Programming Heterogeneous MPSoCs

Jerónimo Castrillón Mazo
Rainer Leupers

Programming Heterogeneous MPSoCs

Tool Flows to Close the Software Productivity Gap

Springer

Jerónimo Castrillón Mazo
Rainer Leupers
Chair for Software for Systems on Silicon
RWTH Aachen University
Aachen
Germany

ISBN 978-3-319-00674-1 ISBN 978-3-319-00675-8 (eBook)
DOI 10.1007/978-3-319-00675-8
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013940298

© Springer International Publishing Switzerland 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Dedicated to my extended family

Acknowledgments

This book is based on my doctoral dissertation, written at the Chair for Software for Systems on Silicon at the RWTH Aachen University. The work presented here is the result of 6 years of research, filled with interactions with people that enriched my life, both personally and academically. I am glad to start this book by thanking them.

I would like to start with a sincere *danke schön* to my advisor Professor Rainer Leupers. Beyond his excellent guidance in the role of an advisor, I learned many things by working with him, more notably professionalism and pragmatism. I learned to focus on building useful solutions to problems that matter, rather than spending time on problems that are only interesting from a theoretical point of view. I would also like to thank Professor Gerd Ascheid and Professor Heinrich Meyr for their inspiring discussions in the context of joint projects I had the fortune to work in.

It would have been impossible to work out the solutions presented in this book without the support of the MAPS team, including previous doctoral works by Jianjiang Ceng and Hanno Scharwächter and contemporary efforts by my colleagues Weihua Sheng, Anastasia Stulova, Stefan Schürmans and Maximilian Odendahl. I was also lucky to count with high quality students that worked on parts of the solutions presented in this book. Special thanks go to Christian Klein for helping me put together the sequential programming flow, to Ricardo Velásquez for his efforts on earlier versions of the parallel programming flow, to Andreas Tretter for helping me extend the parallel flow to new hardware architectures and to Aamer Shah for his work on multiple applications.

I am grateful to Filippo Borlenghi, Jan Weinstock, Felix Engel, Maximilan Odendahl, Juan Eusse, Luis Murillo and Jovana Jovic for proof-reading parts of this book. Their feedback was decisive to bring this work into its final shape.

Finally, I would like to thank those people that contributed to the completion of this book in a *non technical* form—my friends and my family. Thanks to my good *old* friends Jorge and Sebas with whom I shared the *ups and downs*, typical of a doctoral path. Thanks to my *new* friends Filippo, Luis Gabriel, Max and Eusse for making me feel like at home while being thousands of kilometers away from it. Thanks to my *very old* friends Daniel, Ana María, María Adelaida, Andrés and Juan Pablo for their support from the distance.

My biggest thanks go to my family and my girlfriend Hera. It is difficult to express how thankful I am to my parents for their unconditional support throughout my life—they made me what I am today. It was also very important for me to have my siblings to count on, Juli, Jaco, Sebas and Esteban. With my family so far away, it is impossible to measure what it meant to me to count with Hera during the last 5 years. My most sincere thanks for being there with me *Herita*.

Contents

1 Introduction 1
1.1 Evolution and Trends in Embedded Systems. 2
1.1.1 Embedded Hardware 3
1.1.2 Embedded Software 5
1.1.3 Design Gaps. 6
1.2 Tool Flow Requirements: Closing the SW Gap. 7
1.3 MAPS: A Framework for MPSoC Programming 10
1.4 Programming Tool Flows 11
1.5 Synopsis and Outline 13

2 Background and Problem Definition. 15
2.1 Motivational Example 15
2.2 Preliminaries. 17
2.2.1 Mapping and Scheduling 17
2.2.2 Sequential Code Performance Estimation. 19
2.2.3 Notation. 21
2.3 Multi-Application Problem 21
2.3.1 Definitions 21
2.3.2 Problem Statement 28
2.4 Sequential Code Problem 28
2.4.1 Compilers and Parallelism Extraction 29
2.4.2 Problem Statement 38
2.5 Parallel Code Problem 39
2.5.1 Process Networks and Dataflow Models 39
2.5.2 The C for Process Networks Language 44
2.5.3 KPN Mapping and Scheduling 44
2.5.4 Problem Statement 46
2.6 Software Defined Radio Problem 47
2.6.1 The Nucleus Project 48
2.6.2 Extensions to the Parallel Problem 49
2.6.3 Problem Statement 52
2.7 Synopsis. 52

3 Related Work 53
3.1 Runtime Management 53
3.1.1 Survey of Runtime Managers 53
3.1.2 This Book in Perspective 55
3.2 Parallelism Extraction from Sequential Code 56
3.2.1 General-Purpose Domain 56
3.2.2 Embedded Domain 57
3.2.3 Commercial Solutions 59
3.2.4 This Book in Perspective 59
3.3 Synthesis of Parallel Specifications 60
3.3.1 General-Purpose Parallel Programming Models 60
3.3.2 Embedded Parallel Programming 61
3.3.3 This Book in Perspective 65
3.4 Software Defined Radio 66
3.4.1 SDR Approaches. 66
3.4.2 This Book in Perspective 68
3.5 Multiple Applications. 68
3.5.1 Handling Multiple Applications 69
3.5.2 This Book in Perspective 71

4 MPSoC Runtime Management 73
4.1 OSIP Solution. 73
4.1.1 Mapping and Scheduling Approach. 74
4.1.2 OSIP Architecture. 76
4.1.3 Implementation Results 78
4.1.4 OSIP Firmware. 79
4.2 Platform Integration. 80
4.2.1 Hardware Integration. 80
4.2.2 Software Integration 80
4.2.3 System Level Modeling. 81
4.3 Benchmarking. 84
4.3.1 Experimental Setup 84
4.3.2 Synthetic Benchmarking 85
4.3.3 H.264 Application. 86
4.4 Synopsis. 87

5 Sequential Code Flow 89
5.1 Tool Flow Overview 89
5.1.1 Clang and LLVM 90
5.1.2 Tool Flow Components 91
5.2 Application Analysis 93
5.2.1 Application Tracing. 93
5.2.2 Building the Application Model 95
5.2.3 Sequential Performance Estimation Revisited. 97
5.2.4 Analysis Results 98

5.3 Partitioning and Parallelism Extraction . . . 99
5.3.1 Graph Clustering . . . 99
5.3.2 Discovering Parallelism Patterns . . . 100
5.3.3 Global Analysis . . . 105
5.4 Backend . . . 108
5.4.1 C Backend . . . 108
5.4.2 CPN Hints . . . 111
5.5 Case Study . . . 113
5.5.1 Synthetic Example . . . 113
5.5.2 Audio Filter Application . . . 116
5.6 Limitations and Outlook . . . 121
5.7 Synopsis . . . 122

6 Parallel Code Flow . . . 123
6.1 Tool Flow Overview . . . 123
6.1.1 The CPN C Compiler . . . 124
6.1.2 Tool Flow Components . . . 125
6.1.3 Input and Output Modeling . . . 126
6.2 Token Logging and KPN Tracing . . . 130
6.2.1 KPN Traces . . . 130
6.2.2 Sequential Performance Estimation Revisited . . . 131
6.3 Building and Evaluating the Model . . . 132
6.3.1 Model Construction . . . 132
6.3.2 Model Evaluation . . . 137
6.4 Best-Effort Mapping and Scheduling . . . 139
6.4.1 Trace Graph . . . 139
6.4.2 Graph-Based Buffer Sizing . . . 141
6.4.3 Heuristics for Setting Scheduler Parameters . . . 142
6.4.4 Heuristics for Decoupled Process and Channel Mapping . . . 144
6.4.5 Joint Process and Channel Mapping: The GBM Algorithm . . . 147
6.4.6 Post-Processing Phase . . . 151
6.5 Mapping and Scheduling with Timing Constraints . . . 151
6.6 OSIP Backend . . . 154
6.6.1 OSIP Generated Code . . . 154
6.6.2 Debugging Environment . . . 155
6.7 Case Study . . . 156
6.7.1 Experimental Setup . . . 156
6.7.2 Best-Effort Results . . . 159
6.7.3 Results for Real-Time Applications . . . 160
6.7.4 Validation on TI's Keystone Platform . . . 161
6.8 Limitations and Outlook . . . 164
6.9 Synopsis . . . 164

7 Extensions for Software Defined Radio . . . 165
7.1 Tool Flow Overview . . . 165
7.1.1 Tool Flow Components . . . 166
7.1.2 Input and Output Modeling . . . 167
7.2 Tracing for Software Defined Radio . . . 169
7.2.1 Flavor Trace Generation . . . 169
7.2.2 Sequential Performance Estimation Revisited . . . 170
7.3 Configuration Matching . . . 172
7.3.1 From Nucleus to Flavors . . . 172
7.3.2 Interface Matching . . . 173
7.4 Mapping and Code Generation . . . 175
7.4.1 Testing Timing Constraints . . . 175
7.4.2 Code Generator Extensions . . . 176
7.5 Case Study . . . 178
7.5.1 Target Application: MIMO OFDM Transceiver . . . 178
7.5.2 Target Platforms and Flavor Libraries . . . 179
7.5.3 Results of the SDR Flow . . . 181
7.5.4 Execution on the Virtual Platform . . . 183
7.6 Limitations and Outlook . . . 184
7.7 Synopsis . . . 185

8 Multi-Application Flow . . . 187
8.1 Tool Flow Overview . . . 187
8.1.1 Tool Flow Components . . . 188
8.1.2 Input and Output Modeling . . . 189
8.2 Scenario Analysis . . . 191
8.2.1 Generating Configurations for Single Applications . . . 192
8.2.2 Pre-Processing . . . 193
8.2.3 Composability Analysis . . . 195
8.2.4 Results Export . . . 198
8.3 Case Study . . . 198
8.3.1 Experimental Setup . . . 199
8.3.2 Scenario Analysis Results . . . 199
8.4 Limitations and Outlook . . . 201
8.5 Synopsis . . . 203

9 Conclusions and Outlook . . . 205

References . . . 209

Glossary . . . 225

Index . . . 229

Figures

Fig. 1.1	SoC Trends: Processor count
Fig. 1.2	SoC Trends: Processing elements in OMAP and Snapdragon
Fig. 1.3	Software complexity trends
Fig. 1.4	Design Gaps
Fig. 1.5	MPSoC programming flow
Fig. 1.6	Sample target MPSoCs
Fig. 1.7	Tool flows overview
Fig. 2.1	Motivational example
Fig. 2.2	Example of mapping and scheduling for a DAG
Fig. 2.3	Phases in compilation flow
Fig. 2.4	Motivational example for the sequential code problem
Fig. 2.5	3AC example
Fig. 2.6	CDFG Example
Fig. 2.7	Forms of parallelism
Fig. 2.8	MoC examples
Fig. 2.9	MoC expressiveness
Fig. 2.10	Example of mapping for a KPN application
Fig. 2.11	The Nucleus waveform development concept
Fig. 4.1	Hierarchical mapping and scheduling
Fig. 4.2	Profiling results for the OSIP application
Fig. 4.3	OSIP Architecture
Fig. 4.4	Generic OSIP-based platform
Fig. 4.5	Example of OSIP tCEFSM model
Fig. 4.6	OSIP's synthetic benchmark results
Fig. 4.7	OSIP operational ranges
Fig. 4.8	OSIP's results for H.264 decoding
Fig. 5.1	Sequential tool flow
Fig. 5.2	Trace file example
Fig. 5.3	Flow for sequential application model
Fig. 5.4	Sample analysis results
Fig. 5.5	Parallelism patterns in a DFG
Fig. 5.6	Efficiency-speedup tradeoff

Fig. 5.7 CPN hints in the MAPS IDE
Fig. 5.8 The Pitaya platform
Fig. 5.9 Synthetic example results
Fig. 5.10 Gantt Charts from the Pitaya simulator
Fig. 5.11 LP-AF profile
Fig. 5.12 Dynamic call graph of the LP-AF application
Fig. 5.13 Parallelism extraction for the LP-AF application
Fig. 6.1 Parallel tool flow
Fig. 6.2 Conceptual view of a schedule descriptor
Fig. 6.3 Process trace example
Fig. 6.4 Platform graph construction
Fig. 6.5 Measurement of timing constraints
Fig. 6.6 Trace Replay Module
Fig. 6.7 Evaluation results for the JPEG application
Fig. 6.8 Flow of the mapping and scheduling phase
Fig. 6.9 Example of a trace graph
Fig. 6.10 OSIP configuration for code generation
Fig. 6.11 OSIP state from debugger
Fig. 6.12 Experimental setup
Fig. 6.13 Sample randomly generated KPN graphs
Fig. 6.14 Mapping results for platform DCP
Fig. 6.15 Real-time mapping results
Fig. 6.16 Real-time iterative mapping
Fig. 6.17 Keystone results
Fig. 7.1 SDR tool flow
Fig. 7.2 Example of flavor traces
Fig. 7.3 Example of time annotation for an irregular nuclei
Fig. 7.4 Code generation flow SDR applications
Fig. 7.5 MIMO OFDM transceiver diagram
Fig. 7.6 SDR target heterogeneous platforms
Fig. 7.7 Gantt charts examples in the SDR flow
Fig. 7.8 Best rates obtained by the SDR flow
Fig. 7.9 Execution traces for TX-RX obtained from VP
Fig. 8.1 Multi-application tool flow
Fig. 8.2 Example of multi-application description
Fig. 8.3 Scenario analysis flow
Fig. 8.4 Illustration of utilization functions
Fig. 8.5 Pre-processing example
Fig. 8.6 Example of combined processor utilization
Fig. 8.7 Multi-application configurations
Fig. 8.8 Composability results

Tables

Table 3.1	Comparison of runtime management approaches
Table 3.2	Comparison of parallelism extraction approaches
Table 3.3	Comparison of parallel code synthesis approaches
Table 3.4	Comparison of SDR approaches
Table 3.5	Comparison of multi-application approaches
Table 4.1	OSIP synthesis results
Table 5.1	Instrumentation functions
Table 6.1	Test results for 800 random KPNs on platform SCP
Table 7.1	Trace generation in the SDR flow
Table 7.2	Configuration matching for buffer/flag interfaces
Table 7.3	Configuration matching for a register interface
Table 7.4	Overview of latency values (in cycles) for all flavors used in the case study
Table 7.5	SDR flow: Results summary
Table 7.6	Comparison of rates
Table 8.1	Results of the scenario analysis phase