

Run-time Adaptation for Reconfigurable Embedded Processors

Lars Bauer • Jörg Henkel

Run-time Adaptation for Reconfigurable Embedded Processors



Springer

Lars Bauer
Karlsruhe Institute of Technology
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany
lars.bauer@kit.edu

Jörg Henkel
Karlsruhe Institute of Technology
Haid-und-Neu-Str. 7
76131 Karlsruhe
Germany

ISBN 978-1-4419-7411-2 e-ISBN 978-1-4419-7412-9
DOI 10.1007/978-1-4419-7412-9
Springer New York Dordrecht Heidelberg London

© Springer Science+Business Media, LLC 2011

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

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

Contents

1	Introduction	1
1.1	Application-Specific Instruction Set Processors	2
1.2	Reconfigurable Processors	3
1.2.1	Summary of Reconfigurable Processors	4
1.3	Contribution of this Monograph	5
1.4	Monograph Outline	6
2	Background and Related Work	9
2.1	Extensible Processors	9
2.2	Reconfigurable Processors	11
2.2.1	Granularity of the Reconfigurable Fabric	11
2.2.2	Using and Partitioning the Reconfigurable Area	17
2.2.3	Coupling Accelerators and the Processor	21
2.2.4	Reconfigurable Instruction Set Processors	23
2.3	Summary of Related Work	26
3	Modular Special Instructions	29
3.1	Problems of State-of-the-Art Monolithic Special Instructions	29
3.2	Hierarchical Special Instruction Composition	34
3.3	Example Special Instructions for the ITU-T H.264 Video Encoder Application	41
3.4	Formal Representation and Combination of Modular Special Instructions	49
3.5	Summary of Modular Special Instructions	53
4	The RISPP Run-Time System	55
4.1	RISPP Architecture Overview	55
4.1.1	Summary of the RISPP Architecture Overview	58
4.2	Requirement Analysis and Overview	58
4.2.1	Summary of the Requirement Analysis and Overview	65

4.3	Online Monitoring and Special Instruction Forecasting	66
4.3.1	Fine-Tuning the Forecast Values.....	69
4.3.2	Evaluation of Forecast Fine-Tuning.....	73
4.3.3	Hardware Implementation for Fine-Tuning the Forecast Values	76
4.3.4	Summary of the Online Monitoring and SI Forecasting.....	79
4.4	Molecule Selection.....	80
4.4.1	Problem Description for Molecule Selection.....	81
4.4.2	Parameter Identification for the Profit Function	84
4.4.3	Heuristic Solution for the Molecule Selection.....	87
4.4.4	Evaluation and Results for the Molecule Selection	90
4.4.5	Summary of the Molecule Selection	96
4.5	Reconfiguration-Sequence Scheduling	97
4.5.1	Problem Description for Reconfiguration-Sequence Scheduling	98
4.5.2	Determining the Molecule Reconfiguration Sequence	101
4.5.3	Evaluation and Results for the Reconfiguration-Sequence Scheduling	106
4.5.4	Summary of the Reconfiguration-Sequence Scheduling	109
4.6	Atom Replacement.....	110
4.6.1	Motivation and Problem Description of State-of-the-Art Replacement Policies	110
4.6.2	The MinDeg Replacement Policy	114
4.6.3	Evaluation and Results.....	117
4.6.4	Summary of the Atom Replacement.....	122
4.7	Summary of the RISPP Run-Time System	123
5	RISPP Architecture Details.....	125
5.1	Special Instructions as Interface Between Hardware and Software.....	126
5.2	Executing Special Instructions Using the Core Instruction Set Architecture	131
5.3	Data Memory Access for Special Instructions.....	136
5.4	Atom Infrastructure.....	139
5.4.1	Atom Containers and Bus Connectors	143
5.4.2	Load/Store- and Address Generation Units	148
5.4.3	Summary of the Atom Infrastructure.....	152
5.5	RISPP Prototype Implementation and Results	152
5.6	Summary of the RISPP Architecture Details.....	163
6	Benchmarks and Comparisons.....	165
6.1	Benchmarking the RISPP Approach for Different Architectural Parameters.....	166

Contents	vii
6.2 Comparing Different Architectures	169
6.2.1 Assumptions and Similarities	170
6.2.2 Dissimilarities	171
6.2.3 Fairness of Comparison	172
6.2.4 Summary of Comparing Different Architectures.....	173
6.3 Comparing RISPP with Application-Specific Instruction Set Processors	174
6.4 Comparing RISPP with Reconfigurable Processors	182
6.5 Summary of Benchmarks and Comparisons.....	187
7 Conclusion and Outlook	191
7.1 Summary	191
7.2 Future Work	193
Appendix A: RISPP Simulation.....	197
Appendix B: RISPP Prototype.....	205
Bibliography	211
Index.....	221

Abbreviations

AC	Atom container: a part of the reconfigurable fabric that can be dynamically reconfigured to contain an atom, i.e. an elementary data path
AGU	Address generation unit
ALU	Arithmetic logic unit
ASF	Avoid software first: a reconfiguration-sequence scheduling algorithm, as presented in Sect. 4.5
ASIC	Application-specific integrated circuit
ASIP	Application-specific instruction set processor
BC	Bus connector: connecting an → AC to the atom infrastructure
BRAM	Block → RAM: an on-chip memory block that is available on Virtex → FPGAs
cISA	core Instruction Set Architecture: the part of the instruction set that is implemented using the (nonreconfigurable) core pipeline; can be used to implement → SIs as well, as presented in Sect. 5.2
CLB	Configurable logic block: part of an → FPGA, contains multiple → LUTs
CPU	Central processing unit
DCT	Discrete cosine transformation: a computational kernel that is used in H-264 video encoder
EEPROM	Electrically erasable programmable read only memory
FB	Forecast block: indicated by an → FI, containing a set of → SIs with a → FV per SI
FI	Forecast instruction: a special → HI that indicates an → FB
FIFO	First-in first-out buffer
FPGA	Field programmable gate array: a reconfigurable device that is composed as an array of → CLBs, → BRAMs, and further components
FPS	Frames per second

FSFR	First select, first reconfigure: a reconfiguration-sequence scheduling algorithm, as presented in Sect. 4.5
FSL	Fast simplex link: a special communication mechanism for a MicroBlaze processor
FSM	Finite state machine
FV	Forecast value: the expected number of → SI executions for the next computational block, part of the information in an → FB
GPP	General purpose processor
GPR	General purpose register file
GUI	Graphical user interface
HEF	Highest efficiency first: reconfiguration sequence-scheduling algorithm, as presented in Sect. 4.5
HI	Helper instruction: an assembly instruction that is dedicated to system support (e.g., an → FI); not part of the → cISA and not an → SI
HT	Hadamard transformation: a computational kernel that is used in H-264 video encoder
IP	Intellectual property
ISA	Instruction set architecture
ISS	Instruction set simulator
KB	Kilo byte (also KByte): 1,024 byte
LSU	Load/store unit
LUT	Look-up table: smallest element in an → FPGA, part of a → CLB; configurable as logic or memory
MB	Mega byte (also MByte): 1,024 → KB
MinDeg	Minimum degradation: an atom replacement algorithm, as presented in Sect. 4.6
MUX	Multiplexer
NOP	No operation: an assembly instruction that does not perform any visible calculation, memory access, or register manipulation
NP	Nondeterministic polynomial: a complexity class that contains all decision problems that can be solved by a nondeterministic Turing machine in polynomial time
OS	Operating system
PCB	Printed circuit board
PRM	Partially reconfigurable module
PSM	Programmable switching matrix

RAM	Random access memory
RFU	Reconfigurable functional unit: denotes a reconfigurable region that can be reconfigured toward an SI implementation
RISPP	Rotating instruction set processing platform
SI	Special instruction
SJF	Shortest job first: a reconfiguration sequence-scheduling algorithm, as presented in Sect. 4.5
SPARC	Scalable processor architecture: processor family from Sun Microsystems; used for the → RISPP prototype
VLC	Variable length coding: a computational kernel that is used in H-264 video encoder
VLCW	Very long control word: configuration bits for the coarse-grained reconfigurable atom infrastructure, i.e. determining the operation mode and connection of the → ACs, → BC, → AGUs, and → LSUs

List of Figures

Fig. 2.1	Connecting coarse-grained reconfigurable functional units	12
Fig. 2.2	Application-specific and domain-specific CCA design examples	13
Fig. 2.3	Connecting LUTs as the basic building blocks of fine-grained reconfigurable logic to slices and configurable logic blocks.....	14
Fig. 2.4	Two-dimensional array of fine-grained reconfigurable CLBs that are connected with fine-grained reconfigurable switching matrices.....	15
Fig. 2.5	Example for a general framework for partially reconfigurable modules that comprise dedicated IP cores or tasks	18
Fig. 2.6	Erlangen slot machine (ESM) architecture overview	18
Fig. 2.7	Different area models for fine-grained reconfigurable fabric.....	20
Fig. 2.8	2D area model using a dynamic network-on-chip architecture to establish communication	21
Fig. 2.9	Coupling reconfigurable accelerators with the core processor.....	22
Fig. 3.1	Comparing different performance vs. reconfiguration overhead trade-offs	32
Fig. 3.2	Hierarchical composition of special instructions: multiple implementation alternatives – so-called molecules – exist per special instruction and demand atoms for realization	35
Fig. 3.3	Example for the modular special instruction SATD (sum of absolute (Hadamard-) transformed differences), showing the details for the Transform atom and the SAV (sum of absolute values) atom.....	36
Fig. 3.4	Example schedule for a molecule of the SATD special instruction, using two instances of each atom	39
Fig. 3.5	H.264 application flow, highlighting the three main computational blocks (ME, EE, LF) and their embedded kernels	43
Fig. 3.6	Relative frequency of intraprediction MacroBlocks (I-MBs) in a video sequence.....	44
Fig. 3.7	Example of a motion compensation special instruction with three different atoms and the internal data path of the point filter atom	46

Fig. 3.8	Special instruction for in-loop de-blocking filter with example schedule and constituting atoms for filtering conditions and filtering operation.....	47
Fig. 3.9	Example for union, intersection, and determinant operation on molecules.....	50
Fig. 3.10	Example for upgrade operation on molecules.....	51
Fig. 3.11	Example for relation, supremum, and infimum of molecules	52
Fig. 4.1	Extending a standard processor pipeline toward RISPP	56
Fig. 4.2	Fix at design/compile time and adapt at run time	59
Fig. 4.3	Overview of the RISPP run-time system.....	62
Fig. 4.4	State-transition diagram of the run-time system	63
Fig. 4.5	Example control-flow graph showing forecasts and the corresponding special instruction executions	67
Fig. 4.6	Execution sequence of forecast and special instructions with the resulting error back propagation and fine-tuning	67
Fig. 4.7	A chain of forecast blocks, showing how the information in a sliding window is used to determine a forecast error that is back propagated	68
Fig. 4.8	A chain of forecast blocks, showing how multiple previous forecast blocks may be updated, depending on parameter λ	72
Fig. 4.9	Parameter evaluation for α and γ for $\lambda = 0$, showing the resulting application run time	74
Fig. 4.10	Evaluation of the forecast value for different values of α , showing the actual and predicted SI execution.....	75
Fig. 4.11	Accumulated absolute forecast error.....	76
Fig. 4.12	Pipelined implementation for fine-tuning the forecasts	77
Fig. 4.13	Different showcase molecules for two special instructions with the corresponding selection possibilities for different numbers of available atom containers	80
Fig. 4.14	Atom sharing, leading to a size of the combined molecule that is smaller than the accumulated size of the two individual molecules.....	83
Fig. 4.15	Comparing the reconfiguration time and the first execution time of a special instruction.....	86
Fig. 4.16	Impact of the profit-function parameters on the application execution time for four atom containers.....	92
Fig. 4.17	Detailed analysis of the application execution time for the three individual computational blocks motion estimation, encoding engine, and in-loop de-blocking filter for four atom containers....	93
Fig. 4.18	Impact of the profit-function parameters on the application execution time for seven atom containers	94
Fig. 4.19	Statistical analysis of greedy selection for different numbers of atom containers	95
Fig. 4.20	Statistical analysis of optimal selection for different numbers of atom containers	96

Fig. 4.21	Different atom schedules with the corresponding molecule availabilities.....	97
Fig. 4.22	Comparing different scheduling methods for two selected molecules of different SIs.....	102
Fig. 4.23	The problem of upgrade “Gaps” for the SJF scheduler.....	103
Fig. 4.24	Comparing the proposed scheduling schemes for different amount of atom containers	107
Fig. 4.25	Detailed analysis of the HEF scheduler for the motion estimation and encoding engine, showing how the SI latencies (<i>lines</i>) and execution frequencies (<i>bars</i>) change over time	109
Fig. 4.26	High-level H.264 video encoder application flow, showing a typical use case and presenting the different replacement decisions of LRU and MRU in detail	110
Fig. 4.27	Examples for atoms and their utilization in SIs, showing different implementation alternatives (i.e., molecules) and their execution latencies.....	113
Fig. 4.28	Comparing the MinDeg replacement policy with state-of-the-art policies for different reconfiguration bandwidths (a–c) and size of the reconfigurable fabric (<i>x</i> -axis)	118
Fig. 4.29	Summarizing the performance improvement of MinDeg in comparison to state-of-the-art replacement policies	119
Fig. 4.30	Detailed replacement analysis for 20 MB/s reconfiguration bandwidth and 15 ACs	120
Fig. 4.31	Algorithm execution time (accumulated number of innermost loop body executions when encoding 10 frames)	121
Fig. 5.1	Using dual-ported BlockRAMs to implement a general-purpose register file (GPR) with one write and four read ports	126
Fig. 5.2	Levels and contexts to reach the cISA implementation of an SI	132
Fig. 5.3	Memory controller, connecting the memory stage of the core pipeline and both 128-bit ports for SIs with the data cache and an on-chip scratchpad memory	137
Fig. 5.4	Overview of the atom infrastructure and its connection to the core pipeline and the memory controller.....	140
Fig. 5.5	Overview of the nonreconfigurable modules within the atom infrastructure	142
Fig. 5.6	Internal composition of a bus connector, showing the connection to its atom container and the neighboring bus connectors	145
Fig. 5.7	Atom infrastructure with three bus connectors, highlighting two example communication patterns for typical atom computation and data copying	146
Fig. 5.8	Area requirements per bus connector for more buses and finer access granularity	147
Fig. 5.9	Latency changes for increasing amount of buses and bus connectors	147

Fig. 5.10	Internal composition of a load/store unit, showing the connection to the memory port and the address generation units	149
Fig. 5.11	Memory streams, described by base address, stride, span, and skip	150
Fig. 5.12	Problems, if one memory stream shall be accessed with both LSUs in parallel.....	151
Fig. 5.13	Overview of the MicroBlaze system that implements the algorithms of the run-time system and controls the reconfigurations	153
Fig. 5.14	Floorplan of the RISPP prototype implementation, showing the placement of the different components on the Xilinx Virtex-4 LX 160 FPGA	156
Fig. 5.15	Execution time analysis of the RISPP run-time system's algorithms that execute on the MicroBlaze part of the RISPP prototype.....	161
Fig. 6.1	Comparing the impact of the core pipeline operating frequency and the atom infrastructure operating frequency	167
Fig. 6.2	Investigating the effect of different data memory connections.....	168
Fig. 6.3	Impact of the reconfiguration bandwidth and the number of atom containers	169
Fig. 6.4	Analyzing the execution time and the resource usage efficiency using different area deployments when processing 140 video frames with an ASIP.....	174
Fig. 6.5	Detailed ASIP utilization variations for six available atoms.....	176
Fig. 6.6	Application execution time and efficiency of resource usage for encoding 140 video frames on ASIP and RISPP	177
Fig. 6.7	Atom utilization for ASIP and RISPP	179
Fig. 6.8	Detailed atom utilization variations for four available atom containers (ACs), used by ASIP and RISPP, respectively.....	180
Fig. 6.9	Detailed atom utilization variations for five available atom containers (ACs), used by ASIP and RISPP, respectively	181
Fig. 6.10	Detailed atom utilization variations for ten available atom containers (ACs), used by ASIP and RISPP, respectively	182
Fig. 6.11	Comparison of RISPP and Molen, showing the execution time of the h.264 video encoder benchmark (encoding 140 frames in CIF resolution, i.e., 352×288) and the speedup	184
Fig. 6.12	Problem and possible solution when addressing reconfigurable hardware as monolithic RFUs	186
Fig. 6.13	Comparing the RISPP approach with Proteus, executing an H.264 video encoder and showing the impact of RFUs vs. atom containers	187

Fig. A.1	Internal composition of the design space exploration tool, showing module interactions	198
Fig. A.2	Overview of the RISPP simulation visualization: RISPPVis	200
Fig. A.3	RISPPVis zooming into SI execution latency changes.....	201
Fig. A.4	RISPPVis SI latency diagram	202
Fig. A.5	RISPPVis design rule check	203
Fig. B.1	Picture of the Avnet Xilinx Virtex-4 LX160 development kit with periphery for SRAM, SDRAM, reconfiguration EEPROM, audio/video module, and (touch screen) LCDs	207
Fig. B.2	Schematic of the four layers for the PCB for EEPROM, USB, audio, touch screen LCD, and general-purpose connectors.....	208
Fig. B.3	Picture of the developed PCB	208

List of Tables

Table 3.1	Overview of different SATD molecule alternatives.....	38
Table 3.2	Overview of implemented SIs and their required atoms	48
Table 3.3	Overview of high-level molecule and special instruction properties	53
Table 4.1	Hardware requirements for monitoring and forecast unit	79
Table 4.2	Speedup due to HEF scheduling.....	108
Table 4.3	Relevant history-based replacement policies, used for evaluating the performance-guided MinDeg policy	115
Table 5.1	SPARC V8 instruction formats.....	128
Table 5.2	SPARC V8 instruction format 2 with RISPP extensions.....	128
Table 5.3	SPARC V8 format 2 used for UNIMP and helper instructions (HIs)	129
Table 5.4	Overview of implemented helper instructions.....	129
Table 5.5	Instruction format for special instructions as part of SPARC V8 instruction format 2.....	130
Table 5.6	Hardware implementation results for the RISPP prototype; all FPGA utilization numbers are relative to the used Xilinx Virtex-4 LX 160 FPGA	157
Table 5.7	Atom implementation results	158
Table 5.8	Individually constraint parts of the critical path in the atom infrastructure and their processing sequence.....	159
Table 6.1	Investigated architectural parameters	166
Table 6.2	Selected atoms for ASIPs for the computational blocks of H.264 video encoder	175
Table 6.3	Summary of comparison of RISPP and ASIP	178
Table 6.4	Speedup compared with Molen, a state-of-the-art reconfigurable processor with monolithic SIs	185
Table B.1	Resources provided by the FPGA of the prototype	206

List of Algorithms

Algorithm 4.1	Pseudo code of a greedy knapsack solver	87
Algorithm 4.2	Pseudo code of the molecule selection.....	89
Algorithm 4.3	The implemented scheduling method “highest efficiency first”	104
Algorithm 4.4	The performance-guided MinDeg replacement policy	116
Algorithm 5.1	Trap handler to implement special instructions with the cISA and the support of the helper instructions	133
Algorithm 5.2	Examples for implicit and explicit cISA execution of SIs.....	134