Beiträge aus der Informationstechnik

Mobile Nachrichtenübertragung
Nr. 107

Viktor Razilov

On Memory Systems in Parallel Architectures

VOGT

Dresden 2025

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., 2025

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „On Memory Systems in Parallel Architectures“ von Viktor Razilov überein.

Gesetzt vom Autor

ISBN 978-3-95947-084-1

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

On Memory Systems in Parallel Architectures

Dipl.-Ing.
Viktor Razilov

der Fakultät Elektrotechnik und Informationstechnik der
Technischen Universität Dresden

zur Erlangung des akademischen Grades

Doktoringenieur
(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. Matthias Centner
Gutachter:	Prof. Dr.-Ing. Dr. h. c. Gerhard Fettweis
	Prof. Dr. Andreas Herkersdorf
Mitglied der Kommission:	Prof. Dr.-Ing. Kambiz Jamshidi
Tag der Einreichung:	03.03.2025
Tag der Verteidigung:	23.07.2025

Abstract

Memory is the major bottleneck in contemporary processors as its technology scaling keeps lagging behind logic cells. To address this challenge, computer architects have resorted to various parallel archictectures. These archictectures still contain a memory system that needs to be optimized for parallel processing. This thesis first portrays the shift to parallel architectures and how they address the limitations of sequential single-core processors. Out-of-order processors find instructions that can be executed in parallel at runtime. Vector instruction set architectures encode parallelism in the insruction word itself at compile time. Multi-processor systems-on-a-chip (MPSoCs) execute entire instruction threads in parallel. The latter two are more energy-efficient and preferable where applicable. We demonstrate at the example of communications signal processing for current and next-generation radio access networks the strengths and limitations of vector processors and MPSoCs. Based on the analysis, we argue for a heterogeneous MPSoC as a fabric for flexible and scalable communications signal processing platforms. This MPSoC contains general-purpose processors, vector processors, accelerators, and a shared memory system.

In the second part of the thesis, we optimize the different components of this vision. We first take a look at the shared memory system and the conflicts it entails. We opt for access interval prediction as a means to manage the timing impact of conflict arbitration. Drawing inspiration from branch prediction, we derive, optimize and implement an access interval predictor with an accuracy of 97 % outperforming other implementable predictors found in literature. The next optimization regards the access pattern of vector processors on the main memory. In certain situations, which we analyze thoroughly, it is beneficial to load two vectors in parallel. Dual load can boost the performance of vector processors up to 33 % and we demonstrate a gain of 21 % at an area increase of less than 2 % in a proof of concept. Finally, we take a look inside the vector processor, where we can find a small shared memory system, as well. The vector register file architecture needs to avoid, resolve, and mitigate access conflicts. We examine the conflicts and propose dynamic bank layout and an improved arbitration as an alternative to area-expensive deep operand queues for conflict management at no runtime impact or even a slight improvement.

Zusammenfassung

Der Speicher ist der größte Flaschenhals in derzeitigen Prozessoren, denn die Skalierung der Prozesstechnologie für Speicherzellen hinkt der für Logikzellen hinterher. Um dieses Problem zu bewältigen, greifen Rechnerarchitekten auf verschiedene parallele Architekturen zurück. Diese enthalten weiterhin ein Speichersystem, das für parallele Verarbeitung optimiert werden muss. Diese Arbeit schildert erst die Entwicklung zu parallelen Architekturen und wie diese die Beschränkungen von sequentiellen Einzelkern-Prozessoren adressieren. Out-of-order-Prozessoren ermitteln parallel ausführbare Befehle zur Laufzeit. Vektorbefehlssatzarchitekturen kodieren Parallellität in dem Befehlswort selbst während der Kompilierung. Mehrprozessor-Einchipsysteme (MPSoCs) führen ganze Threads parallel aus. Letztere beide Architekturen sind energieeffizienter und zu bevorzugen, wo sie geeignet sind. Wir demonstrieren anhand des Beispiels von nachrichtechnischer Signalverarbeitung für derzeitige und kommende Funkzugangsnetze die Stärken und Grenzen von Vektorprozessoren und MPSoCs. Aufbauend auf dieser Analyse plädieren wir für einen heterogenen MPSoC als Struktur für flexible und skalierbare nachrichtentechnische Signalverarbeitungsplattformen. Dieser MPSoC enthält Universalprozessoren, Vektorprozessoren, Beschleuniger und ein geteiltes Speichersystem.

Im zweiten Teil der Arbeit optimieren wir die einzelnen Komponenten dieser Vision. Zuerst widmen wir uns dem geteilten Speichersystem und den damit einhergehenden Zugriffskonflikten. Wir nutzen Zugriffsintervallvorhersage zur Einhegung des Einflusses der Konfliktschlichtung auf den kritischen Pfad des Systems. Wir entwerfen, optimieren und implementieren einen Zugriffsintervallprädiktor, der inspiriert von gängigen Sprungprädiktoren ist. Dieser erreicht eine Trefferquote von 97 % und schlägt damit alle derzeitig bekannten implementierbaren Zugriffsintervallprädiktoren. Die nächste Optimierung behandelt das Zugriffsverhalten von Vektorprozessoren auf den Hauptspeicher. In bestimmten Fällen, die wir eingehend analysieren, ist es vorteilhaft, zwei Vektoren parallel zu laden. Duales Laden kann die Geschwindigkeit von Vektorprozessoren bis zu 33 % verbessern und wir demonstrieren einen Zuwachs von 21 % bei einer Zusatzfläche von weniger als 2 %. Am Schluss widmen wir uns der inneren Architektur von Vektorprozessoren, welche ebenfalls ein kleines geteiltes Speichersystem enthalten. Die Architektur des Vektorregisterfiles muss Zugriffskonflikte verhindern, lösen und eindämmen. Wir untersuchen die Konflikte und schlagen dynamische Bank-Layouts und eine verbesserte Schlichtung als eine Alternative zu flächenhungrigen tiefen Operandenwarteschlangen vor, um Konflikte ohne Verlangsamung oder sogar einer leichten Verschnellerung zu behandeln.

Acknowledgement

I am indebted to many people who have supported me on this journey. First and foremost, I would like to thank my supervisor Prof. Gerhard Fettweis for offering me the opportunity to work at the Vodafone Chair and to pursue my PhD. He provided me the freedom and the means to investigate my topics of interest and enriched my research with his guidance. I have learned a lot from his pragmatic and scientifically rigorous view of things. I am grateful to Prof. Andreas Herkersdorf for acting as the second referee for this thesis and for the interesting discussions at CSPW 2023 and 2025. I would like to thank my group leader Dr. Emil Matúš for his experienced advice and for always having an open ear for my ideas and questions. Our fruitful dicussions have inspired many ideas that found their way into this thesis.

During my research, I relied on basic infrastructure of the TU Dresden. Therefore, I would like to thank our secretaries and project coordinators Kathrin Fromke, Sylvia Steppat, Caroline Bauder, Berenike Heinrich-Männchen, Claudia Hoffsky, and Nicole Flechs for taking care of so many organizational matters, Rafael Kozerski and Rüdiger Hartmann for their technical support, and the ZIH for their compute resources.

Discussions with research peers have been a great source of inspiration and therefore I want to express my gratitude to the many scientists I met in research projects and at conferences. Werner Haas, Yichao Zhang, and Sebastian Haas stand out in particular. Of course, I had even more great conversations about various topics with many colleagues, who made my time at the chair a great joy. Special thanks go to (in the approximate order I got to know them) Stephan Zeitz, Friedrich Burmeister, Robert Wittig, Mattis Hasler, Stefan Damjančević, Florian Roth, Florian Gast, Simon Friedrich, Shaown Mojumder, Richard Jacob, Martin Schlüter, Daniel Swist, Luis Godoy, Meik Dörpinghaus, Faizan Qureshi, Javad Gholipour, and Joy Halder. I thank the many students I supervised, especially Pruthvi Raj Venkatesha, Shengchun Yang, Zirao Yang, Juncen Zhong, Ipek Gecin, Zehao Chen, and Junyu Chen. I would like to thank the open-source software and hardware community for building all these great tools and designs—especially Ara—that I could build upon.

My greatest motivator in the last years have been my friends, to whom I want to express my gratitude. Among them, I would like to explicitly mention Clemens Löbner for our mathematical discussions and for showing me a way to derive (6.1) with generating functions. Finally, I say “Спасибо” to my parents, grandparents, and my family for their love and for instilling in me an interest for science and technology. A special thanks goes to my sister Sarah Maria Bergmann for proofreading.

Contents

1. Introduction

1.1. The Memory Challenge

Programmable processors have fueled an age of unprecedented technological development known as the information age or the third industrial revolution. As they can be produced for low cost in high quantitites and programmed for almost any use case, they have come to empower a large number of devices ubiquitous in our daily lifes. Applications include, but are by no means limited to, communications technology, internet of things, industrial and home automation, automotive devices, and consumer devices. A continuous process of optimization keeps expanding the range of applications as programmable processors become cheap, energy-efficient, and performant enough for ever more scenarios. Two main strands constitute this evolution: the downscaling of semiconductor technology and advancements in computer architectures. A principal driver of the latter is the need to address limits encountered by the former.

Figure 1.1 visualizes technological and architectural trends in microprocessor design. Around 2000, Dennard scaling (Dennard et al. 1974) and Moore's law (Moore 1965) began to slow down (Hennessy and Patterson 2019a). Chip designers could no longer simply increase the number of transistors and the clock frequency while keeping the same area (Moore's law) and the same power (Dennard Scaling). They had to find new optimizations on the architecture level and resorted to various parallel architectures: Superscalar processors and very long instruction word (VLIW) processors exploit instruction-level parallelism (ILP). Single instruction, multiple data (SIMD) processors exploit data-level parallelism (DLP). Multi-processor systems on a chip (MPSoCs) exploit thread-level parallelism (TLP). For further gains, computing systems might include dedicated accelerators to address efficiency limitations of programmable processors (Hennessy and Patterson 2019b).

All of these systems are fundamentally built as in Fig. 1.2. A computing unit processes data it gets from the memory unit via the communication unit (Kim and Mutlu 2014). These systems include and interact with memory and hence need to cope with its limitations. In contemporary chip designs, memory is one of the biggest—if not the biggest—consumer of area and energy (Zang and Gordon-Ross 2013; Horowitz 2014) and its bandwidth improvements keep lagging behind the microprocessor's improvement since the early 1990s (Hennessy and Patterson 2019a). Static random-access memory (SRAM) is the technology of choice for low-level caches and other

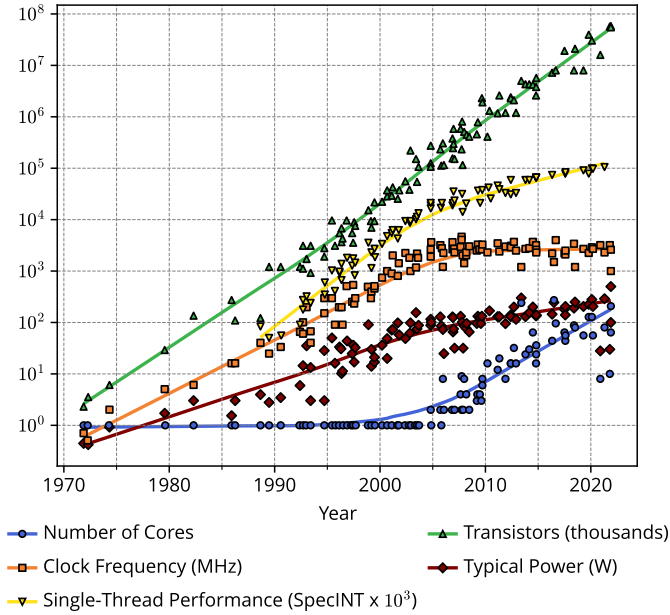

Figure 1.1.: 50 years of microprocessor trend data. Each dot represents the performance indicator of a processor that came out in the respective year. Data from Rupp (2022).

on-chip memories. Especially since the switch to fin field-effect transistors (FinFETs) in newer technology nodes, area scaling of SRAM has struggled to keep up with the scaling of logic cells (Samavedam et al. 2020). In fact, the most recent TSMC 3-nm fabrication technology exhibits almost no shrinking of SRAM cell area, while its logic area went down by 70 % compared to TSMC's 5-nm process (Wu et al. 2022). Confronted with these developments, designers are forced to consider and optimize the interaction between the memory and the core logic, i.e., the memory sytem.

1.2. Related Work and Contribution

Our application of interest is mobile communications—a major pillar of our increasingly digital world (Giordani et al. 2020). As the demand for mobile communication rises, so does the energy consumption of networks (Andrae and Edler 2015; Stobbe et al. 2023). Energy-efficient computing platforms for mobile communications are needed to restrain this worrisome development. We first investigate how parallel

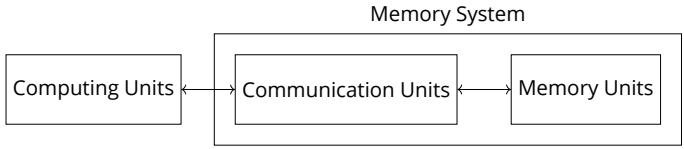

Figure 1.2.: Abstract scheme of a computing system. The computing units process data they get from the memory units via the communication units. The memory system encompasses the communication units and the memory. Adapted from Kim and Mutlu (2014).

processors—MPSoCs and vector processors in particular—enable flexible, yet efficient, communications signal processing. Many communications signal processing specialize the instruction set architecture (ISA) for a small range of algorithms, or even just a particular one (Chen et al. 2021; Shahabuddin et al. 2021; Tourres et al. 2021; Amor et al. 2022). Instead, we target a wider range and gauge the feasibility and limits of such a flexible approach. We propose and analyze architectural optimizations for the memory systems in our platforms of interest on multiple levels—with a particular focus on memory sharing in MPSoCs and vector processors.

Embedded MPSoCs may share a tightly coupled memory (TCM) in the form of a scratchpad memory or a first-level cache. TCMs deliver fast and sometimes even predictable accesses, but sharing degrades these properties. Offline arbitration aided by access interval prediction (AIP) has been suggested for mitigation (Wittig et al. 2019d). Wittig et al. (2019c), Wittig et al. (2019d), Razilov (2020), Wittig (2021), and Friedrich et al. (2023) propose different access interval predictors. Of these, the tagged geometric history length (TAGE) predictor (Razilov 2020)[1] delivers a promising tradeoff between accuracy and area cost. We improve its prediction performance by taking a closer look at the access identification and present and evaluate a register-transfer level (RTL) implementation.

Cray-style vector processors (Russell 1978) leverage SIMD instructions to improve the communication unit. By loading long vectors of data in a pipelined, time-multiplexed manner, vector processors amortize the memory latency (Hennessy and Patterson 2019a). However, the longer execution time of a vector load leads to a longer waiting time in the case of a binary vector operation because it can only fire once the second load has loaded the first elements. As a countermeasure, we propose and analyze dual vector load: to perform the two loads in parallel or interleaved. We examine the feature analytically and state the necessary conditions for performance improvements. Digital signal processors (DSPs) with multiple load-store units (LSUs) have been offering dual load for a long time (Qualcomm 2024; Wilson 2024). It has also been proposed for packed-SIMD processors (Shi 2000). Yet, this work is the first to extend

[1]The cited work is the author's diploma thesis and therefore not considered part of this doctoral thesis. The diploma thesis contains some preliminary work to Chapter 4 but none of the chapter's results (cf. Footnote 1 on Page 45).

this idea to vector processing and to provide a theoretical framework to estimate the performance gain.

Vector processors contain their own memory unit—the vector register file (VRF)—that extends the memory hierarchy with an additional higher, compiler-controlled, level. It acts as the relay over which the functional units (FUs) communicate. An ideal VRF should facilitate the parallel execution of and data exchange between multiple FUs. The high cost of multi-ported memory cells impedes this ideal. It forces designers to either restrict the degree of parallelism or to compose the VRF out of multiple banks and to manage the accompanying access conflicts. There are several studies on the effectiveness of different static bank layouts (Asanović 1998; Asanovic 2019; Cavalcante et al. 2020). We show that these static layouts succeed in certain situations but fail to adapt in others. This analysis leads us to propose dynamic bank layouts to decrease the number of conflicts. In addition, we explore the conflict arbitration in VRFs and make an optimization for mixed-width arithmetics.

1.3. Methodology

We focus on programmable parallel processors. While we are likely to see an increased specialization in future computers (Horowitz 2014; Hennessy and Patterson 2019b), there are still reasons to put programmable processors into account and to improve them. Custom chip development and production is more expensive, time-consuming, and risky and it requires a more specialized expertise than software programming because of the longer iteration cycles. Research activities on making bespoke chip design more accessible include, but are not limited to, high-level synthesis (HLS) (McFarland et al. 1990; Lahti et al. 2019), novel hardware description languages (HDLs) such as Chisel (Bachrach et al. 2012), application-specific instruction set processor (ASIP) design tools (Hu et al. 2021; Hepola et al. 2022), or even HLS-ASIP combinations (Yanık et al. 2023), but much of it is arguably still in its early stages. More importantly, not every application has a market size or efficiency requirements that justify the extra effort. Some might rather benefit from a shorter time-to-market. Furthermore, the additional flexibility might enable energy savings in other places that outweigh the increased compute energy.

The challenge in computer architectures is to build the control logic in a way that makes efficient use of the memory and the data path. Of all possible key performance indicators (KPIs), we therefore primarily target *utilization*, i.e., the ratio of the cycles a given resource is in active use to the overall cycle count. However, this does not mean that we do not consider other KPIs. To the contrary, looking at the utilization gives us an indicator for several important KPIs.

Performance (or speed) is the inverse of the processing time for a given workload and therefore proportional to the utilization for given resources. In most of our studies,

we measure only the cycle count of a single workload occupying the entire computing platform. Throughtput and latency do not differ here; both amount to performance.

Energy efficiency is the inverse of the energy consumed to compute a given workload. Considering information and communication technologies' contribution to rising global energy consumption and climate change (Andrae and Edler 2015), this KPI should be at the forefront of any contemporary processor design research. However, accurate measurement of a computer's energy efficiency requires elaborate physical design. Instead, we rely on heuristics to develop and assess a higher number of different designs for energy efficiency. Since idle resources still consume leakage power, higher utilization is linked to lower energy consumption—especially in low-power, low-frequency designs where static power consumption dominates (Schiavone et al. 2017). In any case, central processor units (CPUs) spend the lion's share of energy in the memory and the control, be it simple in-order processors (Horowitz 2014), application-class CPUs (Zaruba and Benini 2019), or out-of-order (OoO) processors (Kim et al. 2016). We investigate and optimize architectures that reduce the energy of control and memory, i.e., memory sharing and vector processors.

Higher *area* leads to a higher production cost. Therefore one should utilize expensive resources as much as possible. Sometimes, a high utilization may even allow to spend less. Especially the motivation for memory sharing boils down to saving memory area by utilizing it better.

As microprocessors become ever more pervasive, connected, and relied upon, *security* and *trustworthiness* become ever more important (Fettweis and Boche 2022). Despite their importance, we will unfortunately not cover these KPIs in this work. We still note that most security problems revolve around memory systems—around 80 % of common vulnerabilities and exposures (CVEs) are memory safety issues (Miller 2019). They often come as unintended side effects of faulty memory system optimizations, e.g, the famous Spectre (Kocher et al. 2020) or Meltdown (Lipp et al. 2018) exploits. The processor first speculatively executes or loads illegal instructions (Spectre) or memory contents (Meltdown), respectively, to hide memory latency and then discards them once the error is detected. The content still leaves traces that are picked up by the attacker. Examples like these demonstrate the room for future research on the security of memory systems in parallel architectures.

For our research on efficient architectures, we will mostly develop and simulate cycle-accurate software models instead of RTL code. While this method does not give us accurate numbers on the area, timing, and energy of our designs, our research can be reproduced without access to expensive electronic design automation tools and process techologies. The reduced design and simulation time allows us to try out more diverse and novel approaches. Even unrealistic approaches can yield valuable insights into the problems at hand, mark achievability bounds, and inspire more feasible implementations. The aspiration of our scientific venture is not to build a full-fledged market-ready product, but to provide an arsenal of tools that may be used

by designers when applicable. Our proposals are accompanied with an assessment of when they are profitable and when not.

1.4. Outline

The rest of this thesis is structured as follows.

- Chapter 2 contains background information on the developments in processor design this work builds upon. It traces the bottlenecks of programmable processors that motivated the move to parallel architectures. We explain how these exploit ILP, DLP, and TLP and how the memory system of modern processors deal with the limitations of memory technology.
- Chapter 3 studies how parallel architectures can be used for efficient communications signal processing. The architectures under investigation are vector processors and MPSoCs with a stronger emphasis on the former because of the larger body of research on the latter. We make our case for an architecture of a heterogenous MPSoC with general-purpose CPUs, accelerators, and vector processors. The remaining chapters optimize the memory interactions in this system vision.
- Chapter 4 concerns the interactions between the processing elements (PEs) in an MPSoC and the shared TCM. The access conflicts that arise in such a system may be managed by AIP. We present, optimize, and implement on the RTL the tagged geometric history length access interval predictor (TAGE-AIP)—a practical and accurate AIP algorithm, which is inspired by branch prediction (BP).
- Chapter 5 takes a closer look at the interaction between the vector processor's LSU and the MPSoC's main memory. The chapter makes the case for dual vector load—a parallel load of two input vectors. We analyze this proposal and derive statements on conditions for performance improvements.
- Chapter 6 takes a deep dive into the interactions and the conflict management in the VRF, which can be regarded as a small-scale shared TCM system. We dwell on the runtime degradation of conflicts and their characteristics. Based on the study, we suggest dynamic bank layouts and an improved arbitration scheme.
- Chapter 7 gives a summary and an outlook of possible continuations of this work's research.