Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung
Nr. 92

Robert Wittig

Architectures and Theoretical Models for Shared Scratchpad Memory Systems

VOGT

Dresden 2021

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Bibliographic Information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2021

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „Architectures and Theoretical Models for Shared Scratchpad Memory Systems“ von Robert Wittig überein.

Gesetzt vom Autor

ISBN 978-3-95947-050-6

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

Technische Universität Dresden

Architectures and Theoretical Models for Shared Scratchpad Memory Systems

Robert Klaus Wittig

der Fakultät Elektrotechnik und Informationstechnik der
Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr. Frank Ellinger
Gutachter:	Prof. Dr. Gerhard Fettweis
	Prof. Dr. Luca Benini
Tag der Einreichung:	05. Januar 2021
Tag der Verteidigung:	20. Mai 2021

Robert Klaus Wittig

Architectures and Theoretical Models for Shared Scratchpad Memory Systems

Dissertation, January 05, 2021

Vodafone Chair Mobile Communications Systems
Institut für Nachrichtentechnik
Fakultät Elektrotechnik und Informationstechnik
Technische Universität Dresden
01062 Dresden

Abstract

Computer engineering is advancing rapidly. For 55 years, the performance of integrated circuits has almost doubled every 18 months. Mostly, these advancements were enabled by technological progress. Even the end of frequency scaling could not bring the ever-increasing performance growth to a halt. However, technology burdens, like noticeable leakage currents, have piled up, which shifts the focus towards architectural improvements. Especially the multi-core paradigm has proven its virtue for chip designs over the last decade. While having been introduced in high-performance computing areas, modern technology nodes also enable low-cost, low-power embedded designs to benefit from multiple cores and accelerators. Since the majority of cores depend on memory, which requires a considerable amount of chip area, this common resource needs to be shared efficiently. High-performance cores use shared caches to increase memory utilization. However, many accelerators do not use caches as they need predictable and fast scratchpad memory (SM). But sharing SM entails conflicts, questioning its fast and predictable nature. Hence, the question arises on how to adapt architectures for sharing while retaining SM's advantages.

This thesis presents a novel, shared SM architecture that embraces the idea of a minimal logic path between core and memory, thereby increasing the maximum operating frequency. Because of its additional capabilities, like dynamic address translation and programmable priorities, it is also well suited for heterogeneous platforms that use dynamic scheduling and require predictable behavior. Demonstrating its advantages, we analyze the characteristics of the new architecture and compare it to state-of-the-art approaches. To further mitigate conflicts, we present the conception of access interval prediction (AIP). By predicting memory accesses with a granularity of a single clock cycle, AIP guides the allocation of resources. This method maximizes memory utilization while reducing conflict delays. With the help of various methods inspired by branch prediction, we achieve over $90\,\%$ of accurate predictions and reduce stall cycles significantly. Another key contribution of this thesis is the extension of analytic models to estimate the throughput of shared SM systems. Again, the focus lies on heterogeneous systems with different priorities and access patterns. The results show a promising error reduction, boosting the used models applicability for real design use cases.

Kurzfassung

Seit über 55 Jahren erleben wir einen beispielslosen Siegeszug von integrierten Schaltungen. Fortschreitende technologische Erfolge ermöglichen eine Leistungsverdopplung aller 18 Monate. Selbst das Erreichen einer scheinbaren Taktfrequenz-Barriere konnte einem weiteren Leistungszuwachs nicht entgegenwirken. Zunehmende technologische Herausfoderungen, wie steigende Leckströme, lassen jedoch vermehrt architekturelle Verbesserungen in den Fokus rücken. So erlangten im letzten Jahrzehnt Multi-Rechenkern-Systeme an Popularität und trugen erheblich zu weiteren Leistungssteigerungen bei. Anfänglich vorallem im Hochleistungssektor angesiedelt, ermöglichen neue Technolgien den Einsatz von mehreren Kernen auch für Systeme mit restriktivem Energieverbrauch. Dabei brauchen die meisten Kerne Speicher, welcher einen erheblichen Teil der Chipfläche einnehmen kann und deshalb effizient genutzt werden muss. Hochleistungs-Computer verwenden daher meist geteilte Cache-Architekturen, welche jedoch für spezialisierte Recheneinheiten ungeeignet erscheinen, da sie unvorhersehbare Zugriffszeiten aufweisen. Deshalb kommen oft Scratchpad Architekturen (SAs) zum Einsatz, um schnelle, vorhersehbare Speicherzugriffe zu ermöglichen. Doch wie können SAs für mehrere Kerne verwendet werden, ohne ihre Vorteile zu beeinträchtigen?

Um diese Frage zu beantworten erforscht diese Arbeit Möglichkeiten effizienter, geteilter SAs und präsentiert eine neu entwickelte Speicherarchitektur. Durch die Minimierung der benötigten Schaltungslogik sowie die Einführung von Prioritäten bleiben die Vorteile von SAs erhalten, gleichzeitig können Taktfrequenzen gesteigert und neue Anwendungsfälle realisiert werden. Jedoch entstehen durch die gemeinsame Speichernutzung auch Zugriffskonflikte. Deshalb konzipiert diese Arbeit Access-Interval-Prediction (AIP). AIP ist in der Lage Speicherzugriffe taktgenau vorherzusagen und erlaubt es somit die negativen Auswirkungen von Konflikten abzumildern. Die entwickelten Algorithmen für die Umsetzung von AIP erzielen über $90\,\%$ Vorhersagegenauigkeit. Eine Untersuchung des Implementierungsaufwandes zeigt außerdem, dass AIP auch für integrierte Systeme mit restriktivem Ressourcenverbrauch realisierbar ist. Ein weiterer entscheidener Beitrag dieser Arbeit ist die Erweiterung von analytischen Schätzmodellen, welche für die Entwicklung von SAs Verwendung finden. Mit den erzielten Ergebnissen können erstmals auch Heterogenitäten in einem System, wie beispielsweise verschiedene Zugriffshäufigkeiten, bei der analytischen Betrachtung berücksichtigt werden.

Acknowledgement

An dieser Stelle möchte ich den vielen Menschen danken, die mir in den vergangenen vier Jahren mit Rat und Tat beiseite standen und mein Leben durch ihre Erfahrung und ihren Homor bereichert haben.

An erster Stelle stehen hierbei meine betreuenden Hochschullehrer Gerhard Fettweis und Emil Matus. Beide sind es bis heute nicht müde geworden, sich mit mir in ausschweifende Diskussionen zu begeben. Viele in der Arbeit verwirklichte Ideen wurden auf ihre Anregung hin entwickelt und umgesetzt.

Besonders hervorzuheben sind die Arbeiten meines Studenten Viktor Razilov. Im Laufe seiner Diplomarbeit erarbeitete er viele der im Abschnitt 6.2 vorgestellten Ergebnisse.

Ein spezieller Dank geht auch an meine Freunde und Kollegen am Lehrstuhl. Insbesondere an Sebastian, Lucas, Steffen, Mattis, Philipp und Tom. Nicht selten gehörte das diskussionsreiche Mittagessen und Kaffee trinken zu den Höhepunkten langer Tage. Weiterhin möchte ich Kathrin, Sylvia, Rüdiger und Raffael danken, ohne deren Hilfe in Sachen Organisation und IT meine Promotion um Jahre länger gedauert hätte.

Auch Chris aus dem Projekt *fast semantics* möchte ich danken. Seinem Aufwand verdanke ich mein Gehalt der letzten Jahre.

Besonderer Dank gebührt auch meiner Freundin Sandra. Ohne ihren Beistand und ihre liebenswerten Art wären die vergangenen Jahre zweifelsohne trister gewesen. Nicht wenige sagen, dass sie mich zu einem besseren Menschen gemacht hat.

Und natürlich möchte ich aus ganzem Herzen meiner Familie danken, die mich seit jeher bedingungslos unterstützt hat.

Dresden, Januar 2021 Robert Wittig

Contents

Abstract iii

Kurzfassung v

Acknowledgement vii

Contents ix

1 Introduction 1
- 1.1 Related Work and Contribution . . . 2
- 1.2 Outline . . . 5

2 Basic Concepts 7
- 2.1 System Concepts . . . 7
- 2.2 System Model . . . 9
- 2.3 The Origin of Access Conflicts . . . 10
- 2.4 The Implications of Memory and Data Sharing . . . 11

3 A Fast and Versatile Memory Architecture 15
- 3.1 Requirements . . . 15
- 3.2 State of the Art – Hardware Platforms . . . 17
- 3.3 A Queue-Based Memory Management System . . . 19
  - 3.3.1 The Interconnect . . . 19
  - 3.3.2 The Memory Controller . . . 21
  - 3.3.3 Properties of the QMMU Architecture . . . 23
- 3.4 System Analysis . . . 23
  - 3.4.1 Implementation . . . 24
  - 3.4.2 QMMU Area and Cost Analysis . . . 26
  - 3.4.3 Conflict Evaluation . . . 28
- 3.5 Future Prospects - Parallel Trees . . . 28
- 3.6 Summary . . . 29

4 Access Interval Prediction 31

4.1 Gains and Limits of AIP . . . 31
4.2 State of the Art – Conflict Avoidance and Prediction Methods . . . 34
4.3 Security Concerns . . . 35
4.4 Theory of AIP . . . 36
4.5 Summary . . . 38

5 Basic Predictors **39**
5.1 Access Time Analysis . . . 39
5.1.1 The Processor . . . 39
5.1.2 The MiBench . . . 39
5.2 Basic Prediction Methods . . . 41
5.2.1 Access-Lookahead Buffer . . . 41
5.2.2 Statistical Predictors . . . 45
5.3 Summary . . . 49

6 Advanced Predictors **51**
6.1 Hashed Maximum-a-Posteriori Predictor . . . 51
6.2 TAGE Predictor . . . 53
6.2.1 Prediction by Partial Matching . . . 53
6.2.2 PPM for AIP . . . 56
6.2.3 From PPM to TAGE . . . 56
6.2.4 Implementation . . . 58
6.2.5 Performance and Cost . . . 65
6.3 Predictor Comparison . . . 67
6.4 Summary . . . 68

7 Model Based Throughput Estimation **71**
7.1 System Model . . . 72
7.2 State of the Art – Throughput Estimation . . . 74
7.3 The Occupancy Model . . . 75
7.3.1 Including Non-Uniform Access Probabilities . . . 76
7.3.2 Deriving Service Ratios . . . 76
7.3.3 Including Priorities . . . 77
7.4 The Priority Model . . . 78
7.4.1 Including Non-Uniform Access Probabilities . . . 78
7.4.2 Including Arbitrary Priorities . . . 78
7.5 The Markov Model . . . 79
7.5.1 Including Non-Uniform Access Probabilities . . . 81
7.5.2 Deriving Service Ratios . . . 82
7.5.3 Including Priorities . . . 83

7.6 Evaluation 84
7.6.1 Performance of the Original Models 84
7.6.2 Performance with Non-Uniform Access Probabilities 86
7.6.3 Performance with Processor Priorities 89
7.7 Adaptations for the QMMU-System 89
7.8 Summary 92

8 Design Considerations **93**
8.1 Design Flow 93
8.2 Example Design 94

9 Conclusion **97**

Appendix A Interconnect Layouts **99**

Appendix B CPUSpec Benchmark **101**

Appendix C Hash Generation **103**

List of Abbreviations **105**

List of Symbols **107**

List of Figures **111**

List of Tables **115**

Bibliography **117**

Publications of the Author **125**

1 Introduction

The story of computer engineering is one of unparalleled success. Every 18 months, so it was predicted by Moore in 1965[1], the number of transistors on a chip would double, leading to an exponential growth of performance. At the same time, the so-called Dennard scaling ensured that the power consumption of bespoken chips stayed constant, enabling unforeseen applications and use-cases in areas like healthcare, the Internet-of-Things, and automotive. Of course, improvements in computer architecture, like the single instruction multiple data (SIMD) paradigm, have also fueled the success of microelectronics but to a much lesser degree. However, the future is likely to change. While the transistor count still rises according to Moores Law, its limits can be foreseen as the size of transistor features is approaching the scale of single atoms. Maybe new materials like nano-tubes [Hil+19] or new process techniques like 3D-stacking [Bat+09; Bat+11] will prolong the growth of transistor counts for decades to come. But currently, we have already reached the limits of the Dennard scaling [HP19], increasing the energy consumption of transistors, stalling an increase in frequency, resulting in a significant slowdown for the growth of single-threaded performance. Fig. 1.1 underpins these facts with concrete numbers. Because cooling constraints limit the total power consumption of a chip, only a fraction of the available transistors can be active simultaneously. As a consequence, computer architects imagined new ways to increase computing performance. Today, the multi-core paradigm is predominant, introducing new cores, also called processing elements (PEs), on a chip at the pace of Moores Law. But again, cooling constraints limit the number of cores that can be active at the same time. Everything put together, there is an increasing number of cores on a chip that can not operate simultaneously, where the performance of each one is predicted to be almost constant in the foreseeable future. Drawn to a conclusion, the authors in [HP19] suggest that application-specific instruction processors (ASIPs) might be an answer to this challenge. If only a fraction of cores can operate at a given time, then they have to be highly specialized to further gain performance improvements. While this means that single-chip solutions will become more heterogeneous, all kinds of PEs rely on a common resource: memory.

[1] Orignally Moore predicted that the transistor count would double every year. Later he adjusted his prediction.

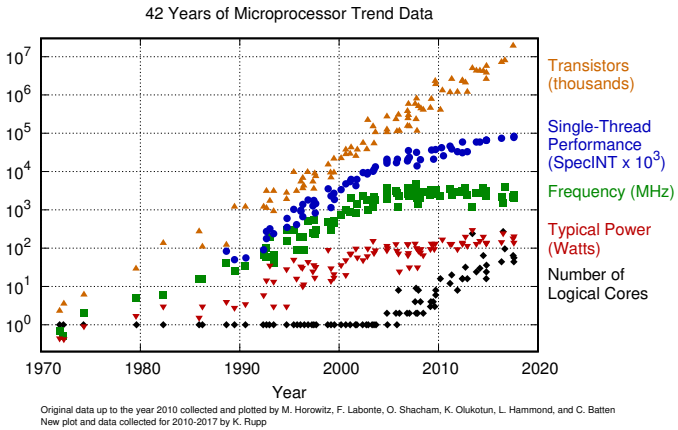

Fig. 1.1. The history of computer engineering in numbers [Rup15].

Memory consumes a considerable amount of on-chip area and contributes extensively to the overall energy consumption [ZG13]. Hence, it is highly desirable to efficiently use this resource by sharing it with as many cores as possible. For desktop and high-performance computing, cores usually share the lower levels of the cache architecture to maximize utilization. In embedded systems, many PEs do not have caches but rely on scratchpad memory (SM) instead. Offering fast and predictable access times, SMs provide a better trade-off for real-time applications that do not have an excessive memory footprint.

1.1 Related Work and Contribution

Ongoing research investigates how SM can efficiently be shared between multiple PEs to gain the same benefits as those exhibited for shared cache architectures. Moreover, the fast and deterministic behavior of SM needs to be preserved. This thesis aims to complement and extend contemporary work, introducing new ideas into the field of shared SM. To provide an overview of the current research, Fig. 1.2 divides the related work into four broad areas.

Architectures encompass the digital circuitry of *interconnects* between memory and PEs, as well as the design of complete *systems*. The dominant roadblock of shared SM is the access delay, because sharing inevitably introduces additional logic between cores and memory. But every additional logic questions the fast access times of

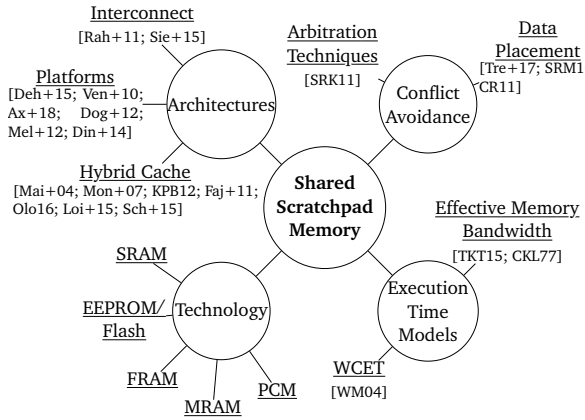

Fig. 1.2. Related research in the area of shared scratchpad memory.

SMs. Important preliminary work in this field was done by [Rah+15], introducing a tree-based interconnect for building a fast, shared memory system. Advancing the idea of a tree-based interconnect, this work contributes the *merging tree* interconnect in combination with *offline arbitration*. Achieving shorter logic paths between PEs and memory, we push the boundaries for access times in systems deploying shared SM. This is done by removing the logic for conflict detection and resolution between PE and memory at the cost of an additional penalty when conflicts occur. Chapter 3 explains the new architecture, named queue-based memory management unit (QMMU), and analyses the involved trade-offs in detail. Offline arbitration also enables address translation and priorities in systems with shared SM. Especially operating systems need this functionality to provide security mechanisms between multiple users. As dynamic scheduling gains momentum, address translation will also be required by specialized hardware accelerators in system-on-chip (SoC) platforms. To demonstrate its potential, we integrate the QMMU into a complete system setup with third party intellectual property cores.

Conflict avoidance plays a crucial role in shared memory systems, where data sharing needs to be performed. Because every conflict entails a penalty for at least one of the contesting PEs, mitigating the delay is crucial for the overall system performance. This is especially true for the newly introduced QMMU, which trades logic against an additional conflict penalty. Contemporary work focuses on optimized data placement in deliberately chosen memory banks to avoid conflicts as much as possible [Tre+17]. This technique finds its limits when data needs to be accessed simultaneously by

multiple PEs, e.g. while working on the same data set. Others [SRK11] focus on dynamic arbitration techniques, but the effect can only take place once a conflict has already occurred. If arbitration is applied statically, cycles might be wasted in situations where one PE is arbitrated but does not access the memory. To address these shortcomings, Chapter 4 introduces AIP. By predicting the interval between consecutive memory transactions, AIP can guide the arbitration before a conflict occurs. To quantify the impact of AIP, we conduct simulation for different systems and configurations. In addition, we develop predictors, i.e. methods to realize AIP in hardware, and analyze their performance. Chapter 6 expands on the idea and presents advanced techniques inspired by branch-prediction, which is used in computer science to predict the outcome of conditional deviations in a program flow.

Execution time models abstract a system with mathematical equations to provide an estimation of the expected system throughput. Thus, designers can use the output of such a model to make decisions about applicable configurations, e.g. the number of PEs and memory banks. However, existing models [TKT15; CKL77] mainly focus on homogeneous systems while we expect future systems to be more heterogeneous due to the explained technology restrictions. Extending existing models in Chapter 7, we show that such heterogeneity can be included in contemporary work. With this, we can estimate not only the overall system throughput but also the individual throughput experienced by a single PE within a complete platform. Furthermore, we explore ways to adapt these models to incorporate the specifics of the new QMMU system.

Technology deals with the physical implementation of memory on-chip. Most commonly, static random access memory (SRAM) is used, but it cannot hold information without a power supply. Persistent storage technologies like Flash do not suffer from this shortcoming but usually have longer access delays. Hence, designers investigate new ways like magnetic random access memory (MRAM) [Eng+05] or phase changing memory (PCM) [Rao+14] to bring together the mentioned advantages while keeping the drawbacks at a minimum. However, this work explicitly deals not with the technologies used for memory implementation but focuses on the other three dimensions.

This section only provided a glimpse into each research area. For more extensive insights, please refer to the corresponding related work sections in the upcoming chapters.

1.2 Outline

The rest of this thesis is structured as follows.

The upcoming *Chapter 2* introduces basic concepts that are relevant throughout the thesis. The chapter expresses the notion of conflicts in shared memory systems, how they arise, and the impact they cause. *Chapter 3* develops a new memory system that is capable of address translation and can transparently handle different word widths. We elaborate on the advantages and drawbacks of the new system and integrate it into a complete platform. *Chapter 4* introduces AIP, a new technique to compensate for conflict penalties. We discuss the gains and limitations of AIP and derive the mathematical theory. *Chapter 5* develops various methods to enable AIP in hardware. *Chapter 6* extends AIP with ideas known from branch prediction to improve the performance of the predictors while reducing the implementational overhead. *Chapter 7* deals with the theoretical background of throughput estimation in shared memory systems. Existing models are evaluated and extended to incorporate more meaningful estimation results for heterogeneous platforms. *Chapter 8* gives a small example to illustrate the decisions made during the design of a system. At last, *Chapter 9* concludes the thesis with an outlook on future research aspects.