages.

Within about 70 years of computer science and engineering, various embodiments of abstractions, programmability, and hardware have given different meanings to hardware/software codesign. The renewed interest on this topic in the last two decades relates to the use of structured design methodologies and tools for hardware and software design. Thus, *electronic systems design automation* had to embrace hardware/software codesign as one of its major tasks and objectives. The formalization of the related design problems enabled synthesis and verification of hardware/software systems through the development of computer-aided design methods and tools.

However, it is our opinion that a sound system design methodology must start by capturing the design specifications at the highest level of abstraction and then proceed toward an efficient implementation by subsequent refinement steps. The partition of the design into hardware and software is indeed a consequence of decisions taken at a higher level of abstraction. The critical decisions are about the architecture of the system (processors, buses, hardware accelerators, memories, and so on) that will carry on the computation and communication tasks associated with the overall specification of the design. This design process is segmented into a series of similar steps. The principles at the basis of each step consist of hiding unnecessary details of an implementation, summarizing the important parameters of the implementation in an abstract model, and limiting the design space exploration to a set of potential platform instances. The design process is a meet-in-the-middle approach where the refinement from specification toward implementation is matched against a library of components whose models are abstractions of possible implementations.

This was indeed the basis for the development of a design methodology that goes under the name of platform-based design where the steps outlined above have been formalized wherever possible. This methodology includes the development of hardware and the related software when the architectural decisions have been made and the design tasks have been mapped to the components of the architecture. A task that is mapped into hardware can then be synthesized with the appropriate tools in parallel to the software development that takes place when the mapping process has allocated the task to a programmable component. Note also that in this framework, it is rather clear that according to the available programmable components, different software design processes can be developed. In fact, programming a microprocessor is quite different than programming a DSP or a special purpose processor.

The first step in the design process is then capturing a set of specifications or requirements on the functionality and the architecture of the design. These will guide the design process through the refinement steps. Requirements are in general denotational statements about what the system is supposed to do. For example, if we are to design a special purpose math processor that computes the solution of nonlinear algebraic equations, the functional requirement would be stated as:

Find x such that $F(x) = 0$,

where no algorithm to accomplish this task has been chosen. The choice of the algorithm is already a refinement step in the design task. This example underlines that requirements are abstract statements about what the design has to accomplish. Some of the requirements may be given in terms of the properties of an implementation but still in abstracted form. For example,

The system has to consume no more than 1kW of power.

Of course this is a constraint that encompasses the entire design space from functionality to final implementation, and while in the first steps of the architecture selection, the power consumption can be estimated, and it will have to be verified at the final implementation where the physics of the solution will be known. The design space exploration is determined in part by these requirements.

In addition to the requirements, often a set of desirable features of the design can be stated. In this case, the mathematical formalism is a function that can be either minimized or maximized. Then the refinement steps take the form of an optimization problem where the objective functions are optimized in the presence of complex constraints.

Often the design process of interest has already been given in terms of high-level functionality where some design decisions have been taken. Using the example above, we may be asked to implement the Newton–Raphson procedure, a choice for the algorithm to be used to meet the requirement. This is given in terms of behavior, i.e., an operational description at the appropriate level of abstraction.

Once the behavior has been selected and described, then it is time to determine an architecture to implement this behavior that optimizes the goal function(s) and satisfies the constraints. The architecture may be developed anew or obtained combining elements in a library of available components or a combination of both whereby library elements are combined with virtual components that have to be designed from scratch.

In the design process, synthesis steps are intertwined with verification steps that check whether the constraints are satisfied, the functionality is correctly implemented, and the design is feasible.

This handbook covers extensively many topics specific to hardware/software codesign intended as system design as described above, namely, modeling, design and optimization, validation and verification, as well as application areas. Modeling has been a key design technology for capturing system-level aspects: it is achieved today via specialized languages and graphical formalisms. The underlying semantics of these representations is key to the application of rigorous methods to capture the real intent of the design and to offer a framework where properties of the design can be assessed. At the same time, the expressive power of the language is important to serve a wide variety of designers and design applications. For example, within general-purpose languages, SystemC – a class library with hardware semantics – has shown to be a viable extension to C++ to capture hardware components in an object-oriented fashion.

System architectures have changed significantly over the last two decades, to exploit the growth and diversification of the underlying semiconductor technology. As a result of the limited growth of clock operational frequencies and the wide availability of devices due to downscaling, multiprocessor architectures with significant on-chip memory (or low-latency off-chip memory) are dominating the market. Indeed, multiprocessing fits the need of realizing systems with limited energy consumption, thus avoiding thermal and dark silicon issues. Codesign in a multiprocessing environment provides major challenges, such as exploration of the design space and of parametric choices that can maximize the return of distributed software applications. Design and optimization require often cross-layer techniques that can span various modeling abstractions and operate on the tuning of various system aspects concurrently.

Much research emphasis on memory architectures has been fueled both by the need to handle big data "in proximity" as well as by the availability of novel memory technologies including their physical stacking. It is important to remark that this problem is not only a hardware design problem, as the potential beneficial use of memory hierarchies affects system and software design. By the same token, on-chip communication has evolved to *networks-on-chips (NoCs),* which encompass various structured interconnect schemes leveraging data packetization and routing. NoC design within multiprocessing systems requires the use of specific design techniques to match hardware structures realizing the network architecture to their operational protocols that are often programmable and specified in software.

Many design tools have been proposed to synthesize, partition, and optimize systems. In the recent years, the use of programmable processor cores (e.g., ARM) as black boxes within multiprocessing systems has led to a specific focus on both memory and communication synthesis and optimization. Conversely, the search for energy-performance optimal computational engines has led to *application-specific instruction-set processors*. Such processors occupy a limited but strategic part of the computing product spectrum and pose a key codesign problem. Indeed, the definition of an instruction set has been the fulcrum of codesign techniques since the invention of the digital computer. Thus, the possibility of designing and optimizing the instruction set can be viewed as searching for an optimum position of this fulcrum to balance the hardware and software cost and performance.

The selection of the functionalities to implement in hardware and of the ones in software is a system design issue that precedes HW/SW codesign. Indeed, system design can be characterized as function/architecture codesign, where function is what we wish to realize and architecture is how we are going to implement the functionality. As described above, architecture can be defined as the functional level as well. In this case, we decompose a function into a network of subfunctions. Each of this subfunction can be further decomposed until we decide to allocate the leaves of the functional decomposition to components of a hardware architecture. The hardware architecture consists of components such as processors, memories, sensors, actuators, communication entities, and specialized hardware components. Once a block of functionality is assigned to a programmable component, its implementation will be a software program running onto that component. If it is assigned to a specialized hardware, then its implementation will be a set of IP blocks, and we have a HW/SW codesign problem at hand. System design is where important decisions are taken and where it is of paramount importance to consider available components to maximize reuse. *Platform-based design* has been a major step forward in conceiving HW/SW systems that enabled the use of synthesis and verification tools with high efficiency. Indeed, a platform is a restriction of the design space.

Methods and tools for software synthesis and optimization have led to the automatic rewriting of specification in terms of the best primitives to be used by a processor. For example, ARM processors benefit from using guarded instructions, and making them explicit in software improves the compiler performance. Software analysis – in terms of execution time – is extremely important to quantify and bound delay in system design, especially in view of satisfying timing constraints for task executions. Thus, software timing analysis and verification is a key task of HW/SW codesign.

Validating system design is the most important task of all, since most digital systems are required to satisfy safety and dependability constraints. An important area is the verification of formal models that abstract parts – if not the entirety – of digital systems. Formal verification is based on choosing specific properties and checking if they are satisfied in all operational instances. Functional and timing behavior are cornerstones of verification. Often such properties are shown to hold with subsystems, and thus system composability is a key asset in proving correctness by construction. Needless to say, few systems are composable in a straightforward way, and this motivates the large research effort in verification. Large systems are often validated by semiformal techniques or by broader but weaker techniques such as simulation, emulation, and prototyping. The inherent weakness of these techniques is in asserting properties that are valid under a wide set of environmental conditions. Unfortunately, when systems fail, they often fail under unusual operating conditions.

Codesign is practiced differently in various application domains. This book covers examples such as datacenter, automotive system, video/image processing, and cyber-physical system design. The peculiarities of these domains in terms of requirements and objectives are reflected in the various ways of applying codesign modeling abstraction as well as synthesis, optimization, and verification methods.

Overall, this handbook presents a broad set of techniques that show the inherent maturity of the state of the art in hardware/software codesign.

University of California at Berkeley Alberto L. Sangiovanni-Vincentelli
USA
November 2016

Institute of Electrical Engineering Giovanni De Micheli
EPFL, Switzerland
November 2016

Preface

Hardware/software (HW/SW) codesign was first introduced as a new design methodology for SoCs (systems-on-chip) in the early 1990s to design hardware and software concurrently with the goal to reduce the design time and cost of such systems. After more than 25 years of incessant research and development, it is now regarded as a de facto standard, and the term has become serving as an umbrella for methodologies to design complex electronic systems, even distributed embedded systems. HW/SW codesign covers the full spectrum of system design issues from initial behavior specification to final implementation. Codesign methodologies also include modeling the system behavior independently of the system architecture at a high level and exploring the design space of system architecture at the early design stage. For fast design space exploration, it is necessary to estimate the system performance and resource requirements. HW/SW cosimulation enables us to develop software before hardware implementations become available. Finally, cosynthesis denoting the process of automatically synthesizing hardware components as well as software from a given specification for implementation on a target platform and including also the interfaces for communication between hardware components and processors belongs to the key problems attacked by codesign.

In spite of its significance and usefulness, we discovered that it is quite difficult to understand and learn about its benefits and full impact on real system design, particularly because there did not exist any book or reference on HW/SW codesign until the time of writing this book. Thus, it is our great pleasure to edit this handbook, quenching the thirst for the reference. In this book, we present to you the core issues of hardware/software codesign and key techniques in the design flow. In addition, selected codesign tools and design environments are described as well as case studies that demonstrate the usefulness of HW/SW codesign. This book will be updated regularly to follow the progress of design techniques and introduce commercial as well as research design tools available for our readers. It is meant to serve as a reference not only to interested researchers and engineers in the field but equally to students. We hope you all will grasp the wide spectrum of subjects that belong to HW/SW codesign and get most benefits out of it for your system design and related optimization problems.

Department of Computer Science and Engineering Soonhoi Ha
Seoul National University
Gwanak-ro 1, Gwanak-gu
Seoul, Korea
June 2017

Department of Computer Science Jürgen Teich
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)
Cauerstr. 11
Erlangen, Germany
June 2017

Contents

Volume 1

About the Editors

Soonhoi Ha
Department of Computer Science and Engineering
Seoul National University
Gwanak-ro 1, Gwanak-gu
Seoul, Korea

Soonhoi Ha received the B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Seoul, Korea, in 1985 and 1987, respectively, and the Ph.D. degree in Electrical Engineering and Computer Science from the University of California at Berkeley, Berkeley, CA, USA, in 1992. He is currently a professor with Seoul National University. His current research interests include HW/SW codesign of embedded systems, system simulation, and robust embedded software design. Prof. Ha has actively participated in the premier international conferences in the EDA area, serving CODES+ISSS 2006, ASP-DAC 2008, and ESTIMedia 2005–2006 as the program cochair and ESWeek 2017 as the vice general chair. He is an IEEE Fellow and a member of ACM.

Jürgen Teich
Department of Computer Science
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)
Cauerstr. 11
Erlangen, Germany

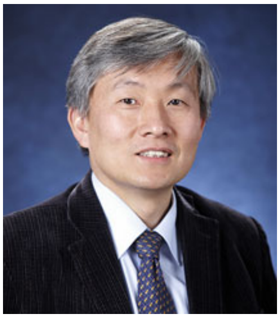
Jürgen Teich is with Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany, where he is directing the Chair for Hardware/Software Codesign since 2003. He received the M.S. degree (Dipl.- Ing.; with honors) from the University of Kaiserslautern, Germany, in 1989 and the Ph.D. degree (Dr.-Ing.; summa cum laude) from the University of Saarland, Saarbruecken, Germany, in 1993.

Prof. Teich has organized various ACM/IEEE conferences/symposia as program chair including CODES+ISSS 2007, FPL 2008, ASAP 2010, and DATE 2016. He also serves as the vice general chair of DATE 2018 as well as in the editorial board of scientific journals including *ACM TODAES*, *IEEE Design and Test*, and *JES*. He has edited two textbooks on hardware/software codesign (Springer).

Since 2010, he has also been the principal coordinator of the Transregional Research Center 89 “Invasive Computing” on multicore research funded by the German Research Foundation (DFG). Since 2011, he is a member of Academia Europaea, the Academy of Europe.

Section Editors

Part I: Introduction to Hardware/Software Codesign

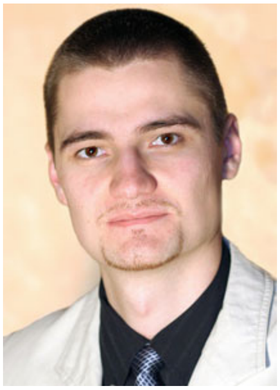


Soonhoi Ha Department of Computer Science and Engineering, Seoul National University, Gwanak-gu, Seoul, Korea
sha@snu.ac.kr

Part II: Models and Languages for Codesign

Christian Haubelt Institute of Applied Microelectronics and Computer Engineering, University of Rostock, Rostock, Germany
christian.haubelt@uni-rostock.de

Part III: Design Space Exploration



Michael Glass Institute of Embedded Systems/Real-Time Systems at Ulm University, Ulm, Germany
michael.glass@uni-ulm.de

Part IV: Processor, Memory, and Communication Architecture Design

Tulika Mitra Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore
tulika@comp.nus.edu.sg

Part V: Hardware/Software Cosimulation and Prototyping

Rainer Dömer Center for Embedded and Cyber-physical Systems, Department of Electrical Engineering and Computer Science, The Henry Samueli School of Engineering, University of California at Irvine, Irvine, CA, USA
doemer@uci.edu

Part VI: Performance Estimation, Analysis, and Verification

Petru Eles Department of Computer and Information Science, Linkoping University, Linköping, Sweden
petru.eles@liu.se

Part VII: Hardware/Software Compilation and Synthesis

Aviral Shrivastava School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, USA
aviral.shrivastava@asu.edu

Part VIII: Codesign Tools and Environment

Andreas Gerstlauer Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA
gerstl@ece.utexas.edu

Part IX: Applications and Case Studies

Shuvra Bhattacharyya Department of Electrical and Computer Engineering and Institute for Advanced Computer Studies, University of Maryland, College Park, USA
Department of Pervasive Computing, Tampere University of Technology, Tampere, Finland
ssb@umd.edu

Contributors

Miguel Angel Aguilar Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany

Leonie Ahrendts Institute of Computer and Network Engineering, Technical University Braunschweig, Braunschweig, Germany

Benny Akesson Eindhoven University of Technology, Eindhoven, The Netherlands

David Atienza Embedded Systems Laboratory (ESL), EPFL, Lausanne, Switzerland

Seyed-Hosein Attarzadeh-Niaki Shahid Beheshti University (SBU), Tehran, Iran

Philip Axer NXP Semiconductors, Hamburg, Germany

Hadi Ahmadi Balef Eindhoven University of Technology, Eindhoven, The Netherlands

Giovanni Beltrame Polytechnique Montréal, Montreal, QC, Canada

Shuvra S. Bhattacharyya Department of Electrical and Computer Engineering and Institute for Advanced Computer Studies, University of Maryland, College Park, MD, USA

Department of Pervasive Computing, Tampere University of Technology, Tampere, Finland

Haseeb Bokhari University of New South Wales (UNSW), Sydney, NSW, Australia

Jens Brandt Faculty of Electrical Engineering and Computer Science, Hochschule Niederrhein, Krefeld, Germany

Gabriela Breaban Eindhoven University of Technology, Eindhoven, The Netherlands

Oliver Bringmann Wilhelm-Schickard-Institut, University of Tübingen, Tübingen, Germany

Embedded Systems, University of Tübingen, Tübingen, Germany

Davide Brunelli Department of Industrial Engineering, University of Trento, Trento, Italy

Jian Cai Arizona State University, Tempe, AZ, USA

Jeronimo Castrillon Center for Advancing Electronics Dresden, TU Dresden, Dresden, Germany

Samarjit Chakraborty TU Munich, Munich, Germany

Wanli Chang Singapore Institute of Technology, Singapore, Singapore

Jae-Min Cho Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea

Kiyoung Choi Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea

Pablo G. Del Valle Embedded Systems Laboratory (ESL), EPFL, Lausanne, Switzerland

Clément Deschamps Antfield SAS, Grenoble, France

Jonas Diemer Symtavision, Braunschweig, Germany

Rainer Dömer Center for Embedded and Cyber-Physical Systems, Department of Electrical Engineering and Computer Science, The Henry Samueli School of Engineering, University of California, Irvine, CA, USA

Nikil Dutt Center for Embedded and Cyber-Physical Systems, University of California Irvine, Irvine, CA, USA

Wolfgang Ecker Infineon Technologies AG, Neubiberg, Germany

Petru Eles Department of Computer and Information Science, Linköping University, Linköping, Sweden

Ahmed Eltawil Center for Embedded and Cyber-Physical Systems, University of California Irvine, Irvine, CA, USA

Rolf Ernst Institute of Computer and Network Engineering, Technical University Braunschweig, Braunschweig, Germany

Juan Fernando Eusse Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany

Heiko Falk Institute of Embedded Systems, Hamburg University of Technology, Hamburg, Germany

Joachim Falk Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany

Franco Fummi Università di Verona, Verona, Italy

Marc Geilen Eindhoven University of Technology, Eindhoven, The Netherlands

Andreas Gerstlauer Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA

Christoph Gerum Embedded Systems, University of Tübingen, Tübingen, Germany

Michael Glaß Institute of Embedded Systems/Real-Time Systems at Ulm University, Ulm, Germany

Kees Goossens Eindhoven University of Technology, Eindhoven, The Netherlands

Sven Goossens Eindhoven University of Technology, Eindhoven, The Netherlands

Dip Goswami Eindhoven University of Technology, Eindhoven, The Netherlands

Soonhoi Ha Department of Computer Science and Engineering, Seoul National University, Gwanak-gu, Seoul, Korea

Christian Haubelt Department of Computer Science and Electrical Engineering, Institute of Applied Microelectronics and Computer Engineering, University of Rostock, Rostock, Germany

Jörg Henkel Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Fernando Herrera GESE Group, TEISA Department, ETSIIT, Universidad de Cantabria, Santander, Cantabria, Spain

Robin Hofmann Institute of Computer and Network Engineering, Technical University Braunschweig, Braunschweig, Germany

Pekka Jääskeläinen Tampere University of Technology, Tampere, Finland

Axel Jantsch Vienna University of Technology, Vienna, Austria

Prachi Joshi Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA

Hanwoong Jung Seoul National University, Gwanak-gu, Seoul, Korea

Amin Khajeh Broadcom Corp., San Jose, CA, USA

Martijn Koedam Eindhoven University of Technology, Eindhoven, The Netherlands

Tim Kogel Synopsys, Inc., Aachen, Germany

Fadi Kurdahi Center for Embedded and Cyber-Physical Systems, University of California Irvine, Irvine, CA, USA

Luciano Lavagno Politecnico di Torino, Torino, Italy

Mihai Teodor Lazarescu Politecnico di Torino, Torino, Italy

Kyunghun Lee Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

Rainer Leupers Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany

Lin Li Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

Yonghui Li Eindhoven University of Technology, Eindhoven, The Netherlands

Shuoxin Lin Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

Guantao Liu Center for Embedded and Cyber-Physical Systems, University of California, Irvine, CA, USA

Yanzhou Liu Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

Martin Lukasiewycz Robert Bosch GmbH, Corporate Research, Renningen, Germany

Grant Martin Cadence Design Systems, San Jose, CA, USA

Peter Marwedel Computer Science, TU Dortmund University, Dortmund, Germany

Julio Medina Software Engineering and Real-Time Group, University of Cantabria, Santander, Cantabria, Spain

Luc Michel Antfield SAS, Grenoble, France

Tulika Mitra Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore

Daniel Mueller-Gritschneder Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany

Majid Nabi Eindhoven University of Technology, Eindhoven, The Netherlands

Andrew Nelson Eindhoven University of Technology, Eindhoven, The Netherlands

Olaf Neugebauer Computer Science, TU Dortmund University, Dortmund, Germany

Hristo Nikolov Leiden University, Leiden, The Netherlands

Sebastian Ottlik Microelectronic System Design, FZI Research Center for Information Technology, Karlsruhe, Germany

Santiago Pagani Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Ali Pahlevan Embedded Systems Laboratory (ESL), EPFL, Lausanne, Switzerland

Preeti Ranjan Panda Department of Computer Science and Engineering, Indian Institute of Technology Delhi, New Delhi, India

Jacopo Panerati Polytechnique Montréal, Montreal, QC, Canada

Sri Parameswaran University of New South Wales (UNSW), Sydney, NSW, Australia

Young-Hwan Park Digital Media and Communications R&D Center, Samsung Electronics, Seoul, Korea

Frédéric Pétrot Université de Grenoble Alpes, Grenoble, France

Andy Pimentel University of Amsterdam, Amsterdam, The Netherlands

Teemu Pitkänen Ajat Oy, Espoo, Finland

William Plishker Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA

Graziano Pravadelli Università di Verona, Verona, Italy

Davide Quaglia Università di Verona, Verona, Italy

Felix Reimann Audi Electronics Venture GmbH, Gaimersheim, Germany

Maurizio Rossi Department of Industrial Engineering, University of Trento, Trento, Italy

Debayan Roy TU Munich, Munich, Germany

Ingo Sander KTH Royal Institute of Technology, Stockholm, Sweden

Santanu Sarma University of California Irvine, Irvine, CA, USA

Gunar Schirner Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA

Frank Schirrmeister Cadence Design Systems, San Jose, CA, USA

Tim Schmidt Center for Embedded and Cyber-Physical Systems, University of California, Irvine, CA, USA

Klaus Schneider Embedded Systems Group, University of Kaiserslautern, Kaiserslautern, Germany

Johannes Schreiner Infineon Technologies AG, Neubiberg, Germany

Donatella Sciuto Politecnico di Milano, Milano, Italy

Muhammad Shafique Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Mansureh Shahraki Moghaddam Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea

Weihua Sheng Silexica GmbH, Köln, Germany

Jun Yong Shin Center for Embedded and Cyber-Physical Systems, University of California Irvine, Irvine, CA, USA

Aviral Shrivastava School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA

Shubhendu Sinha Eindhoven University of Technology, Eindhoven, The Netherlands

Todor Stefanov Leiden University, Leiden, The Netherlands

Sander Stuijk Eindhoven University of Technology, Eindhoven, The Netherlands

Jarmo Takala Tampere University of Technology, Tampere, Finland

Rasool Tavakoli Eindhoven University of Technology, Eindhoven, The Netherlands

Jürgen Teich Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany

Daniel Thiele Elektrobit Automotive GmbH, Erlangen, Germany

Juan Valencia Eindhoven University of Technology, Eindhoven, The Netherlands

Reinier van Kampenhout Eindhoven University of Technology, Eindhoven, The Netherlands

Peter van Stralen Philips Healthcare, Best, The Netherlands

Alexander Viehl Microelectronic System Design, FZI Research Center for Information Technology, Karlsruhe, Germany

Eugenio Villar GESE Group, TEISA Department, ETSIIT, Universidad de Cantabria, Santander, Cantabria, Spain

Sara Vinco Politecnico di Torino, Turin, Italy

Yosinori Watanabe Cadence Design Systems, San Jose, CA, USA

Marilyn Wolf School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Chun Jason Xue City University of Hong Kong, Hong Kong, Hong Kong

Christian Zebelein Valeo Siemens eAutomotive Germany GmbH, Erlangen, Germany

Haibo Zeng Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA

Licong Zhang TU Munich, Munich, Germany

List of Acronyms

6LoWPAN	IPv6 over Low Power Wireless Personal Area Network
ABV	Assertion-Based Verification
AC	Alternating Current
ACK	Acknowledgement
ADAS	Advanced Driver Assistance System
ADC	Analog-to-Digital Converter
ADF	Architecture Description File
ADG	Approximated Dependence Graph
ADL	Architecture Description Language
ADM	Abstract Design Module
ADRS	Average Distance from Reference Set
ADT	Abstract Data Type
AFDX	Avionics Full-Duplex Switched Ethernet
AHB	Advanced High-performance Bus
AIF	Averest Intermediate Format
ALAP	As Late As Possible
ALM	Adaptive Logic Module
ALU	Arithmetic-Logic Unit
ANN	Artificial Neural Network
APB	Advanced Peripheral Bus
API	Application Programming Interface
ARM	Advanced Risc Machines
ARQ	Automatic Repeat Request
ASAP	As Soon as Possible
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instruction-set Processor
ASK	Amplitude Shift Key
ASMBL	Advanced Silicon Modular Block
ASP	Application-Specific Processor
AST	Abstract Syntax Tree
AT	Approximately Timed

AT-BP	Approximately Timed Base Protocol
AV	Architects View
AVB	Audio/Video Bridging
AXI	Advanced eXtensible Interface
BB	Basic Block
BCET	Best-Case Execution Time
BCRT	Best-Case Response Time
BD	Budget Descriptor
BDF	Boolean Data Flow
BER	Bit Error Rate
BERET	Bundled Execution of REcurring Traces
BFD	Best-Fit-Decreasing
BFM	Bus-Functional Model
BIST	Built-In Self-Test
BLB	Bit Lock Block
BLM	Block-Level Model
BLS	Binary-Level Simulation
BNF	Backus-Naur Form
BOM	Bill of Materials
BPSK	Binary PSK
BRF	Bypass Register File
BSP	Board Support Package
BTB	Branch Target Buffer
CA	Cycle Accurate
CAD	Computer-Aided Design
CAL	Cal Actor Language
CAN	Controller Area Network
CCA	Configurable Compute Accelerator
CC	Communication Controller
CCE	Configuration Cache Element
CCSP	Credit-Controlled Static Priority
CDC	Clock Domain Crossing
CDFG	Control-/Data-Flow Graph
CDMA	Code Division Multiple Access
CE	Communication Element
CFDF	Core Functional Data Flow
CFG	Control-Flow Graph
CFU	Custom Functional Unit
CGA	Coarse-Grained Array
CG	Call Graph
CGRA	Coarse Grained Reconfigurable Architecture
CIC	Common Intermediate Code
CIL	Compiler-In-the-Loop
CIM	Computation Independent Model
CIS	Custom Instruction-Set

CLB	Configurable Logic Block
CLDSE	Cross-Layer Design Space Exploration
CM	Communication Memory
CMOS	Complementary Metal-Oxide-Semiconductor
CMP	Chip Multi-Processor
CNN	Convolutional Neural Network
CORDIC	COordinate Rotational DIgital Computer
CVMP	Correlation-aware VM Placement
COTS	Commercial/Components Off-The-Shelf
CPA	Compositional Performance Analysis
CPF	Common Power Format
CPN	C for Process Networks
CPS	Cyber-Physical System
CPU	Central Processing Unit
CRAC	Computer Room Air Conditioning
CRC	Cyclic Redundancy Check
CRPD	Cache-Related Preemption Delay
CSDF	Cyclo-Static Data Flow
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CTI	Charge Transfer Interconnect
CTL	Computation Tree Logic
CUDA	Compute Unified Device Architecture
CV	Computer Vision
D2H	Device-to-Host
DAG	Directed Acyclic Graph
DB	Database
DBT	Dynamic Binary Translation
DC	Direct Current
DCG	Dynamic Call Graph
DCT	Discrete Cosine Transform
DDF	Dennis Data Flow
DDR	Double Data Rate
DE	Discrete Event
DES	Discrete Event Simulation
DFG	Data-Flow Graph/Dependence-Flow Graph
DFT	Discrete Fourier Transfrom
DICE	DSPCAD Integrative Command Line Environment
DIF	Decimation-in-Frequency/Data-flow Interchange Format
DISC	Dynamic Instruction-Set Computer
DIT	Decimation-in-Time
DLMB	Data Local Memory Bus
DLP	Data-Level Parallelism
DMA	Direct Memory Access
DMAMEM	DMA Memory
DMEM	Data Memory

DMI	Direct Memory Interface
DoD	Depth-of-Discharge
DoE	Design of Experiments
DOR	Dimension Ordered Routing
DP	Dynamic Programming
DPLL	Davis-Putnam-Logemann-Loveland
DPM	Dynamic Power Management
DPR	Dynamic Partial Reconfiguration
DRAA	Dynamically Reconfigurable ALU Array
DRAM	Dynamic Random-Access Memory
DRESC	Dynamically Reconfigurable Embedded System Compiler
DSE	Design Space Exploration
DSL	Domain-Specific Language
DSML	Domain-Specific Modeling Language
DSO	Distribution System Operator
DSP	Digital Signal Processor/Digital Signal Processing
DTA	Dynamic Timing Analysis
DTM	Dynamic Thermal Management
DUT	Design Under Test
DUV	Design Under Verification
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
DWM	Domain Wall Memory
DWT	Discrete Wavelet Transform
EA	Evolutionary Algorithm
EBNF	Extended Backus-Naur Form
ECO	Engineering Change Order
ECU	Electronic Control Unit
EDA	Electronic Design Automation
EDF	Earliest Deadline First
EDP	Energy-Delay Product
EDSP	Energy-Delay Square Product
E/E	Electric and Electronic
EEPROM	Electrically Erasable Programmable Read-Only Memory
EFSM	Extended Finite-State Machine
EGRA	Expression Grained Reconfigurable Array
ELF	Executable and Linkable Format
EMB	Electro-Mechanical Brake
EMF	Eclipse Modeling-Framework
EML	Execution Modeling Level
EMS	Edge Centric Modulo Scheduling
EOH	Extremal Optimization meta-Heuristic
ES	Embedded System
ESL	Electronic System Level
ESS	Energy Storage Systems

ET	Event-Triggered/Execution Time
ETSCH	Extended TSCH
EWFD	Equally-Worst-Fit-Decreasing
FBSP	Frame-Based Static Priority
FCFS	First-Come First-Serve
FDS	Force-Directed Scheduling
FeRAM	Ferro-electric Random-Access Memory
FFT	Fast Fourier Transform
FIFO	First-In First-Out
FIR	Finite Impulse Response
ForSyDe	Formal System Design
FPGA	Field-Programmable Gate Array
FS	Feature Selection
FSM	Finite State Machine
FTDMA	Flexible Time Division Multiple Access
FT	Fast Timed
FunState	Functions Driven by State Machines
GA	Genetic Algorithm
GALS	Globally Asynchronous Locally Synchronous
GCC	GNU Compiler Collection
GFRBM	Generic File Reader Bus Master
GIPS	Giga-Instruction Per Second
GLV	Graph-Level Vectorization
GME	Generic Modeling Environment
GOPS	Giga Operations Per Second
GPGPU	General-Purpose computing on Graphics Processing Units
GPIO	General-Purpose Input/Output-pin
GPP	General-Purpose Processor
GPRS	General Packet Radio Service
GPT	General-Purpose Timer
GPU	Graphics Processing Unit
GUI	Graphical User Interface
H2D	Host-to-Device
HAL	Hardware Abstraction Layer
HAPS	High-performance ASIC Prototyping System
HDB	Hardware Database
HDL	Hardware Description Language
HDS	Hardware-Dependent Software
HES	Hybrid Electric Systems
HLS	High-Level Synthesis
HMP	Heterogeneous Multi-core Processor
HPC	Horizontally Partitioned Cache
HRM	Hardware Resource Modeling
HSCD	Hardware/Software Codesign
HSDF	Homogeneous (Synchronous) Data Flow

HTML	Hypertext Markup Language
HVL	Hardware Verification Language
HW	Hardware
I2C	Inter-Integrated Circuit
ICFG	Interprocedural Control-Flow Graph
ICT	Information and Communications Technology
ICU	Input Capture Unit
IDC	Inquisitive Defect Cache
IDE	Integrated Development Environment
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
II	Initiation Interval
ILMB	Instruction Local Memory Bus
ILP	Integer Linear Program/Instruction-Level Parallelism
IMEM	Instruction Memory
IMS	Iterative Modulo Scheduling
IOE	I/O Element
I/O	Input/Output
IoT	Internet of Things
IPC	Inter-Process Communication/Instructions Per Cycle
IP	Intellectual Property
IPB	Intellectual Property Block
IPM	Intellectual Property Module
IPS	Instruction Per Second
IR	Intermediate Representation
ISA	Instruction-Set Architecture
ISEF	Stretch Instruction-Set Extension Fabric
ISR	Interrupt Service Routine
ISS	Instruction-Set Simulator
IT	Information Technology
ITRS	International Technology Roadmap for Semiconductors
ITS	Individual Test Subdirectory
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
KPN	Kahn Process Network
LAB	Logic Array Block
LCS	Live Cache States
LE	Logic Element
LIDE	LIghtweight Data-flow Environment
LIN	Local Interconnect Network
LISA	Language for Instruction-Set Architectures
LLVM	Low-Level Virtual Machine
LRU	Least-Recently Used
LS	List Scheduling

LTF	Largest Task First
LTL	Linear Time Logic
LT	Loosely Timed
LUT	Look-Up Table
M2M	Model-to-Model
MAC	Media Access Control/Multiply-Accumulator
MAPE	Mean Average Percentage Error
MAPS	MPSoC Application Programming Studio
MARTE	Modeling and Analysis of Real-Time Embedded Systems
MBD	Model-Based Design
MCO	Multi-Core Optimization
MCR	Mode Change Request
MCS	Mixed-Criticality System
MDA	Model-Driven Architecture
MDD	Model-Driven Design
MDP	Markov Decision Process
MDSDF	Multi-Dimensional Synchronous Data Flow
MILP	Mixed Integer Linear Programming
MIMO	Multiple Input Multiple Output
MIPS	Million Instructions Per Second
MIR	Medical Image Registration
MISO	Multiple Input Single Output
MJPEG	Motion JPEG
MLBJ	Multi-Level Back Jumping
MLoC	Million Lines of Code
MMC/SD	Multimedia/Secure Digital Card
MMIO	Memory-Mapped I/O
MMU	Memory Management Unit
MoC	Model of Computation
MOF	Meta Object Facility
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MOST	Media Oriented Systems Transport
MPSoC	Multi-Processor System-on-Chip
MRRG	Modulo Resource Routing Graph
MTF	Mean Time to Failure
MTJ	Magnetic Tunnel Junction
MTM	Mode Transition Machine
NDF	Non-Determinate Data Flow
NFP	Non-Functional Property
NI	Network Interface
nML	not a Machine Language
NMOS	Negative-type Metal-Oxide-Semiconductor
NN	Neural Network
NoC	Network-on-Chip
NOCT	Nominal Operating Cell Temperature

NRE	Non-Recurring Engineering
NVIC	Nested Vectored Interrupt Controller
NVM	Non-Volatile Memory
OCL	Object Constraint Language
OFDM	Orthogonal Frequency Dependent Multiplexing
OMG	Object Management Group
OOO PDES	Out-of-Order Parallel Discrete Event Simulation
OSAL	Operation Set Abstraction Layer
OSCI	Open SystemC Initiative
OS	Operating System
OT	Operation Table
OVL	Open Verification Library
OVM	Open Verification Methodology
PAMONO	Plasmon-Assisted Microscopy of Nano-Objects
PB	Pseudo Boolean
PCI	Peripheral Component Interconnect
PCM	Phase Change Memory
PC	Personal Computer
PCP	Peak Clustering-based Placement
PDES	Parallel Discrete Event Simulation
PDF	Probability Density Function
PDU	Power Distribution Unit
PE	Processing Element
PFU	Programmable Functional Unit
PI	Principal Investigator
PIC	Programmable Interrupt Controller
PIM	Platform Independent Model
PIP	Parametric Integer Programming
PLB	Processor Local Bus
PLL	Phase Locked Loop
PLP	Pipeline-Level Parallelism
PMOS	Positive-type Metal-Oxide-Semiconductor
PMU	Power Management Unit
PNG	Portable Network Graphics
PN	Process Network
PPN	Polyhedral Process Network
PREESM	Parallel and Real-time Embedded Executives Scheduling Method
PRISC	Programmable Instruction-Set Processor
PSDF	Parameterized Synchronous Data Flow
PSK	Phase Shift Keying
PSL	Property Specification Language
PSM	Program State Machine/Parameterized Sets of Modes/Platform Specific Model
PSNR	Peak SNR
PSO	Particle Swarm Optimization

PSTC	Path Segment Timing Characterization
PV	Photovoltaic
PVT	Programmers View Time
PWM	Pulse-Width Modulation
QAM	Quadrature Amplitude Modulation
QEA	Quantum-inspired Evolutionary Algorithm
QoS	Quality of Service
QPSK	Quadrature PSK
RAM	Random-Access Memory
RAW	Read-After-Write
RCM	Reconfigurable Computing Module
RC	Resistor-Capacitor/Reconfigurable Cell
RCS	Reaching Cache States
RDF	Random Dopant Fluctuations
RFID	Radio-Frequency Identification
RF	Register File/Radio Frequency
RFTS	Run Fast Then Stop
RISC	Reduced Instruction-Set Processor/Recoding Infrastructure for SystemC
RISPP	Rotating Instruction-Set Processing Platform
RLD	Run Length Decoding
ROM	Read-Only Memory
RR	Round Robin
RRAM	Resistive Random-Access Memory
RSM	Response Surface Modeling
RST	ReServation Table
RT	Response Time
RTC	Real-Time Clock
RTL	Register Transfer Level
RTOS	Real-Time Operating System
RVC	Reconfigurable Video Coding
SADF	Scenario-Aware Data Flow
SANLP	Static Affine Nested Loop Program
SA	Simulated Annealing
SAT	Boolean Satisfiability
SBS	Sequential Backward Selection
SCC	Single Chip Cloud computer/Strongly Connected Component
SCE	System-on-Chip Environment
SCML	SystemC Modeling Library
SDC	Secure Digital Card
SDF	Synchronous Data Flow
SDK	Software Development Kit
SDR	Software Defined Radio
SDS	System Development Suite
SDTC	Scheduling and Data Transfer Configuration

SERE	Sequential Extended Regular Expression
SESE	Single-Entry Single-Exit
SFA	Single Frequency Approximation
SFS	Sequential Forward Selection
SFU	Specialized Functional Unit
SG	Segment Graph
SI	Scheduling Interval
SIMD	Single Instruction, Multiple Data
SIMT	Single Instruction, Multiple Threads
SLDL	System-Level Description Language
SLD	System-Level Design
SLP	System-Level Power
SLS	Source-Level Simulation/System-Level Synthesis
SMP	Symmetric Multi-Processing
SMT	Satisfiability Modulo Theories
SMV	Symbolic Model Verifier
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip/State of Charge
SoH	State of Health
SPI	Serial Peripheral Interface/Signal Passing Interface
SPKM	Split & Push Kernel Mapping
SPMD	Single Program, Multiple Data
SPM	Scratchpad Memory
SPNP	Static-Priority Non-Preemptive
SPP	Static Priority Preemptive
SPU	Synergistic Processor Unit
SRAM	Static Random-Access Memory
SSA	Static Single Assignment
SSTA	Statistical Static Timing Analysis
STC	Standard Test Conditions
STMD	Single Thread, Multiple Data
STree	Schedule Tree
STT-RAM	Spin-Transfer Torque Random-Access Memory
SVA	System Verilog Assertions
SVM	Support Vector Machine
SWC	Software Cache
SW	Software
SysteMoC	SystemC Models of Computation
T-BCA	Transaction-based Bus Cycle Accurate
TB	Translation Block
TCE	TTA-based Codesign Environment
TCL	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TDB	Timing Database
TDM	Time-Division Multiplexing

TDMA	Time-Division Multiple Access
TDP	Thermal Design Power
TD	Temporal Decoupling
TFT	Thin-Film Transistor
TIE	Tensilica Instruction Extension
TIFU	Timer, Interrupt, and Frequency Unit
TLM	Transaction-Level Model
TLP	Task-Level Parallelism/Thread-Level Parallelism
TRM	Trace Replay Module
TSCH	Time-Synchronised Channel Hopping
TSN	Time-Sensitive Networking
TSP	Thermal Safe Power
TTA	Transport-Triggered Architecture
TT-CAN	Time-Triggered CAN
TTEthernet	Time-Triggered Ethernet
TTP	Time-Triggered Protocol
TT	Time-Triggered
TWCA	Typical Worst-Case Analysis
TWCRT	Typical Worst-Case Response Time
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver/Transmitter
UML	Unified Modeling Language
UPF	Unified Power Format
UPS	Uninterruptible Power Supply
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UTP	Universal Testing Profile
UVM	Universal Verification Methodology
VEP	Virtual Execution Platform
VFI	Voltage/Frequency Island
VF	Vectorization Factor
V/f	Voltage/Frequency
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VIVU	Virtual Inlining and Virtual Unrolling
VLIW	Very Long Instruction Word
VLSI	Very-Large-Scale Integration
VM	Virtual Machine
VOS	Voltage Over Scaling
VPU	Virtual Processing Unit
VP	Virtual Prototype
VSIA	Virtual Socket Interface Alliance
VSL	Value Specification Language
VSP	Virtual System Platform
WAR	Write-After-Read

WAW	Write-After-Write
WCC	WCET-aware C Compiler
WCDMA	Wideband CDMA
WCEC	Worst-Case Energy Consumption
WCEP	Worst-Case Execution Path
WCET	Worst-Case Execution Time
WCRT	Worst-Case Response Time
WL	Word Line
WSDF	Windowed Synchronous Data Flow
WSDL	Web Service Definition Language
WSN	Wireless Sensor Network
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema
XSLT	Extensible Stylesheet Language Transformations
YML	Y-chart Modeling Language