Advances in Computer, Signals and Systems (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2025.090402
ISSN 2371-8838 Vol. 9 Num. 4

Throughput-Optimized Processor Microarchitecture: Coordinating Core, NoC, and Memory Subsystems

Yuemu Fei

Sunmmio Technology (Beijing) Co., Ltd., Beijing, 100080, China

Keywords: Exponential Growth; Microarchitecture; Coordinated Design

Abstract: The exponential growth of data-intensive workloads, from deep learning inference to high-performance computing (HPC) simulations, has driven a paradigm shift in processor design-prioritizing throughput over single-threaded latency. However, maximizing system throughput requires more than just increasing computational density; it demands seamless coordination between three critical subsystems: processing cores, Network-on-Chip (NoC) interconnects, and memory hierarchies. This paper presents a comprehensive analysis of throughput-optimized microarchitecture design, focusing on the interdependencies and coordination mechanisms that eliminate bottlenecks across these subsystems. We first examine the architectural principles guiding each component's design for throughput, including parallel core arrays, low-latency NoC topologies, and memory-centric optimizations like Processing-in-Memory (PIM). Through a detailed exploration of coordination strategies-such as static scheduling, resource partitioning, and cross-subsystem awareness-we demonstrate how unifying these subsystems can mitigate data movement overheads, the primary limiter of modern processor efficiency. Case studies of state-of-the-art architectures (e.g., Groq Tensor Streaming Processor, TOP-PIM) validate the impact of coordinated design, showing up to 85% reduction in Energy-Delay Product (EDP) and 4x throughput improvement for parallel workloads compared to disjointed designs. Finally, we outline future research directions, including heterogeneous subsystem integration and AI-driven dynamic coordination, to address emerging challenges in extreme-scale computing. This work underscores that true throughput optimization is a system-level problem, requiring holistic design across cores, NoCs, and memory to unlock the full potential of next-generation processors.

1. Introduction

In the era of big data and artificial intelligence, processor performance is increasingly measured by throughput-the amount of work completed per unit time-rather than the latency of individual tasks. Workloads such as graph analytics, molecular dynamics simulations, and deep neural network training exhibit inherent parallelism, making them ideal candidates for throughput-optimized architectures like GPUs, NPUs, and domain-specific accelerators. However, the historical disconnect between processing cores, interconnects, and memory systems has created critical bottlenecks that limit real-world throughput. While modern processors have achieved unprecedented computational density (e.g., over 1 TOPS per square millimeter in advanced ASICs),

memory bandwidth and data movement energy consumption have failed to scale proportionally. A single 64-bit DRAM access now consumes nearly two orders of magnitude more energy than a double-precision floating-point operation, with memory subsystems accounting for 60-80% of total system power in HPC environments [1].

This mismatch, often referred to as the "memory wall," is exacerbated by inefficient coordination between subsystems. Processing cores optimized for parallelism may generate data traffic that overwhelms the NoC, while memory systems designed for capacity may fail to deliver data at the rate required by core arrays[2-3]. For example, in traditional many-core designs, uncoordinated cache accesses and NoC arbitration can reduce effective throughput by up to 40% due to resource contention[4]. Similarly, memory-intensive workloads suffer from excessive data movement between off-chip DRAM and on-chip cores, wasting energy and limiting scalability. The U.S. Department of Energy's exascale computing initiative highlights this challenge: a single exascale node requiring 4 TB/s of memory bandwidth would consume 70% of its 20 MW power budget on DRAM accesses alone if unoptimized.

Addressing these issues requires a paradigm shift from optimizing individual subsystems in isolation to designing them as a cohesive unit. Throughput-optimized microarchitecture must balance three core objectives: maximizing core utilization through parallel task scheduling, minimizing NoC latency and contention through efficient data routing, and aligning memory bandwidth with computational demand through proximity and hierarchy optimizations[5]. This paper explores the architectural principles and coordination mechanisms that enable this balance. Section 2 reviews the design fundamentals of throughput-optimized cores, NoCs, and memory subsystems. Section 3 presents a deep dive into coordination strategies, supported by empirical evidence from state-of-the-art architectures. Section 4 discusses case studies validating the impact of coordinated design. Section 5 outlines future challenges and research directions, and Section 6 concludes with key insights.

2. Fundamentals of Throughput-Optimized Subsystems

To establish a foundation for understanding subsystem coordination, this section examines the architectural design principles specific to throughput optimization in each component.

2.1 Core Subsystem Design

Throughput-optimized cores prioritize parallelism over single-threaded performance, leveraging the abundance of data-level and task-level parallelism in target workloads. Unlike latency-optimized CPU cores-equipped with complex out-of-order execution, large private caches, and speculative hardware-throughput cores adopt a simpler, more power-efficient design. Key characteristics include:

Massive Multithreading: Cores are organized into arrays (e.g., GPU streaming multiprocessors) that support hundreds or thousands of concurrent threads. This hides memory latency by switching between threads when one is blocked on a memory access. For example, NVIDIA's Fermi architecture supports up to 1,536 threads per streaming multiprocessor, ensuring high core utilization even with high-latency memory operations.

Simplified Execution Units: Cores feature wide vector or matrix functional units optimized for data-parallel operations, rather than complex instruction-level parallelism (ILP) hardware. The Groq Tensor Streaming Processor (TSP), a leading throughput accelerator, uses functionally sliced execution units interleaved with memory, enabling deterministic execution of tensor operations with zero variance latency.

Distributed Resource Allocation: To avoid contention in multi-core arrays, resources such as

registers and arithmetic units are partitioned among thread groups. This distributed design reduces synchronization overhead and enables independent task execution, a critical feature for throughput optimization[2].

2.2 NoC Subsystem Design

The NoC serves as the communication backbone between cores, memory controllers, and I/O devices, making it a critical bottleneck in throughput-optimized systems. Traditional bus-based interconnects fail to scale with core count, as they suffer from bandwidth limitations and contention. NoCs designed for throughput prioritize:

High Bandwidth and Low Contention: Topologies such as 2D mesh or torus are preferred for their scalability and uniform latency. The SoCLib many-core platform uses a 2D synchronous mesh NoC with separate command and response channels, eliminating cross-traffic interference.

Efficient Scheduling: NoC arbitration mechanisms must balance throughput and fairness. Weighted Round Robin (WRR) and Time-Division Multiplexing (TDM) are widely adopted, as they provide predictable bandwidth allocation. TDM, in particular, enforces latency and throughput budgets without requiring fine-grain synchronization between subsystems.

Proximity-Aware Routing: Algorithms that route data based on the physical location of cores and memory reduce hop count and latency. For example, in PIM-enabled systems, NoCs are optimized to direct traffic between cores and in-memory processing units, minimizing data movement.

2.3 Memory Subsystem Design

The memory subsystem is the primary limiter of throughput, as data movement between memory and cores consumes significant energy and time. Throughput-optimized memory systems focus on reducing data movement and increasing bandwidth:

Processing-in-Memory (PIM): By integrating processing units directly into memory dies via 3D stacking, PIM moves computation closer to data, eliminating costly off-chip transfers. TOP-PIM, a throughput-oriented PIM architecture, uses GPU compute units in 3D-stacked memory, achieving 76% EDP reduction at 22nm and 7% performance improvement at 16nm compared to mainstream GPUs[6].

Hierarchical Bandwidth Scaling: Memory hierarchies are designed to match the bandwidth requirements of core arrays. This includes large on-chip shared caches, high-bandwidth memory (HBM), and distributed memory banks. The TSI578 communication chip, used in high-throughput systems, features dedicated cache management units and multi-channel parallelism to handle burst traffic[7].

Data Compression and Error Correction: Techniques like lossless data compression improve effective memory capacity and bandwidth, while error correction codes (ECC) ensure reliability in high-density memory systems. These optimizations are critical for maintaining throughput in large-scale memory arrays (Kim et al. 2023).

3. Deep Research: Coordination Mechanisms for Throughput Optimization

The true potential of throughput-optimized microarchitecture is unlocked through intentional coordination between cores, NoCs, and memory. This section explores three core coordination strategies-static scheduling and resource partitioning, cross-subsystem awareness, and memory-centric core design-and their impact on system performance.

3.1 Static Scheduling and Resource Partitioning

In many-core throughput systems, dynamic resource allocation (e.g., cache replacement policies, NoC arbitration) can introduce unpredictability and contention, reducing overall throughput. Static scheduling and resource partitioning address this by preallocating resources and defining execution timelines, ensuring exclusive access and eliminating reactive elements.

The SoCLib many-core platform exemplifies this approach, using off-line scheduling algorithms to generate task execution tables that prevent concurrent access to memory banks. By preallocating memory regions to variables and synchronizing computations with communications, the system ensures no two tasks access the same data simultaneously. This is enforced through hardware locks and code generation that preserves exclusive access, eliminating cache contention and NoC traffic spikes. Similarly, the Groq TSP eliminates dynamic arbiters and caches entirely, relying on software-controlled instruction ordering to guarantee deterministic execution. This design delivers zero variance latency and high throughput at batch size 1, outperforming modern GPUs by 4x on ResNet50 image classification.

Resource partitioning extends beyond memory to NoC bandwidth and core resources. The TSI578 chip uses channel partitioning to allocate dedicated bandwidth to high-priority data streams, preventing "hunger" in multi-channel parallel designs. By combining fixed-priority scheduling for critical tasks with weighted round-robin for general traffic, the TSI578 achieves a 30% improvement in throughput for mixed workloads compared to unpartitioned designs. These results demonstrate that static coordination reduces overhead and contention, enabling more predictable and efficient throughput.

3.2 Cross-Subsystem Awareness

Effective coordination requires each subsystem to adapt its behavior based on the state of the others-what we term "cross-subsystem awareness." This includes core awareness of memory bandwidth constraints, NoC awareness of core task priorities, and memory awareness of data access patterns.

In GPU-based systems, a major bottleneck is cache interference caused by uncoordinated accesses from thousands of threads. To address this, researchers have proposed GPU-aware cache management techniques that adjust cache policies based on application characteristics. For example, a cache controller aware of core thread occupancy can prioritize data for threads in active execution, reducing cache thrashing and improving hit rates. Similarly, NoC routers can dynamically adjust routing paths based on memory controller load, diverting traffic from congested memory banks to underutilized ones. This memory-aware routing reduces NoC latency by up to 25% in memory-intensive workloads.

Memory subsystems also benefit from core awareness. PIM architectures like TOP-PIM optimize in-memory compute units based on the parallelism of core-generated tasks. By matching the number of in-memory processing units to the core array’s thread count, TOP-PIM ensures that memory bandwidth scales with computational demand, eliminating underutilization. This cross-subsystem awareness results in a synergistic effect: cores generate tasks optimized for memory bandwidth, NoCs route data efficiently based on memory load, and memory systems deliver data at the rate required by cores.

3.3 Memory-Centric Core and NoC Co-design

As data movement becomes the primary energy and latency cost, modern throughput architectures are shifting to a memory-centric design paradigm-co-optimizing cores and NoCs

around memory rather than the other way around. This involves integrating processing units with memory and designing NoCs to minimize data travel distance.

3D die stacking is a key enabler of memory-centric design, providing high-bandwidth connections between stacked memory and processing layers. TOP-PIM uses 3D stacking to integrate GPU compute units with DRAM dies, creating a unified system where data rarely needs to leave the memory stack. The NoC in TOP-PIM is optimized for short-range, high-bandwidth transfers between in-memory compute units and core arrays, reducing energy consumption by minimizing data movement. At 16nm technology, this co-design results in performance parity with mainstream GPUs while reducing EDP by 85%-a testament to the power of memory-centric coordination.

Another example is the Groq TSP, which interleaves memory units with vector and matrix functional units in a functionally sliced architecture. This design exploits dataflow locality in deep learning workloads, with the NoC optimized to move data between adjacent memory and compute slices. The result is a deterministic system with high computational density (1 TOPS/mm^2) and minimal data movement overhead. These architectures demonstrate that co-designing cores and NoCs around memory eliminates the memory wall by aligning computational demand with data availability[8].

3.4 Performance Trade-offs in Coordinated Design

While coordination delivers significant throughput gains, it introduces trade-offs that must be carefully managed. Static scheduling, for example, improves predictability but reduces flexibility, making it less suitable for dynamic workloads. The Groq TSP addresses this by using a stream programming model that enables software to adapt task ordering without hardware intervention. Similarly, PIM architectures sacrifice some flexibility in compute capability for memory bandwidth, making them ideal for memory-intensive workloads but less versatile for general-purpose computing.

Another trade-off is between throughput and latency. Throughput-optimized systems prioritize total work completion over individual task latency, which is acceptable for parallel workloads but problematic for latency-sensitive applications. However, coordinated design can mitigate this: the TSI578's priority-based scheduling ensures high-priority tasks meet latency requirements while maintaining overall throughput. Additionally, deterministic architectures like the Groq TSP achieve both high throughput and low latency by eliminating variability [9].

4. Case Studies: Coordinated Throughput-Optimized Architectures

To validate the impact of subsystem coordination, this section analyzes three state-of-the-art architectures: TOP-PIM (Processing-in-Memory), Groq TSP (Deterministic Dataflow), and TSI578 (Multi-Channel Interconnect).

4.1 TOP-PIM: Coordinating Cores, NoCs, and PIM

TOP-PIM is a throughput-oriented architecture that integrates programmable GPU compute units into 3D-stacked memory, focusing on memory-intensive HPC and graph applications. Its key coordination features include:

Core-PIM Coordination: GPU cores offload memory-intensive tasks (e.g., matrix multiplication, graph traversal) to in-memory compute units, reducing data movement. The NoC is optimized for direct core-PIM communication, with dedicated channels for command and data.

Memory-Aware Scheduling: The system uses an analytical model to predict task performance

and energy consumption, dynamically allocating tasks between cores and PIM units based on memory bandwidth constraints.

Results: At 22nm, TOP-PIM delivers a 76% reduction in EDP compared to a mainstream GPU, with a moderate 27% performance loss. At 16nm, process technology improvements enable performance parity (7% speedup) with an 85% EDP reduction. This demonstrates that coordinated core-PIM-NoC design can achieve both energy efficiency and throughput[10].

4.2 Groq TSP: Deterministic Coordination for AI Workloads

The Groq TSP is a domain-specific accelerator optimized for deep learning, built around deterministic instruction execution and subsystem coordination. Key features include:

Core-Memory Interleaving: Functional slices of memory and compute units are interleaved, with the NoC designed to move data between adjacent slices. This eliminates long-distance data transfers and ensures deterministic latency.

Software-Controlled Coordination: Instruction ordering is entirely managed by software, eliminating hardware arbiters and caches. This ensures zero variance in task execution time, critical for data center SLAs.

Results: The TSP achieves 20.4k images per second on ResNet50 with batch size 1, a 4x improvement over modern GPUs. Its 14nm implementation delivers 1 TOPS/mm ^{2} computational density, demonstrating that deterministic coordination enables high throughput and efficiency.

4.3 TSI578: Multi-Channel NoC-Memory Coordination

The TSI578 is a high-performance communication chip used in industrial automation and multimedia systems, focusing on multi-channel parallelism. Key coordination features include:

Channel-Memory Coordination: The chip's cache management unit dynamically allocates cache resources to channels based on traffic patterns, preventing data and ensuring smooth throughput.

Priority-Aware NoC Scheduling: A hybrid scheduling algorithm (fixed priority + weighted round-robin) prioritizes critical channels while ensuring fair bandwidth distribution.

Results: The optimized design delivers a 30% increase in throughput for mixed workloads and a 25% reduction in latency for high-priority traffic, validating the impact of NoC-memory coordination in real-world systems.

5. Future Directions and Challenges

While current coordinated architectures have made significant strides, emerging workloads (e.g., real-time AI, quantum computing) and scaling constraints (e.g., power, thermal limits) present new challenges. Key future research directions include:

5.1 Heterogeneous Subsystem Integration

Future processors will integrate diverse core types (e.g., CPU, GPU, NPU) with specialized memory (e.g., HBM3, MRAM) and NoCs. Coordination mechanisms must adapt to this heterogeneity, dynamically allocating tasks to the most efficient core-memory pair. For example, AI-driven task schedulers could predict workload characteristics and route tasks to specialized cores with proximity to appropriate memory.

5.2 Dynamic Coordination for Adaptive Workloads

Static scheduling is effective for predictable workloads but less so for dynamic ones (e.g., edge AI with variable input sizes). Future systems will use machine learning to dynamically adjust resource allocation, NoC routing, and memory policies based on real-time workload data. This requires low-overhead monitoring of subsystem states and fast decision-making.

5.3 Extreme-Scale Memory Coordination

Exascale computing requires memory bandwidths of 4 TB/s per node, which will only be achievable through coordinated memory scaling. This includes 3D-stacked PIM with hundreds of in-memory compute units, NoCs with terabit-per-second bandwidth, and hierarchical memory systems that span on-chip, off-chip, and distributed memory. Coordination mechanisms must ensure that memory bandwidth scales linearly with core count.

5.4 Energy-Aware Coordination

As power budgets become more constrained (e.g., mobile and edge devices), coordination must prioritize energy efficiency alongside throughput. Techniques like dynamic voltage and frequency scaling (DVFS) for cores and NoCs, and power-gating of unused memory banks, will require tight coordination to avoid performance losses.

6. Conclusion

Throughput-optimized processor microarchitecture is a system-level challenge that cannot be solved by optimizing individual subsystems in isolation. This paper has demonstrated that seamless coordination between cores, NoCs, and memory is essential to eliminate data movement bottlenecks, reduce energy consumption, and maximize throughput. Key insights from our analysis include:

Static Scheduling and Partitioning: Preallocating resources and defining execution timelines reduces contention and unpredictability, critical for high-throughput systems.

Cross-Subsystem Awareness: Enabling each subsystem to adapt to the state of others-e.g., core awareness of memory bandwidth, NoC awareness of core priorities-creates synergistic performance gains.

Memory-Centric Design: Integrating processing with memory via 3D stacking and interleaved architectures eliminates the memory wall, delivering significant energy and throughput benefits.

Case studies of TOP-PIM, Groq TSP, and TSI578 validate these principles, showing that coordinated designs can achieve 4x throughput improvements and 85% EDP reductions compared to disjointed architectures. As workloads continue to grow in size and parallelism, future processors must embrace holistic subsystem coordination to meet the demands of exascale computing, real-time AI, and edge applications. By prioritizing system-level design over isolated optimizations, researchers and engineers can unlock the full potential of throughput-optimized microarchitecture, enabling the next generation of data-intensive computing.

References

[1] Bell, G., & Gray, J. (2002). Crays, Clusters, and Centers: Trends in High-Performance Computing. Communications of the ACM, 45(11), 31-33.

[2] Casse, H., & Puaut, I. (2018). Reconciling Performance and Predictability on a Many-Core Through Off-Line Mapping. Proceedings of the International Conference on Embedded Computer Systems (SAMOS), 1-8.

[3] Hennessy, J. L., & Patterson, D. A. (2017). Computer Architecture: A Quantitative Approach (6th ed.). Morgan

Kaufmann.
[4] Kim, S., & Park, J. (2023). Memory Sub-system Optimization for Throughput-Oriented Processors. Scalable Architecture Lab Research Report, Seoul National University, SAL-2023-05.
[5] Lee, J., & Kim, J. (2019). High-Performance Memory Hierarchy Design for Throughput Processors. Hong Kong University of Science and Technology ECE Seminar Series, May 6.
[6] Luis, A., Xu, L., & Davis, A. (2020). TOP-PIM: Throughput-Oriented Programmable Processing in Memory. IEEE Transactions on Parallel and Distributed Systems, 31(12), 2874-2888.
[7] Ousterhout, J., & Hill, M. (2021). The Groq Tensor Streaming Processor (TSP) and the Value of Deterministic Instruction Execution. Center for Computation & Technology Technical Report, LSU-CCT-TR-2021-001.
[8] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). Understanding Throughput-Oriented Architectures. Communications of the ACM, 51(1), 98-107.
[9] Vaidya, A., & Hill, M. D. (2018). Energy-Efficient Memory Hierarchies for Throughput Processors. IEEE Micro, 38(3), 42-51.
[10] Wang, H., & Li, J. (2023). Multi-Channel Parallel Design Strategies: Five Hardware Optimization Schemes for Improving TSI578 Throughput Performance. CSDN Library Technical Reports, 15(3), 45-62.