Hardware Software Co-Design of a Multimedia SOC Platform

Hardware Software Co-Design of a Multimedia SOC Platform

by

Sao-Jie Chen
National Taiwan University, Taipei, Taiwan, ROC

Guang-Huei Lin
National Taiwan University, Taipei, Taiwan, ROC

Pao-Ann Hsiung
National Chung Cheng University, Chia-Yi, Taiwan, ROC

and

Yu-Hen Hu
University of Wisconsin-Madison, Madison, USA

Springer

Prof. Sao-Jie Chen
National Taiwan University
Dept. Electrical Engineering
Graduate Inst. of Electronics
Engineering
Taipei 106
Taiwan R.O.C.
csj@cc.ee.ntu.edu.tw

Dr. Guang-Huei Lin
National Taiwan University
Dept. Electrical Engineering
Graduate Inst. of Electronics
Engineering
Taipei 106
Taiwan R.O.C.

Dr. Pao-Ann Hsiung
National Chung Cheng
University
Chia-Yi 621
Taiwan R.O.C.

Dr. Yu-Hen Hu
University of Wisconsin,
Madison
Dept. Electrical & Computer
Engineering
1415 Engineering Drive
Madison WI 53706
2556 Engineering Hall
USA

ISBN: 978-1-4020-9622-8 e-ISBN: 978-1-4020-9623-5

DOI 10.1007/978-1-4020-9623-5

Library of Congress Control Number: 2008941282

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Foreword

System-level design is a key design technology to realize integrated systems of various types, possibly integrated on a single die, i.e., *Systems on Chips*, or assembled in a single package, i.e., *Systems in Package*. As technology advances and systems become increasingly more complex, the use of high-level abstractions and platform-based design becomes more and more a necessity. Indeed the productivity of designers increases with the abstraction level, as demonstrated by practices in both the software and hardware domains. The use of high-level models and platforms allow designers to be productive, even when they have weaker specific skills in circuit design technology.

Software plays a key role in embedded system design and is crucial for system programmability and flexibility. The latter factor is extremely important for extending the life of components across different families of products. Compilation is the crucial technology to achieve effective system operation on platforms, starting from high-level programming constructs. The compiler technology has evolved tremendously, especially in the domain of *application-specific instruction set processor* design which is commonplace in embedded platforms.

Effective software compilation is the companion to hardware compilation, also called high-level synthesis. Both share important objectives, like addressing real-time constraints and system performance. The software and hardware design technologies show complementarities and find their most natural applications on platform-based design.

Multi-media systems benefit largely from modular and flexible realizations relying on platforms. The market for these systems is rapidly expanding and its future hinges in part on the industrial capability to deliver advanced systems with rapid turn-around time. In this perspective, the technologies described in this book are very significant for progress of science and technology and have direct impact on applications ranging from entertainment to medical imaging, from terrestrial environmental monitoring to defense.

Lausanne — Giovanni De Micheli
November 25, 2008

Preface

This book is the outcome of a recent international collaborative research project aiming at developing both the hardware and software of a platform based SoC (System-on-Chip) architecture for embedded multimedia systems. Since its inception a decade ago, SoC has captured the attentions of application specific integrated circuit (ASIC) design houses, computer aided design (CAD) companies, and embedded system developers. In particular, the immense popularity of killer multimedia gadgets such as iPod, and iPhone has fueled unprecedented interests in developing new generation multimedia SoC systems.

However, the high level of integration also brings great challenges to system designers: Conventional component-oriented design methodologies can no longer handle the ever increasing system complexity. Hardware and software are necessarily becoming convergent and must be fully concurrent design endeavors. Hardware engineers must understand higher level signal processing algorithms, functional simulations, and design verifications. Software engineers, on the other hand, must pay great attention to heterogeneous instruction set architectures, timing and power constraints, and hardware-in-the-loop system level simulation and verifications. All these point to a diverse body of knowledge, and skills that a competent SoC designer must be equipped. Currently, such valuable knowledge sources are scattered in many different text books, research papers, and other on-line sources. For someone who is interested in gaining a comprehensive overview of issues related to the hardware and software development of multimedia SoC, a highly integrated book is unavailable.

This book is written to serve this purpose. Based on a joint research project coordinated by S. J. Chen at the National Taiwan University (NTU) in Taiwan, this book is co-authored by key participants of a project entitled: Implementation of a Multimedia SoC Platform for Scalable Power-Aware Custom Embedded Systems. In this 3-year research project, we focused on a novel SoC platform based on a Subword-Parallel Single Instruction Multiple Data (SWP-SIMD) micro-architecture. Much of the materials included in this monograph are drawn from outcomes of this research project. A distinct feature of this book is to incorporate a very diverse list of subjects and tied them together elegantly to the development of an SWP-SIMD based SoC platform. Specifically, subject areas that covered include system level design methodologies, H.264 video coding algorithms, sub-word parallel micro-architecture design, SIMD vectorized compiler technology, and real time operating

systems. Obviously, with limited space, it is impossible to engage in-depth discussion of each subject area. Instead, the authors' approach is to provide sufficient detail so that readers may appreciate the significance of each topic and understand the relations of a particular subject to other topics. Plenty of references are provided so that interested readers may pursue further investigation using materials presented in this book as a stepping stone. We understand that such an ambitious goal would not be easy to reach. We hope our efforts will make some tangible contributions toward promoting the SoC platform design and applications.

This book is written for engineers who are working in integrated circuit design houses, in computer aided design tool companies, and embedded system design companies; as well as graduate students who are pursuing a career in computer engineering, multimedia system implementation and related field. We will be very happy to hear readers' feedback and comments.

Taipei, Taiwan — Sao-Jie Chen
Taipei, Taiwan — Guang-Huei Lin
Chia-Yi, Taiwan — Pao-Ann Hsiung
Madison, Wisconsin — Yu-Hen Hu

Acknowledgements

The authors are greatly thankful to the foreword writer, Professor and Director Giovanni De Micheli at the Institute of Electrical Engineering and the Integrated Systems Center at the École Polytechnique Fédérale de Lausanne, Switzerland.

The authors are also grateful to those who have made great contributions to the development of this book. The PLX instruction set architecture was originally developed by Professor Ruby Lee at the Princeton University. She has provided numerous valuable inputs to the design of the PLX core. She also generously provided us with an LCC compiler, Assembler, and Loader for the PLX architecture.

Many current and former graduate students in our research groups have made tangible contributions to portions of this book. Specifically, Ya-Nan Wen and Chia-Wen Tsai of National Taiwan University have made significant contributions to the PLX and H.264 chip implementations. Xiao-Long Wu, Cheng-Cho Jean, and Hong-Kuei Chen of National Taiwan University have contributed to the development of the SIMD compiler and profiler tools. Chun-Jen Wei of National Taiwan University contributed to the development of algebraic verification tool. Chao-Sheng Lin of National Chung-Cheng University has developed the RTOS described in this book.

Finally, we would like to express our appreciation to our family members for their encouragement and patience during the writing of this book.

Taipei, Taiwan — Sao-Jie Chen
Taipei, Taiwan — Guang-Huei Lin
Chia-Yi, Taiwan — Pao-Ann Hsiung
Madison, Wisconsin — Yu-Hen Hu

Contents

1 Introduction 1

2 Design Consideration 5
2.1 Platform-Based Design 5
2.1.1 OMAP 8
2.2 System Modeling 10
2.2.1 State-Oriented Models 11
2.2.2 Activity-Oriented Models 12
2.3 Video Coding 15
2.3.1 H.264 Coding Process 17
2.3.2 Motion Estimation 17
2.3.3 Intra Prediction 22
2.3.4 Transform and Quantization 24
2.3.5 De-Blocking Filter 26
2.3.6 Entropy Encoding 27
2.4 Image Processing 27
2.5 Cryptography 31
2.5.1 RSA 32
2.5.2 DES 34
2.5.3 AES 34
2.6 Digital Communication 36
2.7 Multimedia Instruction Set Design 39

3 System Level Design 41
3.1 Abstraction Levels 42
3.1.1 Algorithm Level 42
3.1.2 Architecture Level 42
3.1.3 Behavior Level 44
3.2 Algorithm Level Verification 45
3.2.1 Algebraic Simulation 46
3.2.2 Algebraic Analysis 48
3.2.3 Error Evaluation 48

3.3 Transaction Level Modeling 52
3.4 System Level Development Tools 55
3.4.1 SystemC 56
3.4.2 LISA 58

4 Embedded Processor Design 63
4.1 Specific Instruction-Set 63
4.2 Data Level Parallelism 64
4.2.1 SIMD 64
4.2.2 SWP-SIMD 67
4.3 Instruction Level Parallelism 70
4.3.1 SuperScalar 70
4.3.2 VLIW 71
4.3.3 NISC 74
4.4 Thread Level Parallelism 76
4.4.1 Multi-Threading 76
4.4.2 Multi-Processor 77
4.4.3 Massively Parallel 78

5 Parallel Compiler 81
5.1 Vectorization 81
5.1.1 Dependence Analysis 81
5.1.2 Loop Normalization 83
5.1.3 Loop Transformation 84
5.1.4 Dependence Removal 84
5.1.5 Strongly Connected Components 85
5.1.6 Loop Distribution 86
5.2 Simdization 87
5.2.1 Control Flow Conversion 87
5.2.2 Memory Alignment 88
5.2.3 Permutation Optimization 89
5.2.4 Subword Fusion 90
5.2.5 Matrix Transpose 90
5.2.6 Reduction 90
5.2.7 Loop Unrolling 91
5.3 ILP Scheduling 92
5.3.1 Software Pipelining 92
5.3.2 Basic Block Extension 93
5.3.3 Speculation 93
5.4 Threading 94
5.4.1 Profiling and Analysis 94
5.4.2 Pthread 95
5.4.3 Structuring 97
5.4.4 OpenMP 99

5.5 Compiler Technique 100
5.5.1 Lexical Analysis 100
5.5.2 Syntax Analysis 101
5.5.3 Abstract Syntax 102
5.5.4 Semantic Analysis 103
5.5.5 Symbol-Table Management 104
5.5.6 Intermediate Representation 104
5.5.7 Code Optimization 105
5.5.8 Code Generation 106
5.6 Compiler Infrastructures 106
5.6.1 LCC Compiler Infrastructure 107
5.6.2 GCC Compiler Infrastructure 107
5.6.3 SUIF Compiler Infrastructure 109
5.6.4 IMPACT Compiler Infrastructure 113

6 Implementation of H.264 on PLX 115
6.1 Instruction Set Decision for H.264 115
6.2 Hardware/Software Partitioning 116
6.3 Untimed Virtual Prototype 117
6.4 Timed SystemC Modeling 122
6.5 PLX Chip Design 127

7 Real-Time Operating System for PLX 129
7.1 PRRP Scheduler 131
7.2 Memory Management 133
7.3 Communication and Synchronization Primitives 134
7.4 Multimedia Applications in RTOS for PLX 135
7.5 Application Development Environment 136
7.5.1 Compilers 137
7.5.2 Parser, Locator, Loader, and Startup Code 137
7.5.3 PLX Platform Simulator 139
7.6 Experimental Results 139

8 Conclusion 145

References 147

List of Figures

2.1 System platform design concept 6
2.2 ARM-based wireless communication platform 7
2.3 Mix-and-match methodology 8
2.4 OMAP software architecture 9
2.5 Profiling report 10
2.6 Resource utilization 10
2.7 Mealy finite state machine 11
2.8 Petri-nets of two concurrent processes 12
2.9 Flow chart 12
2.10 CDFG 13
2.11 Compound if-else statements in HTG form 14
2.12 An example of PDG 15
2.13 An example of SDG 16
2.14 H.264 coding process 18
2.15 Variable block size and segmented macroblock 19
2.16 1-D motion estimation architecture 20
2.17 Schedule of data input for each PE 21
2.18 Three-step search 22
2.19 INTRA_4×4 prediction neighbors 23
2.20 INTRA_4×4 prediction modes 23
2.21 INTRA_16×16 prediction modes 24
2.22 Filtered edges of 16×16 luma and 8×8 chroma macroblock 27
2.23 Filters with Gaussian and Semi-Gaussian coefficients 28
2.24 Linear edge and non-linear edge 29
2.25 Absolute difference mask 29
2.26 Symmetric feature 30
2.27 DES encryption flow 35
2.28 AES encryption flow 35
2.29 Receiver front-end architectures 36
2.30 Waveform of ASK modulation 37
2.31 OFDM spectrum 38
2.32 Convolution coding 38
3.1 System level design 41

3.2 A 4×4 Matrix transpose in PLX assembly 47
3.3 Matrix transpose modeled in Mathematica 47
3.4 Result of matrix transpose 47
3.5 An implementation of SAD on the PLX platform 49
3.6 Symbolic verification and error prediction algorithm 50
3.7 Inter-execution of untimed and timed models 55
3.8 SystemC versions and scopes 56
3.9 LISA behavior synthesis process 60
4.1 ILLIAC-IV architecture 65
4.2 Cray-1 architecture 66
4.3 Subword-parallel execution 68
4.4 Power-aware reconfigurable pipeline adder 69
4.5 High-performance subword-parallel design 69
4.6 Pentium 4 processor micro-architecture 71
4.7 A generic two-cluster VLIW architecture 72
4.8 Register file 72
4.9 Bypass path in a 2-issue VLIW 73
4.10 Unbundled branch 74
4.11 Processor architectures 76
4.12 nVIDIA GeForce 8800 79
4.13 Many cores: (**a**) MIT RAW Processor and (**b**) Intel Tera-Flops chip 79
5.1 Dependence graphs: (**a**) original and (**b**) after dependence removal 86
5.2 Load vector by streaming 88
5.3 Interleaved average/difference implementations 89
5.4 Matrix transpose in SWP-SIMD 90
5.5 Software pipelining 92
5.6 Basic block tail duplication 93
5.7 Loop parallelism 97
5.8 Data decomposition threading 98
5.9 Pipeline thread structure 98
5.10 SPMD code example 99
5.11 OpenMP code of SPMD 99
5.12 Main phases of a compiler 100
5.13 Communication between lexical and syntax analyzers 101
5.14 Grammar for simple arithmetic expressions 102
5.15 Parse tree for “pos = init + rate * 100;” 102
5.16 AST for “pos = init + rate * 100;” 103
5.17 AST after semantic analysis 103
5.18 An overview of GCC 108
5.19 SUIF system 110
5.20 IMPACT compiler flow 113
6.1 SAD calculation in a 128-bit SWP-SIMD 117
6.2 A PLX-H.264 virtual prototype platform 118
6.3 PLX chip architecture 127
7.1 RTOS components and relations 130

7.2 Extended data structure for the PRRP scheduler 131
7.3 PRRP scheduling algorithm in RTOS for PLX 132
7.4 Memory layout in RTOS for PLX 133
7.5 Toolchain flow for application development on RTOS for PLX 137
7.6 Toolchain incompatibilities solved by our parser 138
7.7 Data flow among parser, locator, and Loader 138
7.8 Test flow using DCT and IDCT 140
7.9 Start addresses for binary files in the DCT/IDCT example 142
7.10 Two-dimensional matrix input/output of DCT 142
7.11 Segment of multimedia extension instructions for (a) DCT and (b) Quantization 143

List of Tables

2.1 Popular platforms 7
2.2 Quantization table 26
6.1 PSNR in SAD with 3 different resolutions 116