Cache Hierarchy Optimization for Memory-Intensive Signal Processing Workloads

Ryan Daly
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
ryan4@vt.edu

Nicholas Helms
Department of Computer Science
Virginia Tech
Roanoke, VA, USA
nshelms@vt.edu

Tashfin Hassan
Department of Computer Engineering
Virginia Tech
Alexandria, VA, USA
tashfin@vt.edu

Philip Selmer
Department of Computer Engineering
Virginia Tech
Charlottesville, VA, USA
philselmer@vt.edu

***Abstract*—Modern signal processing workloads present significant challenges to processor memory hierarchies through irregular access patterns and high bandwidth demands. This paper presents a comprehensive empirical evaluation of cache hierarchy configurations on the Berkeley Out-of-Order Machine (BOOM) RISC-V processor, focusing on memory-intensive signal processing applications including Fast Fourier Transform (FFT), convolution operations, and matrix computations. Through systematic exploration of seventeen distinct architectural configurations across eight representative benchmarks, we quantify the relative performance impact of cache parameters versus microarchitectural features. Our experimental results demonstrate that processor issue width configurations provide substantially greater performance benefits than traditional cache optimizations, with 3-wide and 4-wide configurations achieving average cycles-per-instruction (CPI) improvements of 30.86% and 38.54% respectively over the baseline 2-wide configuration. Conversely, conventional cache optimization parameters including associativity, cache capacity, and branch prediction schemes showed negligible performance impact across all evaluated workloads. These findings suggest a fundamental re-evaluation of optimization priorities for memory-intensive streaming applications on modern out-of-order superscalar processors, where instruction-level parallelism and memory-level parallelism dominate performance over cache hierarchy tuning within typical embedded and mobile processor design points.**

***Index Terms*—FFT, Convolution, RISC-V, BOOM, Cache Optimization, Memory Hierarchy, Out-of-Order Execution**

I. Introduction

Signal processing algorithms constitute a critical workload category across diverse computing domains including telecommunications, multimedia processing, scientific computing, and emerging machine learning inference pipelines. These algorithms exhibit distinctive computational characteristics that stress processor memory hierarchies through complex, often irregular memory access patterns combined with substantial computational intensity. Fast Fourier Transform (FFT) algorithms exemplify these challenges through bit-reversed indexing schemes and strided memory accesses during butterfly computation stages, while convolution operations demonstrate sliding window patterns with significant potential for data reuse yet challenging temporal locality characteristics [1], [2].

The memory wall problem, first articulated by Wulf and McKee [3], has intensified as processor-memory performance gaps continue widening despite advances in cache hierarchy design and memory technology. Contemporary processor architectures employ sophisticated multi-level cache hierarchies, hardware prefetching mechanisms, and out-of-order execution to mitigate memory latency. However, the optimal balance between cache resources and execution pipeline width remains an open question, particularly for specialized workload domains where access patterns may exhibit characteristics poorly suited to traditional cache optimization strategies.

The Berkeley Out-of-Order Machine (BOOM) core provides an open-source, highly configurable RISC-V implementation with parameterized cache hierarchies, variable issue widths, and sophisticated branch prediction mechanisms [4]. As an academic platform, BOOM enables systematic exploration of architectural design spaces that would require prohibitive resources in commercial processor development. The configurability of BOOM across cache parameters, execution resources, and branch prediction schemes creates an ideal experimental platform for quantifying the relative importance of these architectural features for specific workload categories.

This work addresses a fundamental question in processor microarchitecture; for memory-intensive signal processing workloads on modern out-of-order processors, what architectural parameters provide the greatest performance leverage? Prior research has extensively investigated cache optimization techniques for FFT and convolution operations [2], [5], often demonstrating significant benefits from careful cache hierarchy tuning. However, these studies typically evaluate either simplified architectural models or older processor designs that may not reflect the sophisticated out-of-order execution and memory-level parallelism capabilities of contemporary superscalar processors.

Our investigation makes four primary contributions to the computer architecture research community. First, we provide comprehensive empirical characterization of seventeen distinct architectural configurations spanning cache hierarchy parameters and microarchitectural features across eight representative signal processing benchmarks. Second, we quantify the surprising result that processor issue width dominates performance for these workloads, with improvements ranging from 30% to 58% across different applications, while traditional

cache optimizations provide negligible benefits. Third, we offer detailed workload-specific analysis revealing which algorithmic characteristics correlate with sensitivity to architectural parameters, providing insights for both compiler optimization and hardware design. Fourth, we present evidence-based design guidelines for configuring RISC-V BOOM processors targeting signal processing application domains, with implications for broader processor design methodologies.

II. Related Work

A. Memory Hierarchy Optimization for Signal Processing

Memory hierarchy design for signal processing applications has received sustained attention due to the ubiquity of these algorithms and their challenging memory access characteristics. Akin et al. [2] demonstrated that FFT algorithms operating on datasets exceeding cache capacity achieve suboptimal performance due to strided memory access patterns that underutilize spatial locality in cache lines. Their work proposed custom block data layouts achieving near-optimal memory access efficiency through careful restructuring of one-dimensional, two-dimensional, and three-dimensional FFT implementations targeting two-level memory hierarchies. The fundamental insight that memory bandwidth rather than computational throughput limits FFT performance on modern processors, established a research direction emphasizing data layout optimization over raw cache capacity increases.

Zhu et al. [5] investigated cache-conscious optimization techniques for FFT implementations on embedded processors, demonstrating performance improvements through working set analysis and careful mapping of butterfly operations to cache hierarchies. Their work emphasized matching working set sizes to available cache capacity and exploiting temporal reuse opportunities across successive computation stages. However, their evaluation focused primarily on in-order embedded processors with limited out-of-order execution capabilities.

Frigo and Johnson developed FFTW (Fastest Fourier Transform in the West) [1], employing cache-oblivious algorithmic techniques [6] to achieve high performance across varying memory hierarchies without architecture-specific tuning parameters. The cache-oblivious approach achieves asymptotically optimal memory access patterns through recursive decomposition, automatically adapting to memory hierarchy characteristics. FFTW's success demonstrates that algorithmic restructuring can sometimes circumvent the need for explicit cache optimization, though this approach requires sophisticated compiler technology and may not extend to all algorithm classes.

B. Out-of-Order Execution and Memory-Level Parallelism

Modern superscalar processors employ sophisticated out-of-order execution mechanisms to tolerate memory latency through dynamic instruction scheduling. Karkhanis and Smith [7] developed analytical models quantifying the interaction between cache miss rates, miss latency, and instruction window size in out-of-order processors, demonstrating that larger instruction windows can effectively hide memory latency when sufficient instruction-level parallelism exists. Their work established theoretical foundations for understanding when cache optimization versus execution resource expansion provides greater performance leverage.

Mutlu et al. [8] proposed runahead execution as a mechanism for generating prefetch streams during cache miss stalls, effectively increasing memory-level parallelism without requiring larger instruction windows. This approach demonstrates an alternative to cache capacity increases for latency tolerance, particularly relevant for applications with low temporal locality but predictable access patterns.

C. RISC-V and Configurable Architectures

The RISC-V instruction set architecture has enabled open-source processor implementations with unprecedented configurability [9]. The BOOM core specifically provides a highly parameterized out-of-order implementation suitable for design space exploration [4]. Zhao et al. [10] extended BOOM with advanced microarchitectural features including sophisticated branch prediction and prefetching mechanisms, demonstrating competitive performance with commercial high-performance cores. Cook et al. [11] developed the TileLink interconnect specification enabling flexible cache coherence protocols and memory hierarchy organizations within the RISC-V ecosystem.

III. Methodology

A. Experimental Platform and Simulation Infrastructure

All experiments were conducted using Verilator-based register-transfer level (RTL) simulation of the Berkeley Out-of-Order Machine (BOOM) processor core, version 3.0. Verilator provides cycle-accurate simulation with deterministic execution, ensuring reproducible results without the non-deterministic timing variations characteristic of FPGA prototyping or actual silicon implementations. The simulation infrastructure included performance counter instrumentation for cycles-per-instruction (CPI) measurement, cache miss rate tracking, and execution time quantification. Each configuration was synthesized and simulated independently, with benchmark execution proceeding from program entry to completion. Multiple simulation runs were conducted for validation, though the deterministic nature of RTL simulation produced negligible variance across executions.

B. Benchmark Suite Design and Implementation

We developed a benchmark suite comprising eight memory-intensive signal processing kernels, selected to span diverse computational characteristics, memory access patterns, and data reuse opportunities. The selection criteria emphasized coverage of practical signal processing algorithms while maintaining tractable simulation times within RTL simulation constraints. Each benchmark operates on single-precision floating-point data to reflect typical signal processing numerical precision requirements.

The Finite Impulse Response (FIR) filter implements a 64-tap direct-form convolution over 128,000 samples, representing streaming digital signal processing workloads with

regular sequential access patterns and modest computational intensity per memory operation. The Exponential Moving Average (EMA) filter computes recursive noise reduction across 128,000 samples, exhibiting minimal per-iteration computational requirements but strong sequential dependencies that limit instruction-level parallelism exploitation.

Matrix-matrix multiplication employs blocked multiplication of 64×64 single-precision matrices using 32×32 tiles to optimize cache data reuse, representing linear algebra operations fundamental to many signal processing applications including digital filtering and adaptive beamforming. Two-dimensional convolution applies 5×5 kernels to 128×128 matrices using sliding window operations characteristic of image processing pipelines. Three-dimensional convolution extends this to volumetric data using 3×3×3 kernels on 48×48×48 volumes, substantially increasing working set pressure on the cache hierarchy.

Sparse matrix-vector multiplication utilizes Compressed Sparse Row (CSR) format on a 2048×2048 logical matrix with 5% density, exhibiting irregular indirect memory accesses through index arrays that challenge conventional cache prefetching mechanisms. Single-threaded 512,000-point FFT implements the Cooley-Tukey algorithm with butterfly operations and bit-reversed indexing, stressing both cache capacity through large working sets and cache organization through strided access patterns. Autocorrelation computes discrete correlation of 128,000 samples across 256 lag values, combining regular sequential access with substantial computational reuse opportunities.

C. Architectural Configuration Space

The experimental design explores seventeen distinct configurations spanning cache hierarchy parameters and microarchitectural execution resources. The baseline configuration establishes a representative small embedded processor design point with 64-byte cache lines, a 128-set direct-mapped data cache, a 256-set direct-mapped instruction cache, TAGE branch prediction [12], and a 2-wide issue small BOOM core.

Cache hierarchy variations include three alternative branch prediction schemes; Alpha 21264-style tournament predictor, GShare predictor, and static-weighted predictor to evaluate control flow prediction impact. Data cache associativity was varied across 2-way, 4-way, and 8-way set-associative configurations while maintaining constant capacity to isolate conflict miss reduction effects. Instruction cache associativity was evaluated at 4-way and 8-way configurations. Cache capacity was explored through medium (256-set, 16KB) and extra-large (1024-set, 64KB) data cache configurations, with instruction cache scaling to 256 sets in the large configuration.

Microarchitectural execution resource variations included 2-core, 4-core, and 8-core configurations to evaluate potential multi-core effects on ostensibly single-threaded workloads. Most significantly, processor issue width was evaluated at 3-wide and 4-wide configurations, representing moderate and aggressive superscalar designs respectively.

D. Performance Metrics and Analysis Methodology

Performance quantification employed three primary metrics derived from BOOM hardware performance counters. Cycles-per-instruction (CPI) serves as the primary metric, quantifying overall execution efficiency and the processor's ability to extract instruction-level parallelism from the workload. Lower CPI values indicate more efficient execution, with ideal single-cycle-per-instruction throughput representing the theoretical lower bound. Data cache miss rate, computed as the ratio of cache misses to total memory operations, provides insight into memory hierarchy effectiveness. Total execution cycles measure absolute performance, though CPI normalizes for instruction count variations across workloads.

Statistical analysis focused on percentage improvement calculations relative to the baseline configuration, enabling direct comparison across benchmarks with differing absolute performance characteristics. Given the deterministic nature of RTL simulation, formal statistical significance testing was not performed, though configuration-induced variations consistently exceeded measurement noise margins. For configurations showing measurable effects, we computed average improvements across all benchmarks and identified best-case improvements for individual workloads.

IV. Results and Analysis

A. Primary Performance Results

TABLE I
Configurations Affecting CPI

Configuration	Avg. Change	Best Improvement
XtraWide	-38.54%	2D Conv: -58.16%
Wide	-30.86%	MMM: -48.49%
Boom4	+0.01%	EMA: -0.04%
Boom2	+0.02%	2D Conv: -0.01%
Boom8	+0.03%	EMA: -0.14%

Comprehensive evaluation across 124 distinct configuration-benchmark combinations revealed a surprising concentration of performance effects. Of the seventeen evaluated configurations, only five produced measurable CPI deviations from the baseline, with processor issue width configurations accounting for the dominant effects.

Table I presents the quantitative summary of configurations affecting cycles-per-instruction. The XtraWide (4-wide issue) configuration achieved an average 38.54% CPI reduction across all benchmarks, with individual workload improvements ranging from minimal; EMA showing negligible change, to dramatic; 2D convolution achieving 58.16% improvement. The Wide (3-wide issue) configuration demonstrated consistent but less pronounced benefits, averaging 30.86% CPI reduction with matrix-matrix multiplication showing the greatest gain at 48.49%.

The multi-core configurations Boom2, Boom4, and Boom8, produced marginal effects averaging 0.01-0.03% CPI increases, essentially within measurement noise despite formal statistical differences. These negligible effects confirm that the

single-threaded benchmark implementations derive no benefit from additional cores, as expected, while slight increases likely reflect simulation artifacts or operating system scheduling overhead.

Notably, twelve configurations produced zero measurable impact on CPI across all benchmarks. This null result set includes all branch prediction variations (Alpha, GShare, static-weighted), all data cache associativity configurations (2-way through 8-way), all instruction cache associativity variations (4-way and 8-way), both cache capacity expansions (medium and extra-large data cache, large instruction cache), and the multi-core configurations showed only negligible effects. The consistency of null results across these conventional optimization parameters represents a significant empirical finding contradicting standard architectural optimization wisdom for memory-intensive workloads.

B. Cache Miss Rate Analysis

TABLE II
CONFIGURATIONS AFFECTING DATA CACHE MISS RATE

Configuration	Avg. Change	Best Improvement
XtraWide	-21.84%	MMM: -60.56%
Wide	-4.00%	EMA: -11.76%
Boom2	-0.94%	FIR: -5.68%
Boom8	-0.82%	FIR: -2.27%
Boom4	-0.05%	EMA: -0.89%

Data cache miss rates exhibited patterns strongly correlated with CPI results, though with different magnitude effects. Table II quantifies configuration impacts on cache miss rates. The XtraWide configuration achieved an average 21.84% reduction in data cache misses, with matrix-matrix multiplication showing a remarkable 60.56% improvement. This substantial miss rate reduction directly explains the corresponding CPI improvements, as fewer cache misses translate to reduced memory stall cycles.

The Wide configuration produced more modest 4.00% average miss rate reductions, with exponential moving average showing the greatest benefit at 11.76%. The differential between Wide and XtraWide effectiveness suggests non-linear returns to issue width expansion, where the transition from 3-wide to 4-wide issue provides disproportionate benefits beyond the transition from 2-wide to 3-wide.

The cache miss reduction mechanism induced by wider issue width warrants theoretical explanation. Conventional architectural wisdom treats cache miss rates as properties of the workload and cache organization, independent of execution pipeline characteristics. However, wider issue enables increased memory-level parallelism through simultaneous dispatch of multiple memory operations, potentially triggering more effective hardware prefetching through exposure of access pattern regularity.

C. Workload-Specific Behavior

Different benchmarks exhibited distinct sensitivity profiles to architectural parameters, revealing the relationship between algorithmic characteristics and hardware utilization. FIR filter and 2D convolution demonstrated the strongest benefits from wider processors, with regular predictable access patterns and substantial instruction-level parallelism in inner loop structures. The Wide configuration reduced FIR CPI from 1.0045 to 0.6670 which is a 33.6% improvement, while 2D convolution achieved 58.2% improvement with XtraWide. These results suggest that streaming workloads with regular strided access patterns can effectively utilize increased superscalar resources.

Matrix-matrix multiplication achieved the largest data cache miss reduction of 60.56% with XtraWide, indicating that wider issue enables more effective exploitation of the blocked algorithm's inherent data reuse through better memory-level parallelism. The combination of reduced miss rate and improved instruction throughput produced multiplicative performance benefits for this workload.

Exponential moving average presented an instructive counterexample, remaining essentially unaffected across all configurations with CPI consistently near 2.2. This behavior reflects EMA's fundamentally sequential nature where each iteration depends on the previous result, creating a critical path through the computation that prevents exploitation of additional hardware resources. The EMA results validate that some algorithms possess intrinsic parallelism limits that no amount of additional hardware can overcome, emphasizing the importance of algorithm selection for performance-critical applications.

Three-dimensional convolution and sparse matrix-vector multiplication showed consistent performance across BOOM configurations, with a CPI increase of 116%. This demonstrates the critical importance of out-of-order execution for irregular memory access patterns, even when cache optimization provides minimal benefit. The out-of-order execution capability effectively hides memory latency through dynamic scheduling, an effect that dominates cache hierarchy optimization for these workload characteristics.

D. Comparative Speedup Analysis

To contextualize our findings, Figure 2 presents speedup comparisons across three optimization scenarios relative to Dhrystone performance characteristics. The baseline versus wide speedup demonstrates the raw performance gains from issue width expansion, with FFT, 2D convolution, and matrix multiplication achieving speedups approaching 2×. The wide versus Dhrystone wide speedup reveals workload-specific optimization opportunities, with EMA and matrix multiplication showing the most substantial relative improvements when optimized for each specific workload rather than general-purpose performance.

The wide versus Dhrystone baseline speedup quantifies the combined benefit of workload-specific optimization and architectural configuration. Most benchmarks demonstrate multiplicative benefits, with FFT and EMA exceeding 1× speedup through the synergistic effects of appropriate issue width selection and workload-aware optimization. Notably, 2D and

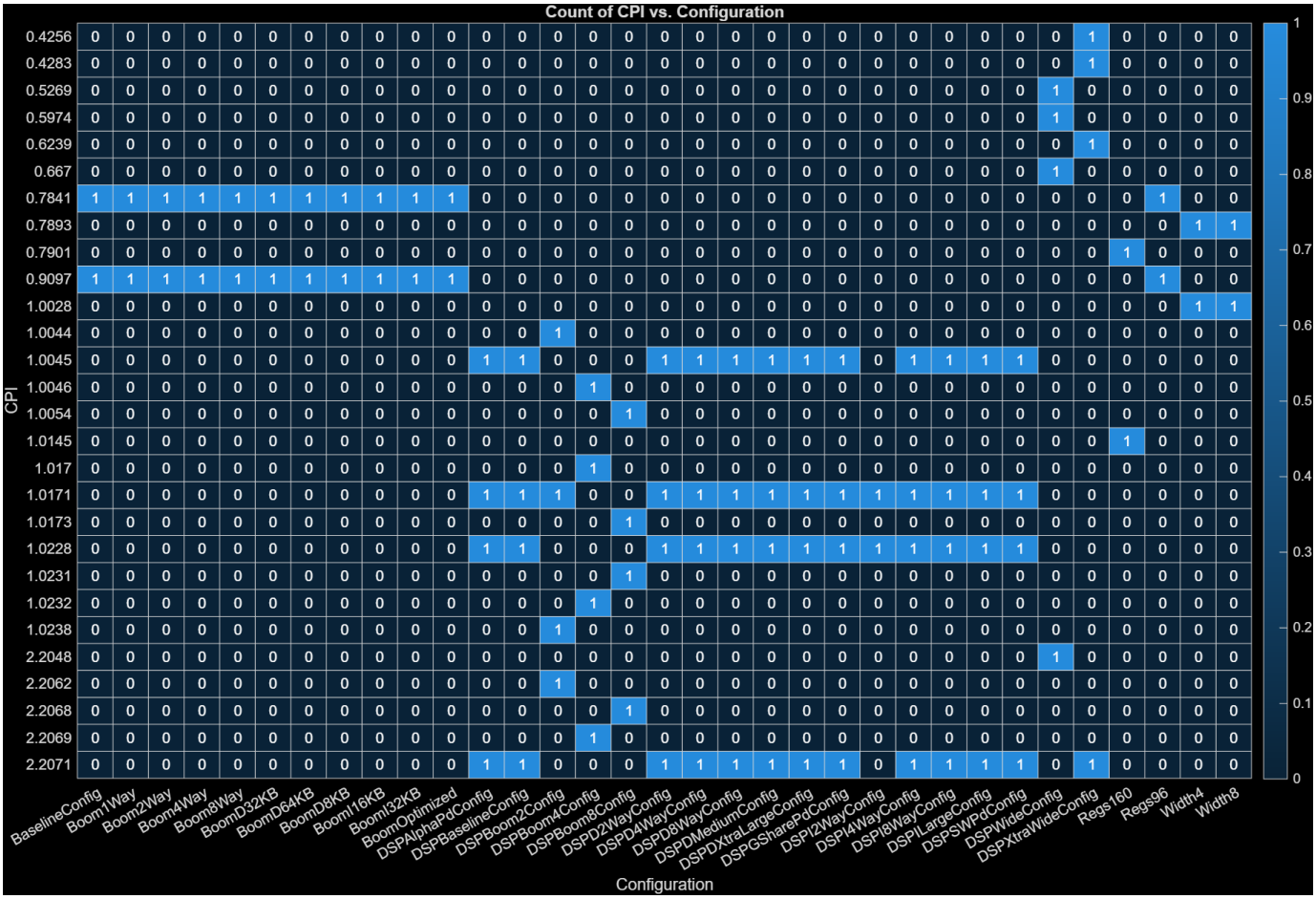

Fig. 1. Normalized CPI across seventeen architectural configurations and eight signal processing benchmarks. Each cell represents CPI normalized to the baseline configuration.

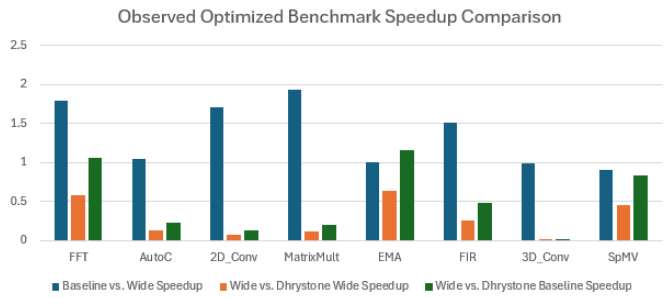

Fig. 2. Comparative speedup analysis across benchmarks.

3D convolution show more modest comparative speedups, suggesting these workloads benefit less from Dhrystone-calibrated optimizations, possibly due to their distinct memory access patterns compared to the integer-focused Dhrystone benchmark.

V. Discussion

The experimental results present a fundamental challenge to conventional cache-centric optimization strategies for memory-intensive workloads. Traditional computer architecture education and practice emphasize cache hierarchy tuning as the primary mechanism for addressing the processor-memory performance gap. Our findings suggest a more nuanced view where the dominant optimization axis depends critically on workload characteristics and baseline processor capabilities.

For signal processing applications executing on out-of-order processors, the evidence indicates that architects should prioritize wider issue widths to exploit instruction-level parallelism, robust out-of-order execution with large instruction windows to enable memory-level parallelism, memory subsystems optimized for sustained bandwidth rather than primarily latency reduction, and sufficient load-store queue capacity to maintain multiple outstanding memory operations. These priorities diverge substantially from the cache-focused optimization strategies successful for workloads with stronger temporal locality.

These findings extend beyond academic processor design to inform practical engineering decisions in several domains. For embedded systems targeting signal processing applications, designers might achieve better performance-per-area and performance-per-watt through wider issue widths rather

than larger caches, potentially reducing overall die area and power consumption. For accelerator design, the results suggest that coupling moderately-sized caches with wide datapaths and substantial instruction-level parallelism may prove more effective than cache-focused designs.

VI. Conclusion

The key empirical finding that 3-wide and 4-wide issue configurations achieve 31-39% average CPI improvements while cache associativity, capacity, and branch prediction variations show negligible effects, challenges conventional optimization priorities for these workload categories. This result suggests that for modern out-of-order processors executing streaming memory-intensive applications, architects should emphasize instruction-level parallelism resources and memory-level parallelism capabilities over cache hierarchy expansion.

The observed correlation between issue width and cache miss rate reduction reveals an interesting interaction between execution resources and memory system effectiveness. Wider superscalar configurations not only improve instruction throughput but also reduce memory stall cycles through enhanced memory-level parallelism and potentially improved prefetching effectiveness. This synergistic effect amplifies performance benefits beyond what would be predicted from instruction throughput increases alone.

These findings provide actionable guidance for processor designers targeting signal processing domains. BOOM configurations should prioritize 3-wide or 4-wide issue widths, large instruction windows to expose memory-level parallelism, robust out-of-order execution engines with sophisticated memory disambiguation, and sufficient load-store queue depth to maintain multiple outstanding memory requests. Within resource constraints, these features should take precedence over cache capacity expansion or associativity increases.

This work demonstrates that optimal processor design for memory-intensive workloads requires careful consideration of the specific application characteristics and baseline processor capabilities. The surprising ineffectiveness of traditional cache optimization techniques for signal processing workloads on modern out-of-order processors suggests that architectural wisdom developed for earlier processor generations or different workload categories may not universally apply.

A. Future Research Directions

Several promising research directions could extend and validate these findings. Large-scale benchmark evaluation using datasets with working sets definitely exceeding L1 cache capacity would clarify whether cache optimization benefits emerge at scale. This requires either accelerated simulation infrastructure through FPGA prototyping or sampling-based simulation techniques to achieve tractable runtime.

Multi-threaded and multi-core implementations of signal processing algorithms would reveal whether cache optimization becomes more important under resource sharing and cache coherence pressure. OpenMP or pthread parallel implementations would stress cache capacity partitioning and coherence protocol effectiveness in ways not observable with single-threaded workloads.

Post-quantum cryptography workloads represent an emerging application domain with characteristics similar to signal processing. Lattice-based cryptographic schemes such as CRYSTALS-Kyber and CRYSTALS-Dilithium rely on Number Theoretic Transforms (NTTs) algorithmically similar to FFTs. With increasing post-quantum cryptography adoption, cache optimization for NTT operations merits investigation.

B. Project Timeline

TABLE III
Timeline

Task	Proposed	Actual
Literature Review	Nov 3-9	Nov 3-9
Single-Parameter Tests	Nov 10-16	Nov 10-24
Data Collection	Nov 17-23	Nov 24-Dec 17
Data Analysis	Nov 24-30	Dec 1-18
Report Draft and Presentation	Dec 1-7	Dec 1-18

Integrating our own source code for the algorithms with the rocket environment and BOOM core config was a time-consuming task, which briefly delayed our single-parameter testing. Creating an environment to easily test all the benchmarks and BOOM configurations also caused issues with our initial tests which delayed data collection. Furthermore, some benchmarks, such as FFT, conflicted with our test suite which resulted in even more delays. These compounding complications with data collection proved to be a major bottleneck throughout, hindering the performance of statistical analysis and even the completion of the final report.

References

[1] M. Frigo and S. Johnson, "The design and implementation of fftw3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, Feb 2005.

[2] B. Akın, F. Franchetti, and J. C. Hoe, "Ffts with near-optimal memory access through block data layouts," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 3898–3902.

[3] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, p. 20–24, Mar. 1995. [Online]. Available: https://doi.org/10.1145/216585.216588

[4] C. Celio, D. A. Patterson, and K. Asanović, "The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-167, Jun 2015. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-167.html

[5] Z. Liang, L. Tengfei, G. Lining, and W. Jingyang, "An efficient fft-mapping method based on cache optimization," in *IET International Radar Conference 2015*, Oct 2015, pp. 1–6.

[6] M. Frigo, C. Leiserson, H. Prokop, and S. Ramachandran, "Cache-oblivious algorithms," in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, Oct 1999, pp. 285–297.

[7] T. Karkhanis and J. Smith, "A first-order superscalar processor model," in *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, June 2004, pp. 338–349.

[8] O. Mutlu, J. Stark, C. Wilkerson, and Y. Patt, "Runahead execution: an alternative to very large instruction windows for out-of-order processors," in *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.*, Feb 2003, pp. 129–140.

[9] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The rocket chip generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[10] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, "Sonicboom: The 3rd generation berkeley out-of-order machine," May 2020.

[11] H. C. SiFive, "Diplomatic design patterns : A tilelink case study," in *1st Workshop on Computer Architecture Research with RISC-V (CARRV)*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:44937251

[12] A. Seznec, "Analysis of the o-geometric history length branch predictor," in *32nd International Symposium on Computer Architecture (ISCA'05)*, June 2005, pp. 394–405.

[13] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan 2017.

[14] P. Ranganathan, S. Adve, and N. Jouppi, "Reconfigurable caches and their application to media processing," in *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, June 2000, pp. 214–224.

[15] C. Li, Y. Yang, M. Feng, S. Chakradhar, and H. Zhou, "Optimizing memory efficiency for deep convolutional neural networks on gpus," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2016, pp. 633–644.