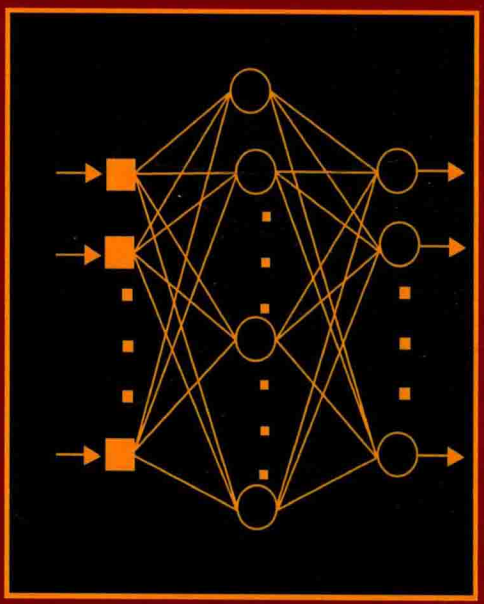
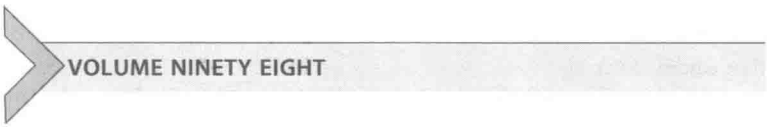
Advances in COMPUTERS

Volume 98



Edited by
ALI HURSON

Series Editors
Ali Hurson and Atif Memon



ADVANCES IN COMPUTERS

Edited by

ALI R. HURSON
Missouri University of Science and Technology Rolla, MO, USA

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Academic Press is an imprint of Elsevier

Academic Press is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA
525 B Street, Suite 1800, San Diego, CA 92101-4495, USA
125 London Wall, London, EC2Y 5AS, UK
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK

First edition 2015

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-12-802132-3
ISSN: 0065-2458

PREFACE

Traditionally, *Advances in Computers*, the oldest series to chronicle the rapid evolution of computing, annually publishes four volumes, each typically comprised of four to eight chapters, describing new developments in the theory and applications of computing. The theme of this 98th volume is inspired by the advances in information technology. Within the spectrum of information technology, this volume touches a variety of topics ranging from computer architecture and energy-efficient design techniques, cyber-physical critical infrastructure systems, model-based testing, and multi-objective optimization methods. The volume is a collection of four chapters that were solicited from authorities in the field, each of whom brings to bear a unique perspective on the topic.

In Chapter 1, "An Overview of Architecture-Level Power- and Energy-Efficient Design Techniques," Ratković *et al.* articulate the holistic consideration of power and energy consumption of computer systems at all design levels without sacrificing the processing power. Several circuit and architectural metrics are defined, the notion of dynamic and static energy is discussed, distinction between power and energy is articulated, and metrics such as "Energy-Delay Product" and "Energy per Instruction" are introduced. The Chapter then classifies, surveys, and analyzes, in detail, recent power- and energy-efficient optimization techniques such as dynamic voltage and frequency scaling (DVFS), power gating, and clock gating, as advanced in the literature.

In Chapter 2, "A Survey of Research on Data Corruption in Cyber-Physical Critical Infrastructure Systems," Woodard *et al.* emphasize the effect of data corruption, either intentional (i.e., cyber, physical, or cyber-physical attacks) or unintentional (i.e., failures in sensors, processors, storage, or communication hardware), within the scope of critical cyber-physical systems (i.e., power grids, intelligent water distribution networks, smart transportation systems). It presents a comprehensive analysis of various data corruption and mitigation techniques. Finally, a number of studies on the negative effects of system execution on corrupted data are presented.

In Chapter 3, "A Research Overview of Tool-Supported Model-Based Testing of Requirements-Based Designs," Marinescu *et al.* address software testing objectives as a means to gain confidence in software products through fault detection, by observing the differences between the behavior of the

implementation and the expected behavior described in the specification. The Chapter presents an overview of and classifies the state of the art in tool-supported model-based testing with an eye toward gaining insight into the gaps in the current tools used by industry and academia. Four representative tools (i.e., ProTest, UPPAAL Cover, MaTeLo, and CompleteTest) are chosen to show the differences in modeling notations, test-case generation methods, and the produced test cases.

Finally, in Chapter 4, "Preference Incorporation in Evolutionary Multiobjective Optimization: A Survey of the State of the Art," Bechikh *et al.* emphasize on the application of evolutionary algorithms as a means for multiobjective optimization. A classification of preference-based Multi-objective Optimization Evolutionary Algorithms based on the structure of the decision maker's preference information is presented and several approaches are discussed and analyzed. Finally, the future trends in this research area and some possible paths for future research are outlined.

I hope that you find these articles of interest, and useful in your teaching, research, and other professional activities. I welcome feedback on the volume and suggestions for future volumes.

Ali R. Hurson
Missouri University of Science and Technology
Rolla, MO, USA

CONTENTS

CHAPTER ONE

An Overview of Architecture-Level Power- and Energy-Efficient Design Techniques

Ivan Ratković*,†, Nikola Bežanić‡, Osman S. Ünsal*, Adrian Cristal*,†,§, Veljko Milutinović‡

*Barcelona Supercomputing Center, Barcelona, Spain
†Polytechnic University of Catalonia, Barcelona, Spain
‡School of Electrical Engineering, University of Belgrade, Belgrade, Serbia
§CSIC-IIIA, Barcelona, Spain

Contents

Abstract

Power dissipation and energy consumption became the primary design constraint for almost all computer systems in the last 15 years. Both computer architects and circuit designers intent to reduce power and energy (without a performance

Advances in Computers, Volume 98
ISSN 0065-2458
http://dx.doi.org/10.1016/bs.adcom.2015.04.001

degradation) at all design levels, as it is currently the main obstacle to continue with further scaling according to Moore's law. The aim of this survey is to provide a comprehensive overview of power- and energy-efficient "state-of-the-art" techniques. We classify techniques by component where they apply to, which is the most natural way from a designer point of view. We further divide the techniques by the component of power/energy they optimize (static or dynamic), covering in that way complete low-power design flow at the architectural level. At the end, we conclude that only a holistic approach that assumes optimizations at all design levels can lead to significant savings.

ABBREVIATIONS

A Switching Activity Factor
ABB Adaptive Body Biasing
BHB Block History Buffer
C Capacitance
CMP Chip-Multiprocessor
CPI Cycles per Instruction
CU Control Unit
d Delay
DCG Deterministic Clock Gating
DVFS Dynamic Voltage and Frequency Scaling
DVS Dynamic Voltage Scaling
E Energy
EDP Energy-Delay Product
E^iD^jP Energy^i-Delay^j Product
EPI Energy-per-Instruction
FP Floating Point
FU Functional Unit
GALS Globally Asynchronous Locally Synchronous
IQ Instruction Queue
IPC Instructions Per Cycle
LSQ Load/Store Queue
LUT Look-up Table
MCD Multiple-Clock-Domain
MFLOPS Millions of Floating point Operations Per Second
MILP Mixed-Integer Linear Programming
MIPS Millions of Instructions Per Second
NEMS Nanoelectromechanical Systems
P Power
PCPG Per-Core Power Gating
RBB Reverse Body Biasing
RDO Runtime DVFS Optimizer
RF Register File
ROB Reorder Buffer
SIMD Single Instruction, Multiple Data
UC Micro-Operation Cache

1. INTRODUCTION

After the technology switch from bipolar to CMOS, in the 1980s and early 1990s, digital processor designers had high performance as the primary design goal. At that time, power and area remained to be secondary goals. Power started to become a growing design concern when, in the mid- to late-1990s, it became obvious that further technology feature size scaling according to Moore's law [1] would lead to a higher power density, which could became extremely difficult or almost impossible to cool.

While, during the 1990s, the main way to reduce microprocessor power dissipation was to reduce dynamic power, by the end of the twentieth century the leakage (static) power became a significant problem. In the mid-2000s, rapidly growing static power in microprocessors approaches to its dynamic power dissipation [2]. The leakage current of a MOSFET increases exponentially with a reduction in the threshold voltage. Static power dissipation, a problem that had gone away with the introduction of CMOS, became a forefront issue again.

Different computer systems have different design goals. In high-performance systems, we care more about power dissipation than energy consumption; however, in mobile systems, the situation is reverse. In battery-operated devices, the time between charges is the most important factor; thus, lowering the microprocessor energy as much as possible, without spoiling performance, is the main design goal. Unfortunately, the evolution of the battery capacity is much slower than the electronics one.

Power density limits have already been spoiling planned speed-ups by Moore's law, and this computation acceleration degradation trend is still growing. As technology feature size scaling goes further and further, power density is getting higher and higher. Therefore, it is likely that, very soon, majority of the chip's area is going to be powered off; thus, we will have "dark silicon." Dark silicon (the term was coined by ARM) is defined as the fraction of die area that goes unused due to power, parallelism, or other constraints.

Due to the above described facts, power and energy consumption are currently one of the most important issues faced by computer architecture community and have to be reduced at all possible levels. Thus, there is a need to collect all efficient power/energy optimization techniques in a unified, coherent manner.

This comprehensive survey of architectural-level energy- and power-efficient optimization techniques for microprocessor's cores aims to help low-power designer (especially computer architects) to find appropriate techniques in order to optimize their design. In contrast with the other low-power survey papers [3–5], the classification here is done in a way that processor designers could utilize in a straightforward manner—by component (Section 3). The presentation of the techniques (Section 4) was done by putting the emphasis on newer techniques rather than older ones. The metrics of interest for this survey are presented in Section 2 which help reading for audience with less circuit-level background. Future trends are important in the long-term projects as CMOS scaling will reach its end in a few years. Current state of microprocessor scaling and a short insight of novel technologies are presented in Section 5. At the end, in Section 6 we conclude this chapter and a short review of the current low-power problems.

2. METRICS OF INTEREST

Here we present the metrics of interest as a foundation for later sections. We present both circuit- and architectural-level metrics.

2.1 Circuit-Level Metrics

We can define two types of metrics which are used in digital design—basic and derived metrics. The first one is well-known, while the latter is used in order to provide a better insight into the design trade-offs.

2.1.1 Basic Metrics

Delay (d) Propagation delay, or gate delay, is the essential performance metric, and it is defined as the length of time starting from when the input to a logic gate becomes stable and valid, to the time that the output of that logic gate is stable and valid. There are several exact definitions of delay but it usually refers to the time required for the output to reach from 10% to 90% of its final output level when the input changes. For modules with multiple inputs and outputs, we typically define the propagation delay as the worst-case delay over all possible scenarios.

Capacitance (C) is the ability of a body to hold an electrical charge, and its unit according to IS is the *Farad* (*F*). Capacitance can also be defined as a measure of the amount of electrical energy stored (or separated) for a given electric potential. For our purpose more appropriate is the last definition.

Switching Activity Factor (A) of a circuit node is the probability the given node will change its state from 1 to 0 or vice versa at a given clock tick. Activity factor is a function of the circuit topology and the activity of the input signals. Knowledge of activity factor is necessary in order to analytically compute—estimate dynamic power dissipation of a circuit and it is sometimes indirectly expressed in the formulas as $C_{switched}$, which is the product of activity factor and load capacitance of a node C_L. In some literature, symbol α is used instead of A.

Energy (E) is generally defined as the ability of a physical system to perform a work on other physical systems and its SI unit is the *Joule* (*J*). The total energy consumption of a digital circuit can be expressed as the sum of two components: dynamic energy (E_{dyn}) and static energy (E_{stat}).

Dynamic energy has three components which are results of the next three sources: charging/discharging capacitances, short-circuit currents, and glitches. For digital circuits analysis, the most relevant energy is one which is needed to charge a capacitor (transition 0→1), as the other components are parasitic; thus, we cannot affect them significantly with architectural-level low-power techniques. For that reason, in the rest of this chapter, the term dynamic energy is referred to the energy spent on charging/discharging capacitances. According to the general energy definition, dynamic energy in digital circuits can be interpreted as: When a transition in a digital circuit occurs (a node changes its state from 0 to 1 or from 1 to 0), some amount of electrical work is done; thus, some amount of electrical energy is spent. In order to obtain analytical expression of dynamic energy, a network node can be modeled as a capacitor C_L which is charged by voltage source V_{DD} through a circuit with resistance R. In this case, the total energy consumed to charge the capacitor C_L is:

$$E = C_L V_{DD}^2 \tag{1}$$

where the half of the energy is dissipated on R and half is saved in C_L,

$$E_C = E_R = \frac{CV_{DD}^2}{2}. \tag{2}$$

The total static energy consumption of a digital network is the result of leakage and static currents. Leakage current I_{leak} consists of drain leakage, junction leakage, and gate leakage current, while static current I_{DC} is DC bias current which is needed by some circuits for their correct work. Static energy at a time moment $t(t > 0)$ is given as follows:

$$E(t) = \int_0^t V_{DD}(I_{leak} + I_{DC})\mathrm{d}\tau = V_{DD}(I_{DC} + I_{leak})t. \quad (3)$$

As CMOS technology advances into sub-100 nm, leakage energy is becoming as important as dynamic energy (or even more important).

Power (P) is the rate at which work is performed or energy is converted, and its SI unit is the *Watt* (*W*). Average power (which is, for our purpose, more important than instantaneous power) is given with the formula: $P = \frac{\Delta E}{\Delta t}$, in which ΔE is amount of energy consumed in time period Δt. Power dissipation sources in digital circuits can be divided into two major classes: dynamic and static. The difference between the two is that the former is proportional to the activity in the network and the switching frequency, whereas the latter is independent of both.

Dynamic power dissipation, like dynamic energy consumption, has several sources in digital circuits. The most important one is charging/discharging capacitances in a digital network and it is given as:

$$P_{dyn} = AC_L V_{DD}^2 f, \quad (4)$$

in which f is the switching frequency, while A, C_L, and V_{DD} were defined before. The other sources are results of short-circuit currents and glitches, and they are not going to be discussed due to the above-mentioned reasons.

Static power in CMOS digital circuits is a result of leakage and static currents (the same sources which cause static energy). Static power formula is given as follows:

$$P_{stat} = V_{DD}(I_{DC} + I_{leak}). \quad (5)$$

Another related metric is surface power density, which is defined as power per unit area and its unit is $\frac{W}{m^2}$. This metric is the crucial one for thermal studies and cooling system selection and design, as it is related with the temperature of the given surface by *Stefan–Boltzmann law* [6].

2.1.2 Derived Metrics

In today's design environment where both delay and energy play an almost equal role, the basic design metrics may not be sufficient. Hence, some other metrics of potential interest have been defined.

Energy-Delay Product (EDP) Low power often used to be viewed as synonymous with lower performance that, however, in many cases, application runtime is of significant relevance to energy- or power-constrained systems. With the dual goals of low energy and fast runtimes in mind,

EDP was proposed as a useful metric [7]. *EDP* offers equal "weight" to energy and performance degradation. If either energy or delay increases, the *EDP* will increase. Thus, lower *EDP* values are desirable.

Energy^i-Delay^j Product (E^iD^jP) *EDP* shows how close the design is to a perfect balance between performance and energy efficiency. Sometimes, achieving that balance may not necessarily be of interest. Therefore, typically one metric is assigned greater weight, for example, energy is minimized for a given maximum delay or delay is minimized for a given maximum energy. In order to achieve that, we need to adjust exponents i and j in E^iD^jP. In high-performance arena, where performance improvements may matter more than energy savings, we need a metric which has $i < j$, while in low-power design we need one with $i > j$.

2.2 Architectural-Level Metrics

$\frac{MIPS}{Watt}$ Millions of Instructions Per Second (MIPS) per Watt is the most common (and perhaps obvious) metric to characterize the power-performance efficiency of a microprocessor. This metric attempts to quantify efficiency by projecting the performance achieved or gained (measured in MIPS) for every watt of power consumed. Clearly, the higher the number, the "better" the machine is.

$\frac{MIPS^i}{Watt}$ While the previous approach seems a reasonable choice for some purposes, there are strong arguments against it in many cases, especially when it comes to characterizing high-end processors. Specifically, a design team may well choose a higher frequency design point (which meets maximum power budget constraints) even if it operates at a much lower $\frac{MIPS}{W}$ efficiency compared to one that operates at better efficiency but at a lower performance level. As such, $\frac{MIPS^2}{Watt}$ or even $\frac{MIPS^3}{Watt}$ may be appropriate metric of choice. On the other hand, at the lowest end (low-power case), designers may want to put an even greater weight on the power aspect than the simplest MIPS/Watt metric. That is, they may just be interested in minimizing the power for a given workload run, irrespective of the execution time performance, provided the latter does not exceed some specified upper limit.

Energy-per-Instruction (EPI) One more way of expressing the relation between performance (expressed in number of instructions) and power/energy.

$\frac{MFLOPS^i}{Watt}$ While aforementioned metrics are used for all computer systems in general, when we consider scientific and supercomputing, $\frac{MFLOPS^i}{Watt}$ is the

most common metric for power-performance efficiency, where Millions of Floating point Operations Per Second (MFLOPS) is a metric for floating point performance.

3. CLASSIFICATION OF SELECTED ARCHITECTURE-LEVEL TECHNIQUES

This section presents a classification of existing examples of architectural-level power and energy-efficient techniques. In the first section, the classification criteria are given. The classification criteria were chosen to reflect the essence of the basic viewpoint of this research. Afterward, the classification tree was obtained by application of the chosen criteria. The leaves of the classification are the classes of examples (techniques). The list of the most relevant examples for each class is given in the second section.

3.1 Criteria

The classification criteria of interest for this research as well as the thereof are given in Table 1. All selected classification criteria are explained in the caption of Table 1 and elaborated as follows:

C1 Criterion C1 is the top criterion and divides the techniques by level at which they can be applied, core- or core blocks level. Here, the term "Core" implies processor's core without L1 cache.

C2 This criterion divides core blocks into front- and back-end of the pipeline. By front-end, we assume control units and RF, while back-end assumes functional units. Where an optimization technique optimizes both front- and back-end, we group them together and call them only *pipeline*.

Table 1 Classification Criteria (C1, C2, C3): Hierarchical Level, Core Block Type, and Type of Power/Energy Being Optimized

C1: Hierarchical level	- Core - Functional blocks
C2: Core block type	- Front-end - Back-end
C3: Type of power/energy being optimized	- Dynamic - Static

C1 is a binary criterion (core, functional blocks); C2 is also binary criterion (functional units, control units, and RF); and C3 is, like the previous two criteria, is binary (dynamic, static).

C3 Application of the last criterion gave us the component of the metric (power or energy) that we optimize.

The full classification tree, derived from the above introduced classification criteria, is presented in Fig. 1. Each leaf of the classification tree is given a name. Names on the figure are short form of the full names as it is presented in Table 2.

3.2 List of Selected Examples

For each class (leaf of the classification), the list of the most relevant existing techniques (examples) is given in Table 3. For each selected technique, the past work is listed in Table 3. The techniques are selected using two criteria. The first criterion by which we chose the most important works is the number of citation. In order to obtain this number, Google Scholar [8] was used. Important practical reasons for this are that Google Scholar is freely available to anyone with an Internet connection, has better citation indexing and

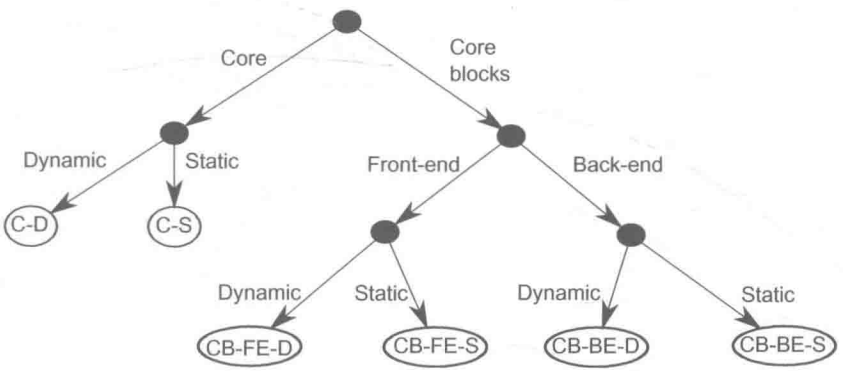


Figure 1 Classification tree. Each leaf represents a class derived by criteria application.

Table 2 Class Short Names Explanations and Class Domains

Short Name	Full Name	Covered Hardware
C-D	Core-Dynamic	Whole core
C-S	Core-Static	
CB-FE-D	Core Blocks-FE-Dynamic	Front-end
CB-FE-S	Core Blocks-FE-Static	
CB-BE-D	Core Blocks-BE-Dynamic	Back-end
CB-BE-S	Core Blocks-BE-Static	

Table 3 List of Presented Solutions
Core-Dynamic

Dynamic Voltage and Frequency Scaling (DVFS)
"Scheduling for reduced CPU energy," M. Weiser, B. Welch, A. J. Demers, and S. Shenker [11]
"Automatic performance setting for dynamic voltage scaling," K. Flautner, S. Reinhardt, and T. Mudge [12]
"The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction," C. Hsu and U. Kremer [13]
"Energy-conscious compilation based on voltage scaling," H. Saputra, M. Kandemir, N. Vijaykrishnan, M. Irwin, J. Hu, C.-H. Hsu, and U. Kremer [14]
"Compile-time dynamic voltage scaling settings: opportunities and limits," F. Xie, M. Martonosi, and S. Malik [15]
"Intraprogram dynamic voltage scaling: bounding opportunities with analytic modeling," F. Xie, M. Martonosi, and S. Malik [16]
"A dynamic compilation framework for controlling microprocessor energy and performance," Q. Wu, V. J. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. W. Clark [17]
"Identifying program power phase behavior using power vectors," C. Isci and M. Martonosi [18]
"Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," C. Isci, G. Contreras, and M. Martonosi [19]
"Power and performance evaluation of globally asynchronous locally synchronous processors," A. Iyer and D. Marculescu [20]
"Toward a multiple clock/voltage island design style for power-aware processors," E. Talpes and D. Marculescu [21]
"Dynamic frequency and voltage control for a multiple clock domain microarchitecture," G. Semeraro, D. H. Albonesi, S. G. Dropsho, G. Magklis, S. Dwarkadas, and M. L. Scott [22]
"Formal online methods for voltage/frequency control in multiple clock domain microprocessors," Q. Wu, P. Juang, M. Martonosi, and D. W. Clark [23]
"Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott [24]