

Daniel Gregorek

Hardware Enhanced Run-Time Management for Many-Core Processors

Hardware Enhanced Run-Time Management for Many-Core Processors

Vom Fachbereich für Physik und Elektrotechnik
der Universität Bremen

zur Erlangung des akademischen Grades eines
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von
M.Sc. Daniel Gregorek
aus Hannover

Referent: Prof. Dr.-Ing. Alberto García-Ortiz
Korreferent: Prof. habil. Gilles Sassatelli

Eingereicht am: 14. September 2018
Tag des Promotionskolloquiums: 27. November 2018

Berichte aus der Elektrotechnik

Daniel Gregorek

**Hardware Enhanced Run-Time Management
for Many-Core Processors**

D 46 (Diss. Universität Bremen)

Shaker Verlag
Düren 2019

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Zugl.: Bremen, Univ., Diss., 2018

Copyright Shaker Verlag 2019

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-6623-4

ISSN 0945-0718

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren

Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9

Internet: www.shaker.de • e-mail: info@shaker.de

Abstract

Many-core architectures integrate a large number of comparatively small processing cores into a single chip. However, the high degree of parallelism increases the run-time resource management complexity. Additionally, present CMOS technology does not allow anymore to fully utilize the available on-chip resources due to temperature issues. The properties of many-cores and the paradigm of device under-utilization (also called dark silicon) therefore require novel approaches for the run-time resource management.

This thesis investigates the development of hardware enhancements for run-time resource management on homogeneous MIMD many-core processors. The analysis shall find the potential advantages of hardware enhancements compared to software-based resource management.

In the course of this work, existing tools revealed limitations to appropriately evaluate many-core architectures including hardware enhancements for resource management. Thereupon, the novel framework Agamid which is based on SystemC / TLM-2.0 for many-core simulation was implemented. An integrated modeling approach comprising the hardware model, the user application model and the resource management model was chosen for the Agamid framework.

For the hardware enhanced resource management, a scalable and cluster-based system architecture was developed. The resulting architecture (DRACON) utilizes message passing based communication, a dedicated infrastructure and hardware accelerators for resource management. DRACON was implemented as a SystemC model and partly as a prototype at the register-transfer level. The register-transfer model additionally serves for the calibration of the timing model of Agamid.

A comprehensive evaluation for DRACON and reference architectures was performed using the simulation framework Agamid and dynamic task management as a use case. As benchmarks, synthetic models and task graph models of real-world applications were applied. The results reveal the limited scalability of classical architectures for resource management on many-cores. It is therefore necessary to apply cluster-based or moderately distributed architectures for many-core resource management. Further, the results demonstrate a significant performance improvement for dedicated hardware enhancements at a number of hundreds of processing cores.

The contribution of the DRACON architecture enables a high quality of the resource management while management overhead is significantly mitigated. Additionally, contention between user application and resource manager is avoided. Subsequent research approaches evolve for adaptive implementations of hardware enhanced resource management.

Kurzfassung

Many-Core Architekturen integrieren eine große Anzahl von vergleichsweise kleinen Prozessoren auf einem einzelnen Chip. Durch den hohen Grad an Parallelismus steigt jedoch die Komplexität der Ressourcenverwaltung. Weiterhin können mit heutiger CMOS Technologie die vorhandenen on-chip Ressourcen aufgrund von Wärmeentwicklung nicht mehr uneingeschränkt genutzt werden. Die Eigenschaften von Many-Cores und das Phänomen der eingeschränkten Ressourcennutzung (auch Dark Silicon genannt) erfordern daher neue Ansätze auf der Ebene der Ressourcenverwaltung.

Diese Arbeit beschäftigt sich mit der Entwicklung von Hardware-Erweiterungen für die Ressourcenverwaltung in homogenen MIMD Many-Cores. Es soll untersucht werden, ob die Anwendung von Hardware-Erweiterungen einen Vorteil gegenüber einer softwarebasierten Ressourcenverwaltung bietet.

Im Verlaufe der Arbeit wurde deutlich, dass existierende Werkzeuge nicht hinreichend geeignet sind, um eine Bewertung von Many-Cores mit Hardware-Erweiterungen zur Ressourcenverwaltung vorzunehmen. Es wurde daraufhin ein neuer Simulator auf Basis von SystemC / TLM-2.0 entwickelt. Für die entwickelte Simulationsumgebung Agamid wurde ein integrierter Ansatz gewählt, welcher sowohl die Modellierung der Hardware als auch der Benutzeranwendung und der Ressourcenverwaltung beinhaltet.

Für die Ressourcenverwaltung mit Hardware-Erweiterungen wurde eine skalierbare und clusterbasierte Systemarchitektur implementiert. Die entstandene Architektur (DRACON) verwendet ein nachrichtenbasiertes Kommunikationsprotokoll, eine dedizierte Infrastruktur und Hardwarebeschleuniger für die Ressourcenverwaltung. DRACON wurde sowohl in SystemC als auch in Teilen als Prototyp auf der Register-Transfer Ebene implementiert. Darüber hinaus dient das Register-Transfer Modell zur Kalibrierung des Zeitverhaltens von Agamid.

Auf Basis der Simulationsumgebung Agamid wurde eine umfangreiche Bewertung von DRACON und weiteren Referenzarchitekturen vorgenommen. Die Untersuchung wurde anhand des Problems der dynamischen Taskverwaltung durchgeführt. Dabei kamen sowohl synthetische als auch von realen Anwendungen abgeleitete Task-Graphen als Benchmarks zum Einsatz. Die Ergebnisse zeigen, dass klassische Architekturen zur Ressourcenverwaltung im allgemeinen nicht die erforderliche Skalierbarkeit für den Einsatz in Many-Cores aufweisen. Stattdessen werden clusterbasierte,

oder moderat verteilte Systeme zur Ressourcenverwaltung in Many-Cores erforderlich. Die Ergebnisse zeigen weiterhin einen deutlichen Performancegewinn für die DRACON Architektur bei einer Größe des Many-Cores von einigen hundert Prozessoren. Die Hardware-Erweiterungen ermöglichen eine hohe Qualität der Ressourcenverwaltung bei geringem Mehraufwand. Zusätzlich wird eine wechselseitige Beeinträchtigung zwischen Benutzeranwendung und Ressourcenverwaltung vermieden. Weitere Forschungsansätze ergeben sich bezüglich einer adaptiven Implementierung der dedizierten Hardware-Erweiterungen.

Acknowledgements

I would like to thank Alberto Garcia-Ortiz for inviting me to Bremen and giving me the opportunity to work on hardware design for operating systems. Without his feedback and inducement this thesis would not have been realized. Further, I would like to thank Gilles Sassatelli and the committee for the helpful critics and additional perspectives about my work.

Also, I would like to thank Christof Osewold, Yanqiu Huang, Wanli Yu, Jochen Rust and the many other colleagues who accompanied my work days with their suggestions, discussions and encouragement. Likewise, I would like to thank the numerous students who supported me with their efforts and projects.

Most of all, I would like to thank my beloved Anja for her enduring patience and Charlotte for her bracing curiosity.

Contents

1	Introduction	1
1.1	Problem Description	1
1.2	Contribution	2
1.3	Organization	4
2	Fundamentals	5
2.1	Many-Core Computing	5
2.1.1	Nanometer-CMOS	6
2.1.2	Many-Core Architecture	8
2.1.3	Dark Silicon	11
2.1.4	Applications and Concurrency	12
2.2	Run-Time Management	13
2.2.1	Fundamental Concepts	14
2.2.2	Synchronization and Communication	15
2.2.3	Scheduling and Mapping	17
2.2.4	Power Management	20
2.2.5	RTM Classification	22
2.2.6	Many-Core RTM	23
2.3	Hardware Enhanced RTM	26
2.3.1	Hardware Architecture	28
2.3.2	Enhancements for Synchronization	29
2.3.3	Enhancements for Scheduling	32
2.4	Summary	36
3	DRACON Architecture	37
3.1	Introduction	37
3.2	Related Work	39
3.3	RTM Architecture	40
3.3.1	RTM Taxonomy	40
3.3.2	Analytical Timing Model	41
3.4	Hardware Architecture	43
3.4.1	Baseline System	44

3.4.2	Dedicated Infrastructure	45
3.5	Task Management	47
3.5.1	Task Mapping and Scheduling	49
3.5.2	Task Synchronization and Communication	49
3.5.3	Application Start-Up	51
3.5.4	Load Balancing	52
3.5.5	Dedicated Message Protocol	52
3.6	RT-Level Implementation	54
3.7	Gate-Level Analysis	57
3.8	Summary	58
4	Agamid Framework	61
4.1	Many-Core Simulation	62
4.1.1	Simulation Scope	63
4.1.2	Simulation Speed	64
4.2	Related Work	66
4.3	Agamid Approach	68
4.4	Implementation	69
4.4.1	Processing Core	70
4.4.2	Hardware Enhanced RTM	72
4.4.3	Interconnect	75
4.5	Generic Run-Time Manager	78
4.5.1	Generic RTM Master	79
4.5.2	Generic Task Manager	79
4.6	Task Description Language	82
4.7	Task Execution Model	83
4.8	Calibration	83
4.8.1	Baseline System	85
4.8.2	Hardware Accelerators	87
4.9	Summary	91
5	Evaluation	93
5.1	Experimental Setup	93
5.2	Synthetic Benchmarks	95
5.2.1	Scheduling	95
5.2.2	Task Mapping	99
5.2.3	Synchronization	104
5.2.4	Dispatching	107
5.2.5	Online Load Balancing	109

5.2.6	RTM Master	109
5.2.7	Sensitivity to Interconnects	111
5.3	MCSL Benchmarks	115
5.3.1	Single-Program	115
5.3.2	Multi-Program	118
5.3.3	Random Injection	124
5.4	Evaluation of Agamid	128
5.4.1	Simulation Speed	129
5.4.2	Simulation Accuracy	130
6	Conclusions	133
6.1	Contributions	133
6.2	Outlook	135
A	Appendix	137
A.1	DRACON	137
A.1.1	DRACON System Calls	137
A.1.2	DRACON Messages	138
A.1.3	DRACON C-Library	139
A.2	Agamid	141
A.2.1	Agamid Generic RTM Templates	141
A.2.2	Agamid Manual	144
A.3	MCSL Benchmark	145
A.3.1	MCSL Task Graphs	145
A.3.2	Offline Graph Clustering	147
A.3.3	Single-Program	150
A.3.4	Multi-Program	154