Research Journal of Computer and Information Technology Sciences _____ISSN 2320 – 6527
Vol. **14(1),** 11-17, June **(2026)** *Res. J. Computer and IT Sci.*

Review Paper

Predictive Cache Coherence: Current Techniques and Future Trends

Mukesh Kashyap[1*], Sanjay Kumar[1] and Swati Jain[2]
[1]S.o.S. in Computer Science & IT, Pt. Ravishankar Shukla University, Raipur, Chhattisgarh, India
[2]Govt. J. Yoganandam Chhattisgarh College, Raipur, Chhattisgarh, India
kashyapmukesh86@gmail.com

Available online at: www.isca.in
Received 30th November 2025, revised 2nd March 2026, accepted 30th April 2026

Abstract

The rapid evolution of multicore, manycore, and heterogeneous processor architectures has intensified the challenges of maintaining efficient cache coherence. Traditional reactive protocols, such as MESI and MOESI, struggle to scale due to increased latency, interconnect congestion, and energy overhead. Predictive Cache Coherence (PCC) has emerged as a promising paradigm that anticipates memory access patterns and proactively manages cache state transitions, thereby reducing coherence miss penalties and optimizing system performance. This paper provides a comprehensive review of PCC techniques, including history based, heuristic driven, machine learning based, hybrid, and speculative approaches. We analyse their methodologies, performance trade-offs, hardware overhead, and scalability across diverse architectures, highlighting the comparative advantages and limitations of each strategy. Furthermore, we identify key research gaps, including prediction accuracy under dynamic workloads, hardware constraints, standardized evaluation frameworks, security considerations, and challenges in integrating PCC into emerging system architectures such as chiplets and disaggregated memory. Finally, we outline future research directions aimed at developing lightweight, adaptive, and secure predictive coherence mechanisms that can sustain high performance and energy efficiency in next generation computing systems.

Keywords: Cache coherence; predictive cache coherence; multiprocessor systems; heterogeneous architectures.

Introduction

The increasing integration of multicore and manycore processing units has made efficient cache coherence a fundamental challenge for modern shared memory systems. As core counts scale beyond dozens and into the hundreds, the communication overhead, latency, and energy cost associated with traditional coherence protocols become major performance bottlenecks. Foundational analyses in computer architecture demonstrate that coherence traffic already contributes significantly to on-chip energy consumption and interconnect pressure, especially in data sharing workloads[1,2]. Traditional directory and snoop-based protocols[3] such as MESI and MOESI are therefore increasingly inadequate for emerging high core count systems.

Recent architectural directions including heterogeneous computing, chiplet based systems, accelerators, and disaggregated memory further complicate coherence management. Largescale commercial architectures like AMD's chipletbased EPYC processors and Intel's multitile Xeon designs highlight that inter-die coherence introduces new latency and bandwidth constraints that conventional protocols fail to address effectively[4]. Moreover, emerging technologies such as Compute Express Link (CXL) expand the memory hierarchy into the system fabric, requiring coherence mechanisms that operate efficiently across nonuniform latency domains[5].

To address the scalability limitations of traditional coherence, researchers have explored several directions, including directory compression, timestamp-based protocols, software assisted coherence, and decentralized coherence mechanisms. Highimpact works such as Tardis[6] and Token Coherence[7] demonstrate how rethinking fundamental protocol assumptions can significantly reduce traffic and complexity. However, these techniques remain inherently reactive coherence actions occur only after cores initiate memory requests.

Predictive Cache Coherence (PCC) attempts to overcome this limitation by proactively forecasting future coherence events such as incoming invalidations, sharer changes, data migrations, or write bursts before they occur. Prediction may be based on dynamic access histories, machine learning models, heuristic policies, or hybrid logic.

Early predictive coherence studies showed promising reductions in miss latency and interconnect congestion[8], and the rise of lightweight hardware learning units has renewed interest in predictive coherence as a scalable solution for next generation architectures.

Despite growing attention, PCC research remains scattered across multiple isolated proposals. There is no unified framework for comparing predictive mechanisms, evaluating prediction accuracy, or assessing hardware overhead. Moreover, PCC introduces new challenges regarding misprediction penalties, predictor design complexity, and applicability to heterogeneous or chiplet based systems.

A comprehensive understanding of PCC techniques and their limitations is essential for advancing scalable and energy efficient coherence mechanisms.

This paper provides an in-depth survey of Predictive Cache Coherence techniques, categorizing existing research into history based, heuristic driven, machine learning based, hybrid, and speculative approaches. We evaluate their architectural implications, prediction accuracy, hardware cost, and scalability under modern system constraints. We further identify key research gaps and propose future research directions for developing robust, lightweight, and architecture aware predictive coherence mechanisms suitable for next generation manycore and heterogeneous processors.

Background and Motivation

The increasing integration of cores, accelerators, and memory subsystems in modern multiprocessor architectures has amplified the complexity and performance sensitivity of cache coherence. Early shared bus symmetric multiprocessing systems relied on snooping-based coherence, where all coherence transactions were broadcast on a shared medium. While effective for small scale systems, these schemes fail to scale due to quadratic growth in broadcast traffic and rapid interconnect saturation as core counts increase[9]. To overcome these limits, directory-based coherence protocols were introduced, providing point-to-point communication and explicit sharer tracking. Although directory protocols significantly reduce broadcast overhead, they introduce additional latency, metadata storage, and multi-hop traffic, which become pronounced in large scale and manycore systems[10,11].

As architects pursue increasingly parallel designs including 64-512core manycore CPUs, heterogeneous System on a Chips (SoCs) and chiplet based processors the underlying interconnect becomes the dominant determinant of performance and energy. Coherence traffic often accounts for a substantial portion of on-chip network load, particularly for workloads with finegrained sharing and synchronization.

Recent studies show that invalidation storms, migratory sharing, and read-write ping-pong patterns create high coherence pressure, degrading throughput and increasing energy consumption even in optimized mesh networks[12]. The emergence of chiplet integration exacerbates these issues, as coherence messages must traverse high latency die-to-die links, magnifying the cost of unnecessary communication[13].

Conventional coherence protocols such as MESI, MOESI, token coherence, and timestamp-based models remain fundamentally reactive. A request is processed after a core experiences a miss, requiring communication with sharers or the directory, resulting in multiple round trips on the network. Although several innovations attempt to reduce overhead including self-invalidation, relaxed consistency models, and hierarchical directories these techniques still struggle in environments where sharing behaviour changes rapidly or where interconnect topology imposes significant latency penalties[14,15].

These increasing pressures motivate a shift toward predictive cache coherence (PCC)**.** Instead of reacting to events after they occur, PCC techniques aim to anticipate coherence activity such as upcoming writes, sharing transitions, or phase changes in memory behaviour and take pre-emptive actions to reduce latency and communication. Predictive coherence has gained attention due to its potential to reduce directory lookups, avoid invalidation storms, and improve data locality, thereby alleviating NoC congestion and energy overheads in future manycore and heterogeneous systems. However, existing predictive approaches remain fragmented across history based, heuristic, and machine learning driven techniques, with varying assumptions and hardware costs. This motivates a comprehensive survey and critical analysis of predictive coherence techniques, situating them within the broader landscape of scalable on-chip coherence mechanisms.

Predictive Cache Coherence

Predictive Cache Coherence (PCC) represents a departure from the traditional reactive coherence paradigm by attempting to anticipate coherence events before they occur. Rather than waiting for a miss, invalidation, or directory lookup to trigger communication, predictive approaches analyze memory access behaviour, spatial and temporal patterns, and application phases to proactively manage cache states. This shift is motivated by the observation that coherence activity exhibits strong regularity across time, particularly in parallel regions with structured data sharing, loop nests, and recurring synchronization patterns. Modern predictive techniques fall into several methodological category's history based, heuristic driven, machine learning based, hybrid, and speculative coherence each offering unique trade-offs in terms of accuracy, hardware cost, and robustness across workloads.

Early predictive coherence mechanisms relied on history-based observation of previous coherence events. These techniques assume that access patterns exhibit locality across time and that future coherence actions can be inferred from observed past patterns. Alisafaee's Spatiotemporal Coherence Tracking (SCT)[11] stands as a foundational work in this category, introducing a predictor that monitors inter-core access sequences to anticipate migratory sharing. SCT reduces unnecessary invalidations and directory lookups by pre-emptively transferring ownership to the core most likely to issue the next write, improving performance in migratory workloads. Subsequent efforts refined these ideas by incorporating more detailed temporal signatures and spatial sharing vectors. For example, RegionScout[16] demonstrated that coherence traffic within loop-based programs could be significantly reduced by predicting read sharing regions and avoiding directory level sharer lookups.

Beyond history tracking, heuristic driven predictive techniques rely on lightweight rules derived from architectural insights. These approaches aim to reduce coherence overhead without resorting to fullblown learning mechanisms. Amoeba[14], for instance, employs an adaptive heuristic that reshapes coherence domain sizes based on observed sharing intensity, thus minimizing unnecessary invalidations in large many-core systems. Similarly, self-invalidation and self-downgrade protocols often incorporate predictive heuristics to anticipate when a line will no longer be reused, enabling early invalidation to eliminate directory communication under release consistency models[17]. These heuristic methods achieve low hardware cost but may suffer when sharing behaviour becomes irregular or data dependent.

With the increasing availability of percore access counters, miss profiles, and hardware performance monitoring units, machine learning based coherence prediction has emerged as a promising direction. These techniques leverage classifiers, neural predictors, or online learning models to detect patterns in memory access sequences. Recent studies demonstrate that ML based predictors can differentiate between private, read-mostly, read-shared, and migratory patterns with higher accuracy than rule-based schemes[8]. By learning application specific behaviours, ML based PCC reduces coherence misses, shrinks sharer sets, and prefetches coherence permissions. However, ML driven coherence remains constrained by hardware overhead, training convergence time, and the risk of misprediction induced coherence violations. For this reason, ML predictors are typically integrated with fallback mechanisms or selective deployment across critical memory regions.

Hybrid predictive coherence architectures combine history based and ML based strategies, capturing both long-term access trends and fine-grained behavioural nuances. Hybrid designs attempt to balance the high accuracy of ML models with the stability and low overhead of heuristic rules. For example, recent proposals integrate spatiotemporal access signatures with runtime learning modules to anticipate sharing phase transitions with improved responsiveness[18]. These hybrid methods provide robust prediction even under rapidly shifting workload phases, such as during synchronization barriers or irregular memory accesses in graph workloads.

Speculative predictive coherence extends predictive mechanisms into aggressive territory by performing coherence actions ahead of time with rollback support. In these systems, ownership transfers, invalidations, or prefetches of permissions are issued speculatively. If the predicted behaviour matches actual program behaviour, coherence latency is significantly reduced; if not, hardware performs rollback without violating memory consistency. Speculative coherence avoids the complexity of ML inference but requires careful correctness handling and potential buffering overhead, making it suitable primarily for high performance manycore processors with advanced verification support[19].

The landscape of predictive coherence is rapidly expanding, driven by the scaling demands of manycore processors, heterogeneous accelerators, and chiplet based architectures. Each technique whether history driven or ML driven contributes to reducing coherence induced NoC pressure, improving data locality, and enhancing overall performance under parallel workloads. However, their benefits must be balanced against hardware cost, training complexity, misprediction penalties, and challenges related to consistency and verification. A comprehensive evaluation of these predictive techniques is essential for determining how they integrate within future scalable system architectures.

Critical Analysis of Predictive Cache Coherence Techniques

Predictive Cache Coherence (PCC) techniques represent an important departure from conventional reactive coherence mechanisms, yet their effectiveness varies significantly depending on the architectural context, workload characteristics, prediction methodology, and hardware support. A critical examination of state-of-the-art PCC approaches reveals substantial diversity in design goals ranging from latency reduction and NoC traffic minimization to enhancing ownership locality and mitigating invalidation storms but also exposes several fundamental limitations that impede widespread deployment in commercial manycore processors.

History based prediction mechanisms, such as Spatiotemporal Coherence Tracking[19], demonstrate strong performance for migratory and producer consumer sharing patterns by exploiting predictable temporal correlations. Their primary advantage lies in their low implementation complexity, a modest set of perline signatures or timestamps often suffices to forecast the next coherence action. However, these methods degrade significantly under irregular or data dependent sharing patterns common in graph analytics, machine learning workloads, and pointer chasing applications. Moreover, history-based predictors are vulnerable to aliasing and pollution effects, where misleading past behaviour corrupts future predictions. This undermines robustness, especially in large scale systems where coherence behaviour may rapidly shift at barrier boundaries or phase transitions.

Heuristic driven predictive schemes seek lightweight solutions, but their simplicity often becomes a double-edged sword. Protocols such as Amoeba[14] and self-invalidation-based predictors[20] can effectively reduce invalidation traffic in structured parallel loops, yet they implicitly assume spatial or temporal uniformity in memory usage.

When such assumptions break down, heuristics may aggressively trigger premature invalidations or ownership transfers, ironically increasing traffic or stalling critical threads. Additionally, these techniques tend to provide coarse-grained control and struggle with fine-grained per-line behavioural differences an increasingly important factor in heterogeneous

systems where CPU cores, accelerators, and near memory units exhibit distinct access characteristics.

Machine learning based PCC mechanisms appear promising in addressing these shortcomings by capturing non-linear dependencies and complex behavioural signatures. Recent ML driven predictors[21] achieve significantly higher accuracy than heuristic or history-based designs across diverse workloads. However, their adoption raises practical concerns. First, ML models incur non-trivial hardware overhead for storage, inference, and training, which directly competes with on-chip area and power budgets particularly in energy-constrained environments such as mobile SoCs and edge processors. Second, ML predictors must remain interpretable and verifiable to guarantee correctness under stringent consistency models. Current ML based coherence proposals require fallbacks to conventional protocols in cases of misprediction, limiting the achievable performance gain and complicating verification pipelines for industry grade processors.

Hybrid predictors attempt to balance accuracy and cost by selectively combining ML inference with lightweight temporal signatures, but they inherit the complexity of both worlds. Designs such as Zuckerman et al.'s hybrid learning-based coherence orchestration mechanism[22] demonstrate improved responsiveness to rapid phase changes, yet they introduce additional metadata structures, decision logic, and multi-level filters. This increases verification difficulty and adds latency to coherence decision-making, which may counteract some of the performance benefits gained from correct predictions. Furthermore, hybrid mechanisms often rely on accurate identification of "predictable" versus "unpredictable" memory regions, a non-trivial challenge when workloads exhibit fine-grained dynamism.

Speculative coherence techniques push PCC into even more aggressive territory by executing ownership transfers or invalidations ahead of time and rolling back on misprediction. While speculative coherence such as Huh et al.'s speculative incoherent cache protocol[19] offers substantial reductions in write latency and directory indirection, it introduces rollback buffers, mis-speculation penalties, and microarchitectural replay mechanisms that greatly complicate hardware design. Maintaining consistency and avoiding live lock under speculative execution require delicate correctness conditions that remain difficult to guarantee at scale, especially in systems with heterogeneous memory regions or chiplet-based disaggregated memory.

In summary, predictive coherence offers significant potential for improving scalability and reducing coherence overhead in next generation multiprocessors. Yet, its practical adoption requires addressing fundamental gaps: robustness under irregular workloads, predictable hardware overhead, formal correctness guarantees, and adaptability to heterogeneous interconnects and chipletbased architectures. A unified, architecture aware predictive framework remains an open research challenge one that must balance prediction accuracy, energy efficiency, verification feasibility, and real-world workload diversity.

Challenges and Limitations

Despite their promise, predictive cache coherence (PCC) techniques face several fundamental limitations that complicate their adoption in real multicore and emerging heterogeneous architectures. At the core of these challenges lies the unpredictability of modern workloads. Contemporary applications particularly graph analytics, key-value stores, scientific simulations, and adaptive runtimes exhibit irregular and phase dependent sharing patterns that weaken the stability of temporal and spatial locality assumptions. Prior research has consistently shown that irregular data structures disrupt locality driven optimizations and often cause coherence systems to behave unpredictably under dynamic phases[23]. As a result, prediction accuracy deteriorates when the underlying access behaviour lacks repetition, and mispredictions may introduce unnecessary invalidations, premature ownership transfers, or additional coherence retries. Such overheads can erase the intended benefits of PCC and, in extreme cases, degrade overall system performance.

A major limitation concerns hardware cost and complexity. Modern coherence hierarchies already impose significant area, power, and verification burdens, as documented extensively in industrial and academic analyses of commercial processors[23]. Introducing prediction tables, history buffers, ML based classifiers, or speculative state machines inevitably increases storage requirements and criticalpath control logic. These additional components must operate with strict timing constraints to avoid stalling cache pipelines. The challenge is not merely one of extra hardware, but of integrating predictors without inflating energy consumption or compromising cycle time. Since predictors must react to coherence events in real time, even modest hardware latency is unacceptable, which further restricts their complexity and design freedom.

Verification emerges as an even more serious concern. Classical coherence protocols such as MESI, MOESI, and their directory-based variants are already difficult to reason about formally, yet they remain fundamentally deterministic and reactive. In contrast, predictive transitions introduce conditional, speculative, and history-dependent behaviours that complicate the state space beyond what conventional verification methods can easily manage. Sorin, Hill, and Wood highlight that even modest increases in coherence state complexity dramatically escalate formal verification difficulty[9]. When the protocol's behaviour is partially driven by prediction rather than direct architectural events, ensuring that mispredictions do not violate memory consistency becomes a non-trivial task. This tension between prediction and correctness represents one of the primary barriers preventing PCC from entering commercial processors.

Another limitation arises from the interaction between PCC and modern memory consistency models. Contemporary CPUs increasingly rely on relaxed models (e.g., RC, TSO, ARMv8) to improve performance. Predictive invalidations or speculative ownership upgrades risk exposing reordering behaviours that, although invisible in reactive protocols, may unintentionally violate ordering guarantees. Integrating PCC therefore requires careful rollback mechanisms, speculation guards, or model aware prediction boundaries to maintain correctness under all permitted executions.

These technical challenges are aggravated by the lack of standardized tools for evaluating coherence research. The literature repeatedly notes that coherence proposals are evaluated under heterogeneous conditions different network assumptions, coherence granularities, system topologies, and benchmark suites making it difficult to compare results across studies[9,23]. Without consistent evaluation frameworks, the true benefit of PCC remains hard to quantify, especially for emerging manycore systems.

The rise of chiplet based architectures and disaggregated memory introduces new sources of latency variability that PCC must account for. Cross-die communication in chiplet systems is significantly more expensive than on-die communication, and off-die memory pools accessed through fabrics like CXL introduce non-uniform coherence costs. Haris et al. demonstrate that even traditional coherence schemes suffer significant performance penalties in chiplet-based designs due to increased interconnect latency[5]. Predictive schemes that inadvertently increase cross-die messages may worsen performance rather than improve it. Moreover, as with all speculation-based microarchitectural mechanisms, predictive coherence opens potential pathways for security vulnerabilities analogous to those observed in speculative execution attacks.

In summary, while PCC provides an attractive opportunity to reduce coherence overheads, its practical deployment is constrained by prediction accuracy limits, hardware overhead considerations, verification difficulty, consistency model interactions, evaluation limitations, and emerging architectural trends. These constraints collectively explain why PCC remains a vibrant research direction but has not yet achieved widespread industrial deployment.

Future Trends and Research Directions

The evolution of future cache coherence systems will be shaped by architectural scaling trends, heterogeneous computation models, and the growing demand for predictable performance across irregular workloads. In this context, predictive cache coherence (PCC) must progress beyond single policy speculation and move toward lightweight, adaptive, and workload aware predictors**.** Since modern applications exhibit complex and time varying sharing patterns, static predictors are insufficient; effective PCC designs will require mechanisms that can learn and adjust their behaviour dynamically while remaining compatible with tight hardware latency constraints. This balance between adaptability and efficiency is central to enabling prediction without jeopardizing cycle time or energy budgets[23].

A second direction concerns verification centric protocol design. Predictive transitions, if left unconstrained, significantly expand the protocol's state space, making correctness proofs harder to construct and reason about. Since industrial designs prioritize formal guarantees, future PCC schemes must incorporate speculation models that are explicitly verification friendly potentially by embedding rollback safe transitions, bounded speculation limits, or modular microarchitectural invariants that preserve correctness even under misprediction. As noted by Nagarajan et al.[9], simplifying the reasoning structure of coherence protocols is essential for practical deployment.

The architectural environment in which coherence operates is also changing rapidly. Chipletbased processors, heterogeneous compute units, and CXL attached memory pools introduce multidomain latency and no uniform coherence cost structures**.** Predictive coherence schemes that ignore this no uniformity risk increasing off die communication and eroding performance benefits. Consequently, next-generation PCC must incorporate topology aware and latency sensitive decision frameworks**,** capable of distinguishing between on-die, cross-die, and disaggregated memory interactions to avoid counterproductive predictions[5].

Security will inevitably become a defining constraint. Just as speculative execution created pathways for microarchitectural side channels, prediction driven coherence actions may expose timing or occupancy patterns that attackers can exploit. Although little work has investigated this space, any realistic future PCC design will need security conscious predictors, reduced leakage pathways, and may require partitioned predictor structures or noise resilient mechanisms to ensure that speculation does not compromise isolation.

Progress in PCC research depends on standardized evaluation ecosystems. The absence of uniform benchmarking practices makes it difficult to compare architectural proposals or quantify real benefits across diverse workloads. Future efforts must aim to establish common simulation infrastructures, representative workload suites, and reproducible methodologies to enable meaningful cross-study comparison. With such frameworks, PCC research can transition from isolated experimental designs toward cohesive architectural progress.

Conclusion

Predictive cache coherence has emerged as a promising direction for addressing the scalability and latency limitations of traditional directory based and snooping protocols in manycore systems. By leveraging machine learning assisted prediction, temporal spatial access analysis, and lightweight history guided

speculation, recent protocols demonstrate meaningful reductions in coherence traffic, improved energy efficiency, and lower memory access latency. However, prediction driven coherence still faces challenges related to misprediction control, hardware overhead, formal verification, and integration with heterogeneous architectures such as chiplet based systems and CXL enabled memory fabrics. Future systems will likely adopt hybrid predictive reactive coherence, combining learning-based predictors with robust fallbacks and verification friendly designs. This positions predictive coherence as a key enabling technology for next generation manycore processors and emerging memory hierarchies.

References

1 Hardavellas, N., Ferdman, M., Falsafi, B., and Ailamaki, A. (2009). Reactive NUCA: Near-optimal block placement and replication in distributed caches. *Proceedings of the 36th Annual International Symposium on Computer Architecture*, New York, USA, pp. 184–195.

2 Chang, J., Herrero, E., Canal, R., and Sohi, G. S. (2011). Cooperative caching for chip multiprocessors. *Cooperative Networking*, Wiley, pp. 217–275.

3 Stenstrom, P. (1990). A survey of cache coherence schemes for multiprocessors. *Computer*, 23(6), 12–24.

4 Naffziger, S., Beck, N., Burd, T., Lepak, K., Loh, G. H., Subramony, M., & White, S. (2021). Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families: Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (pp. 57-70). IEEE.

5 Das Sharma, D., Blankenship, R., and Berger, D. (2024). An introduction to the Compute Express Link (CXL) interconnect. *ACM Computing Surveys*, 56(11), 1–37.

6 Yu, X., and Devadas, S. (2015). Tardis: Time traveling coherence algorithm for distributed shared memory. *Proceedings of the International Conference on Parallel Architecture and Compilation Techniques*, IEEE, 227–240.

7 Martin, M. M. K., Hill, M. D., and Wood, D. A. (2003). Token coherence: Decoupling performance and correctness. *ACM SIGARCH Computer Architecture News*, 31(2), 182–193.

8 Cho, M. and Chung, E.Y. (2025). MLCRP: ML-based GPU cache performance modeling featured with reuse profiles. *IEEE Access*, 13, 126610–126622.

9 Nagarajan, V., Sorin, D. J., Hill, M. D., and Wood, D. A. (2020). *A Primer on Memory Consistency and Cache Coherence*. Springer International Publishing, Cham.

10 Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., ... & Falsafi, B. (2012). Clearing the clouds: a study of emerging scale-out workloads on modern hardware. *Acm sigplan notices*, 47(4), 37-48.

11 Alisafaee, M. (2012). Spatiotemporal coherence tracking. *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, pp. 341–350.

12 Xiong, G., Luo, X., and Liu, W. (2025). Learning cache coherence traffic for NoC routing design. *Proceedings of the Great Lakes Symposium on VLSI*, ACM, pp. 420–426.

13 Wang, R., Yu, L., and Zhuang, W. (2024). Design of an efficient hybrid cache coherence protocol on chiplet architecture. *Proceedings of the Third International Conference on Algorithms, Microchips, and Network Applications (AMNA)*, SPIE, p. 29.

14 Kumar, S., Zhao, H., Shriraman, A., Matthews, E., Dwarkadas, S., and Shannon, L. (2012). Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy. *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, pp. 376–388.

15 Han, L., An, J., Gao, D., Fan, X., Ren, X., and Yao, T. (2012). A survey on cache coherence for tiled many-core processor. *Proceedings of the IEEE International Conference on Signal Processing, Communication and Computing*, IEEE, pp. 114–118.

16 Cantin, J. F., Smith, J. E., Lipasti, M. H., Moshovos, A., and Falsafi, B. (2006). Coarse-grain coherence tracking: Region Scout and region coherence arrays. *IEEE Micro*, 26(1), 70–79.

17 Zhang, R., Biswas, S., Balaji, V., Bond, M. D., and Lucia, B. (2021). Neat: Low-complexity, efficient on-chip cache coherence. *arXiv Preprint*.

18 Marandola, J., Louise, S., and Cudennec, L. (2016). Pattern based cache coherency architecture for embedded manycores. *Procedia Computer Science*, 80, 1542–1553.

19 Huh, J., Burger, D., Chang, J., and Sohi, G. S. (2004). Speculative incoherent cache protocols. *IEEE Micro*, 24(6), 104–109.

20 Abdulla, P. A., Atig, M. F., Kaxiras, S., Leonardsson, C., Ros, A., and Zhu, Y. (2018). Mending fences with self-invalidation and self-downgrade. *Logical Methods in Computer Science*, 14(1).

21 Zhao, Y., Shi, B., Zhang, Q., Yuan, Y., and He, J. (2023). Research on cache coherence protocol verification method based on model checking. *Electronics*, 12(16), 3420.

22 Zuckerman, J., Giri, D., Kwon, J., Mantovani, P., and Carloni, L. P. (2021). Cohmeleon: Learning-based orchestration of accelerator coherence in heterogeneous SoCs. *Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, ACM, pp. 350–365.

23 Martin, M. M. K., Hill, M. D., and Sorin, D. J. (2012). Why on-chip cache coherence is here to stay. *Communications of the ACM*, 55(7), 78–89.