
COMPILERS AND
OPERATING SYSTEMS FOR
LOW POWER

COMPILERS AND OPERATING SYSTEMS FOR LOW POWER

Edited by
LUCA BENINI
University of Bologna

MAHMUT KANDEMIR
The Pennsylvania State University

J. RAMANUJAM
Louisiana State University

Kluwer Academic Publishers
Boston/Dordrecht/London

Contents

List of Figures	xi
List of Tables	xv
Contributing Authors	xvii
Preface	xix
1	
Low Power Operating System for Heterogeneous Wireless Communication System	1
<i>Suet-Fei Li, Roy Sutton, Jan Rabaey</i>	
1 Introduction	2
2 Event-driven versus General-purpose OS	3
2.1 PicoRadio II Protocol Design	3
2.2 General-purpose Multi-tasking OS	4
2.3 Event-driven OS	8
2.4 Comparison Summary	9
3 Low Power Reactive OS for Heterogeneous Architectures	12
3.1 Event-driven Global Scheduler and Power Management	12
3.2 TinyOS Limitations and Proposed Extensions	14
4 Conclusion and Future Work	15
References	16
2	
A Modified Dual-Priority Scheduling Algorithm for Hard Real-Time Systems to Improve Energy Savings	17
<i>M. Angels Moncusí, Alex Arenas, Jesus Labarta</i>	
1 Introduction	17
2 Dual-Priority Scheduling	19
3 Power-Low Modified Dual-Priority Scheduling	21
4 Experimental Results	28
5 Summary	36
References	36
3	
Toward the Placement of Power Management Points in Real-Time Applications	37

Nevine AbouGhazaleh, Daniel Mossé, Bruce Childers, Rami Melhem

1	Introduction	37
2	Model	39
3	Sources of Overhead	40
	3.1 Computing the New Speed	40
	3.2 Setting the New Speed	40
4	Speed Adjustment Schemes	41
	4.1 Proportional Dynamic Power Management	41
	4.2 Dynamic Greedy Power Management	42
	4.3 Evaluation of Power Management Schemes	43
5	Optimal Number of PMPs	44
	5.1 Evaluation of the Analytical Model	45
6	Conclusion	48
	Appendix: Derivation of Formulas	48
	References	51

4

Energy Characterization of Embedded Real-Time Operating Systems 53

Andrea Acquaviva, Luca Benini, Bruno Riccó

1	Introduction	53
2	Related Work	55
3	System Overview	56
	3.1 The Hardware Platform	56
	3.2 RTOS overview	57
4	Characterization Strategy	59
5	RTOS Characterization Results	60
	5.1 Kernel Services	60
	5.2 I/O Drivers	62
	5.2.1 Burstiness Test	62
	5.2.2 Clock Speed Test	63
	5.2.3 Resource Contention Test	64
	5.3 Application Example: RTOS vs Stand-alone	65
	5.4 Cache Related Effects in Thread Switching	66
6	Summary of Findings	66
7	Conclusions	67

References 72

5

Dynamic Cluster Reconfiguration for Power and Performance 75

Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, Taliver Heath

1	Motivation	77
2	Cluster Configuration and Load Distribution	78
	2.1 Overview	78
	2.2 Implementations	81
3	Methodology	83
4	Experimental Results	84
5	Related Work	89
6	Conclusions	91

<i>Contents</i>	vii
References	91
6	
Energy Management of Virtual Memory on Diskless Devices	95
<i>Jerry Hom, Ulrich Kremer</i>	
1 Introduction	96
2 Related Work	97
3 Problem Formulation	98
4 EEL _{RM} Prototype Compiler	100
4.1 Phase 1 - Analysis	100
4.2 Phase 2 - Code Generation	101
4.3 Performance Model	102
4.4 Example	102
4.5 Implementation Issues	103
5 Experiments	105
5.1 Benchmark Characteristics	106
5.2 Simulation Results	107
6 Future Work	110
7 Conclusion	111
References	111
7	
Propagating Constants Past Software to Hardware Peripherals on Fixed-Application Embedded Systems	115
<i>Greg Stitt, Frank Vahid</i>	
1 Introduction	116
2 Example	119
3 Parameters in Cores	120
4 Propagating Constants from Software to Hardware	123
5 Experiments	125
5.1 8255A Programmable Peripheral Interface	126
5.2 8237A DMA Controller	127
5.3 PC16550A UART	128
5.4 Free-DCT-L Core	128
5.5 Results	131
6 Future Work	133
7 Conclusions	134
References	134
8	
Constructive Timing Violation for Improving Energy Efficiency	137
<i>Toshinori Sato, Itsujiro Arita</i>	
1 Introduction	137
2 Low Power via Fault-Tolerance	139
3 Evaluation Methodology	143
4 Simulation Results	143
5 Related Work	147
6 Conclusion and Future Work	151
References	151

9

Power Modeling and Reduction of VLIW Processors 155

Weiping Liao, Lei He

1	Introduction	155
2	Cycle-Accurate VLIW Power Simulation	156
	2.1 IMPACT Architecture Framework	156
	2.2 Power Models	157
	2.3 PowerImpact	158
3	Clock Ramping	159
	3.1 Clock Ramping with Hardware Prescan (<i>CRHP</i>)	160
	3.2 Clock Ramping with Compiler-based Prediction (<i>CRCP</i>)	162
	3.2.1 Basic CRCP Algorithm	162
	3.2.2 Reduction of Redundant Ramp-up Instructions	164
	3.2.3 Control Flow	165
	3.2.4 Load Instructions	165
4	Experimental Results	165
5	Conclusions and Discussion	169
	References	170

10

Low-Power Design of Turbo Decoder with Exploration of Energy-Throughput Trade-off 173

Arnout Vandecappelle, Bruno Bougard, K.C. Shashidhar, Francky Catthoor

1	Introduction	173
2	Data Transfer and Storage Exploration Methodology	176
3	Global Data Flow and Loop Transformations	178
	3.1 Removal of Interleaver Memory	178
	3.2 Enabling Parallelism	179
4	Storage Cycle Budget Distribution	180
	4.1 Memory Hierarchy Layer Assignment	181
	4.2 Data Restructuring	182
	4.3 Loop Transformations for Parallelization	183
	4.3.1 Loop Merging	183
	4.3.2 Loop Pipelining	184
	4.3.3 Partial Loop Unrolling	184
	4.3.4 Loop Transformation Results	185
	4.4 Storage Bandwidth Optimization	185
5	Memory Organization	186
	5.1 Memory Organization Exploration	186
	5.2 Memory Organization Decision	188
6	Conclusions	190
	References	190

11

Static Analysis of Parameterized Loop Nests for Energy Efficient Use of Data Caches 193

Paolo D'Alberto, Alexandru Nicolau, Alexander Veidenbaum, Rajesh Gupta

1	Introduction	193
2	Energy and Line Size	195
3	Background	195
4	The Parameterized Loop Analysis	197

<i>Contents</i>		ix
4.1	Reduction to Single Reference Interference	199
4.2	Interference and Reuse Trade-off	200
5	STAMINA Implementation Results	200
5.1	Swim from SPEC 2000	201
5.2	Self Interference	201
5.3	Tiling and Matrix Multiply	202
6	Summary and Future Work	203
	References	203
12		
	A Fresh Look at Low-Power Mobile Computing	209
	<i>Michael Franz</i>	
1	Introduction	209
2	Architecture	211
3	Handover and the Quantization of Computational Resources	212
3.1	Standardization of Execution Environment's Parameters	214
3.2	A Commercial Vision: Impact on Billing, Customer Loyalty and Churn	215
4	Segmentation of Functionality: The XU-MS Split	215
4.1	Use of Field-Programmable Hardware in the Mobile Station	217
4.2	Special End-To-End Application Requirements	217
5	Status and Research Vision	218
	References	219
	Index	221

List of Figures

1.1	Model of computation for PicoRadio protocol stack	5
1.2	Implementing PicoRadio II with VCC	6
1.3	Code generation with general-purpose eCOS	7
1.4	PicoRadio II chip floorplan. Xtensa is the embedded microprocessor	7
1.5	Implementing PicoRadio II Protocol stacks in TinyOS. Arrows show events/commands propagation in the system	9
1.6	Total cycle count comparison: General-purpose versus event-driven OS. Key at right identifies system components	10
1.7	Percentage breakdown comparison: General-purpose versus event-driven OS. Key at right identifies system components	11
1.8	Behavior diagram of the PicoRadio sensor node	13
1.9	Architectural diagram for PicoRadio sensor node	14
2.1	Pseudo code for Power Low Modified Dual-Priority (PLMDP) Scheduling	22
2.2	Maximum extension time in three different situations	24
2.3	Execution time in LPFPS when all tasks use 100% WCET	25
2.4	Execution time in PLMDP when all tasks use 100% WCET	25
2.5	Execution time in LPFPS when all tasks use 50% WCET	27
2.6	Execution time in PLMDP when all tasks use 50% WCET	28
2.7	Comparison of both algorithms in the task set proposed by Shin and Choi [4]	28
2.8	Comparison of both algorithms when the workload of the system is 80%	30
2.9	System workload variation when all tasks consume the 100% of WCET	31
2.10	System workload variation when all tasks consume the 50% of WCET	31
2.11	System workload and harmonicity of the tasks periods variation	32

2.12	Maximum task workload variation Non-harmonics periods	32
2.13	Tmin/Tmax variation	33
2.14	Comparison of both algorithms in the avionics task set [9]	34
2.15	Comparison of both algorithms in the INS task set [10]	35
2.16	Comparison of both algorithms in the CNC task set [11]	35
3.1	Actual execution times of a task set using the Static, Proportional and Dynamic Greedy schemes	42
3.2	Total energy consumption for different schemes versus the number of PMPs	43
3.3	Total energy consumption for the Proportional scheme versus the number of PMP	46
3.4	Total energy consumption for Dynamic Greedy scheme versus the number of PMPs	46
4.1	The hardware platform: HP SmartBadgeIV	57
4.2	The software layer: eCos structure	58
4.3	Thread switch experiment: Energy consumption for different clock frequencies at the maximum switching frequency	62
4.4	Energy consumption of the audio driver for different clock speeds at fixed data burstiness	64
5.1	Cluster evolution and resource demands for the WWW server	85
5.2	Power consumption for the WWW server under static and dynamic cluster configurations	86
5.3	Cluster evolution and resource demands for the WWW server	86
5.4	Cluster evolution and resource demands in the power-aware OS	88
5.5	Power consumption for the power-aware OS under static and dynamic cluster configurations	88
5.6	Cluster evolution and resource demands in the power-aware OS	89
6.1	Comparison of compiler vs. OS directed power management	99
6.2	Sample code	103
6.3	Partial view of <i>tomcatv</i> 's page fault behavior during execution	108
6.4	One iteration of <i>tomcatv</i> 's primary, outermost loop	109
7.1	Core-based embedded system design	116
7.2	A simple example of propagating constants to hardware (a) soft core, (b) synthesized core structure, (c) synthesized core structure after propagating constants <code>cont_reg(0)=0</code> and <code>cont_reg(1)=1</code>	121
7.3	The Intel 8255A parallel peripheral interface	122
7.4	Method for propagating constants to peripheral cores	124

7.5	Block diagram of DCT core	129
8.1	ALU utilizing proposed technique	141
8.2	Clock signals	142
8.3	Energy consumption (Squash) 164.zip	144
8.4	Energy consumption (Squash) 175.vpr	144
8.5	Energy consumption (Squash) 176.gcc	144
8.6	Energy consumption (Squash) 186.crafty	145
8.7	Energy consumption (Squash) 197.parser	145
8.8	Energy consumption (Squash) 252.eon	145
8.9	Energy consumption (Squash) 255.vortex	146
8.10	Energy consumption (Squash) 256.bzip2	146
8.11	Energy consumption (Reissue) 164.zip	148
8.12	Energy consumption (Reissue) 175.vpr	148
8.13	Energy consumption (Reissue) 176.gcc	148
8.14	Energy consumption (Reissue) 186.crafty	149
8.15	Energy consumption (Reissue) 197.parser	149
8.16	Energy consumption (Reissue) 252.eon	149
8.17	Energy consumption (Reissue) 255.vortex	150
8.18	Energy consumption (Reissue) 256.bzip2	150
9.1	Flow diagram for IMPACT	156
9.2	Overall structure of PowerImpact	159
9.3	The relationship of states	161
9.4	Utilization rate for FPUs	161
9.5	Distribution of instruction numbers in bundles, with maximum bundle width = 6	162
9.6	Insert ramp-up instructions	163
9.7	Insertion of ramp-up instructions beyond the current hyperblock	164
9.8	Performance loss (in percentage as the Z-axis variable) of CRHP and CRCP approaches for <i>equake</i>	166
9.9	Power reduction (in percentage as the Z-axis variable) of CRHP and CRCP approaches for <i>equake</i>	166
9.10	Performance loss (in percentage as the Z-axis variable) of CRHP and CRCP approaches for <i>art</i>	166
9.11	Power reduction (in percentage as the Z-axis variable) of CRHP and CRCP approaches for <i>art</i>	167
9.12	Performance loss (in percentage) for $T_r = 10$ and $T_a = 16$	167
9.13	Power reduction (in percentage) for $T_r = 10$ and $T_a = 16$	168

9.14	Performance Loss (in percentage) before and after the amendment for load instruction, for $T_r = 10$, $T_a = 16$ and $T_p = 9$	169
10.1	Turbo coding-decoding scheme	174
10.2	Energy-performance trade-off	177
10.3	Transformed data flow of turbo decoding scheme	178
10.4	Parallelization of the MAP algorithm	179
10.5	Turbo decoding data flow and timing	180
10.6	Dependencies between memory accesses of two loops	183
10.7	Dependencies after merging the two loops of Figure 10.6	184
10.8	Dependencies after pipelining the merged loop of Figure 10.7	185
10.9	Pareto curves for 7 workers, for two and for seven dual-port memories per worker	187
11.1	Grid cells and band cells in a plane	198
11.2	Tiling of Matrix Multiply. 6 parameters: loop bounds and A,B and C offsets	205
11.3	SWIM: calc1() in C code	206
11.4	Matrix Multiply. Two parameters: loop bounds and A offset	206
11.5	Self interference and analysis results	207
12.1	System architecture	212

List of Tables

1.1	General comparison	9
1.2	Memory requirements comparison	10
2.1	Benchmark task set used by Shin and Choi [4]	24
2.2	Avionics benchmark task set [9]	33
2.3	INS benchmark task set [10]	33
2.4	CNC benchmark task set [11]	34
3.1	Theoretical versus Simulation choice of optimal number of PMPs for the Proportional scheme	47
3.2	Theoretical versus Simulation choice of optimal number of PMPs for the Dynamic Greedy scheme	47
4.1	Thread switch experiment: Energy variation due to different switching frequencies with a fixed clock frequency (103.2Mhz)	62
4.2	Audio driver average power consumption due to different level of data burstiness at a fixed clock frequency	63
4.3	Average power consumption of the wireless LAN driver due to different level of data burstiness at a fixed clock frequency	64
4.4	Variation of the energy consumed by the audio driver in presence of device contention for different switch frequencies	65
4.5	Comparison between the energy consumed by two version of the speech enhancer: OS based and stand-alone	65
4.6	Testing parameters for the experiment related to Tables 4.7 thru 4.9	65
4.7	Energy consumption of thread management and scheduler functions at minimum and maximum clock frequencies	68
4.8	Energy consumption of thread communication and synchronization functions at minimum and maximum clock frequencies	69

4.9	Energy consumption of time management functions at minimum and maximum clock frequencies	71
4.10	Energy cost of thread switching in presence of cache-related effects	72
6.1	Page faults for different memory sizes in terms of pages, assuming that each array requires 4 pages of memory space	103
6.2	Dynamic page hit/miss prediction accuracy	105
6.3	Benchmark parameters	106
6.4	Relative energy consumption of benchmark programs with EEL_{RM} energy management. Energy values are percentages of OS approach. Active WaveLAN card contributes 40% to overall energy budget	107
6.5	Relative performance of benchmark programs under OS or EEL_{RM} energy management. Reported values are percentages of ∞ threshold — card always awake	110
7.1	Comparison of cores before and after constant propagation	132
8.1	Processor configuration	143
8.2	Benchmark programs	146
9.1	Partitions in our power models	158
9.2	System configuration for experiments	163
10.1	Data structures, sizes and memory hierarchy layer assignment. N is the window size, M is the number of workers, $2NM$ is the size of one frame which is iteratively decoded	181
10.2	Data structures, sizes and memory hierarchy layer assignment after data restructuring. $2N$ is the size of one worker. Each of these data structures exists M times, i.e. once for each worker	182
10.3	Effect of parallelizing loop transformations on maximally achievable throughput and latency	186
10.4	Memories architecture with simulated access energy and number of accesses per frame	189
11.1	Self interference example	201
11.2	Interference table, for the procedure in Figure 11.4	202
11.3	Interference table for the procedure <i>ijk_matrix_multiply_4</i> in Figure 11.2	202
12.1	Different classes of execution units and applicable usage scenarios	213

Contributing Authors

Nevine AbouGhazaleh	University of Pittsburgh, USA
Andrea Acquaviva	University of Bologna, Italy
Alex Arenas	Universitat Rovira i Virgili, Spain
Itsujiro Arita	Kyushu Institute of Technology, Japan
Luca Benini	University of Bologna, Italy
Ricardo Bianchini	Rutgers University, USA
Bruno Bougard	IMEC, Belgium
Francky Catthoor	IMEC, Belgium
Bruce Childers	University of Pittsburgh, USA
Paolo D’Alberto	University of California–Irvine, USA
Michael Franz	University of California–Irvine, USA
Rajesh Gupta	University of California–Irvine, USA
Taliver Heath	Rutgers University, USA
Lei He	University of California–Los Angeles, USA
Jerry Hom	Rutgers University, USA
Ulrich Kremer	Rutgers University, USA
Jesus Labarta	Universitat Politecnica de Catalunya, Spain
Weiping Liao	University of California–Los Angeles, USA
Suet-Fei Li	University of California–Berkeley, USA
Rami Melhem	University of Pittsburgh, USA
M. Angels Moncusí	Universitat Rovira i Virgili, Spain
Daniel Mossé	University of Pittsburgh, USA
Alexandru Nicolau	University of California–Irvine, USA
Eduardo Pinheiro	Rutgers University, USA
Jan Rabaey	University of California–Berkeley, USA
Bruno Riccó	University of Bologna, Italy
Toshinori Sato	Kyushu Institute of Technology, Japan
K.C. Shashidhar	IMEC, Belgium
Greg Stitt	University of California–Riverside, USA
Roy Sutton	University of California–Berkeley, USA
Enrique V. Carrera	Rutgers University, USA
Frank Vahid	University of California–Riverside, USA
Arnout Vandecappelle	IMEC, Belgium
Alexander Veidenbaum	University of California–Irvine, USA

Preface

In the last ten years, power dissipation has emerged as one of the most critical issues in the development of large-scale integrated circuits, and electronic systems in general. Technology scaling is not the only cause for this trend: in fact, we are moving toward a world of pervasive electronics, where our cars, houses, and even our environment and our bodies will be linked in a finely-knit network of communicating electronic devices capable of complex computational tasks materializing a vision of “ambient intelligence,” the ultimate goal of embedded computing. Today, power consumption is probably the main obstacle in the realization of this vision: current electronic systems still require too much power to perform critical ambient intelligence tasks (e.g., voice processing, vision, wireless communication). For this reason, power, or energy (i.e., power-performance ratio) minimization is now aggressively targeted in all the phases of electronic system design.

While early low-power (or energy-efficient) design focused on technology and hardware optimization, it is now clear that software power optimization is an equally critical goal. Most of complex integrated systems are highly programmable. In fact, the new millennium has seen the rapid diffusion of embedded processor cores as the basic computational workhorse for large-scale integrated systems on silicon, and today we are witnessing the rebirth of multiprocessor architectures, fully integrated on a single silicon substrate. It is therefore obvious that the power consumption of integrated systems dominated by core processors and memories is heavily dependent on the applications they run and the middleware supporting them.

In general, we can view the software infrastructure as layered in applications and run-time support middleware (often called “operating system”). Applications control the user-level functionality of the system, but they interface to the SoC platform via hardware abstraction layers provided by the middleware. Software energy minimization can be tackled with some hope of success only if both application-level software and middleware are both optimized for maximum energy efficiency. The Compilers and Operating Systems for Low Power (COLP) Workshop aims at creating a forum that brings together researchers operating in both application-level energy optimization and low-power operat-

ing systems. The main objective of this initiative is to create opportunities for cross-fertilization between closely related areas that can greatly benefit from a tighter interaction. Papers presented at COLP are work-in-progress and are selected based on their potential for stimulating thoughts and creative discussions.

This book is the result of a careful (and sometimes painful) process of selection and refinement of the most significant contributions to the 2001 edition of COLP. The editors have first selected the papers based both on reviewer evaluations and on feedback from the audience at the oral presentation. They have then solicited an extended version of the papers, in a format more suitable for archival publications. The extended versions have then been reviewed by the editors to ensure consistency. The results of this “distillation” process have been collected in this book, which we hope will bring the reader a wealth of fresh and valuable ideas for further research as well as technology transfer.

Organization

The book is divided into twelve chapters. The first six chapters focus on low energy operating systems, or more in general, energy-aware middleware services. The following five chapters are centered on compilation and code optimization. Finally, the last chapter takes a more general viewpoint on mobile computing.

Chapter 1, entitled “Low Power Operating System for Heterogeneous Wireless Communication Systems” is contributed by Suet-Fei Li, Roy Sutton, and Jan Rabaey, from UC Berkeley. The chapter describes an ultra-low overhead operating system for wireless microsensors and compares it with more traditional embedded operating systems.

Chapter 2, “Low Power Approach in a Modified Dual Priority Scheduling for Hard Real-Time Systems” (by M. Angels Moncusí, A. Arenas, and J. Labarta from Universitat Rovira i Virgili and Universitat Politecnica de Catalunya) deal with task scheduling, one of the most classical problems in real-time operating systems, and investigates a novel dual-priority algorithm with high energy efficiency.

The third chapter, contributed by N. Nevine AbouGazelah, D. Mossé, R. Melhem, and B. Childers (from University of Pittsburgh) entitled “A Restricted Model for the Optimal Placement of Power Management Points in Real Time Applications” deals with an important issue at the boundary between applications and operating systems, namely the optimal insertion of systems calls that dynamically change the supply voltage (and operating frequency) during the execution of an application.

The fourth chapter, by A. Acquaviva, L. Benini and B. Riccò (Università di Bologna), is entitled “Energy Characterization of Embedded Real-Time Oper-

ating Systems.” The chapter describes a methodology for characterizing the energy cost of most primitives and function calls in embedded operating systems.

Chapter 5, by E. Pinheiro, R. Bianchini, E. Carrera and T. Heath (Rutgers University), is entitled “Load Balancing and Unbalancing for Power and Performance in cluster-Based Systems” and it deals with an important emerging topic, namely low-energy multiprocessors. The chapter gives a fresh look at load balancing issues in cluster-based systems when energy constraints are tight.

Chapter 6 closes the first group. It is entitled “Energy Management of Virtual Memory on Diskless Devices” (by J. Hom and U. Kremer, Rutgers University) and it deals with virtual memory, one of the basic hardware abstraction layers provided by standard operating systems.

The next chapter, entitled “Propagating Constants Past Software to Hardware Peripherals in Fixed-Application Embedded Systems,” contributed by G. Stitt and F. Vahid, discusses how propagating application-level constant to hardware improves both power and form factor, leading to up to 2-3 times reductions in peripheral size.

In Chapter 8, entitled “Constructive Timing Violation for Improving Energy Efficiency,” T. Sato and I. Arita present a technique that relies on a fault-tolerance mechanism and speculative execution to save power. Their technique, called *constructive timing violation*, guarantees that the timing constraints for critical paths are not violated.

In the next chapter, entitled “Power Modeling and Reduction of VLIW Processors,” the authors W. Liao and L. He present an in-depth study of power behavior of a VLIW architecture, and develop an infrastructure which can be used for architecture-based as well as compiler studies.

Chapter 10, entitled “Low Power Design of Turbo Decoder Module with Exploration of Power-Performance Tradeoffs,” demonstrates how a systematic data transfer and storage exploration methodology helps characterize energy and performance behavior of Turbo Coding. Vandecappelle, Bougard, Shashidbar, and Catthoor also discuss the cycle budget-energy tradeoff.

In the next chapter, “Static Analysis of Parameterized Loop Nests for Energy Efficient Use of Data Caches,” P. D’Alberto, A. Nicolau, A. Veidenbaum, and R. Gupta demonstrate that the compiler analysis of loop with regular access patterns can reveal useful information for optimizing power.

Finally, in Chapter 12, entitled “A Fresh Look at Low-Power Mobile Computing,” M. Franz presents a technique that allows large portions of applications to be offloaded to a base station for execution.

We believe that, with the proliferation of power-constrained devices, energy optimizations will become even more important in the future. Consequently, it is hard to imagine that architectural and circuit-level optimizations alone will

provide the required level of energy efficiency for demanding applications of next generation computing. The research papers presented here do not only demonstrate state-of-the-art, but they also prove that, to obtain the best energy/performance characteristics, compiler, system software, and architecture must work together.

Acknowledgments

This book grew out of the Workshop on Compilers and Operating Systems, 2001 (COLP 01). We acknowledge the active contribution of the program committee of COLP 01: Eduard Ayguade, R. Chandramouli, Bruce Childers, Marco Cornero, Rudi Eigenmann, Manish Gupta, Rajiv Gupta, Mary Jane Irwin, Uli Kremer, Rainer Leupers, Diana Marculescu, Enric Musoll, Anand Sivasubramaniam, Mary Lou Soffa, Vamsi K. Srikantam, Chau-Wen Tseng, Arnout Vandecappelle, and N. Vijaykrishnan. In addition, we thank the following reviewers for their thoughtful reviews of the initial submissions to the workshop: Bharadwaj Amrutur, Eui Young Chung, Anoop Iyer, Miguel Miranda, Phillip Stanley-Marbell, Emil Talpes, Chun Wong, and Peng Yang. The feedback from the audience at the COLP 01 workshop is greatly appreciated.

We sincerely thank Alex Greene and Melissa Sullivan, and the editorial team at Kluwer for their invaluable help, enthusiasm and encouragement throughout this project. We gratefully acknowledge the support of the U.S. National Science Foundation through grants CCR-9457768, CCR-0073800, and CCR-0093082 during this project.

LUCA BENINI, MAHMUT KANDEMIR, J. RAMANUJAM