RESEARCH STATEMENT

Anirudh Mohan Kaushik

Cyber-physical systems (CPS) are interconnected systems of sensors and computing devices that sense and collect information about the physical world, and in turn operate on this sensed information to interact with and influence the physical world. Application domains of CPS range from healthcare in the form of implantable medical devices such as defibrillators and pacemakers [1, 2], agriculture where sensors and satellites monitor and regulate soil and crop activity [3], and transportation and aerospace in the form of assisted driver autonomous systems (ADAS), unmanned aerial vehicles (UAVs) and drones [4, 5]. These systems harness the power of compute to enrich our experiences with the physical world and have the potential to solve important societal challenges such as mitigating climate change [6, 7]. As we continue to entrust more parts of our lives to CPS, it is imperative that the underlying hardware and software computing systems on which these CPS rely on are designed *correctly* for safe operation. However, what constitutes correct system design *differs* between manufacturers that design and distribute the systems and CPS domains that use these systems. For manufacturers, producing correct values for all deployed software constitutes as correct design, whereas in safety-critical CPS domains, producing correct values *and the time it takes to produce the correct values (timing predictability)* constitute as correct design. As a concrete example, in ADAS, the time taken to correctly detect and classify objects on the road affects the engagement and execution of collision avoidance maneuvers, and hence, affects the safety of passengers. This notion of time as a first-class design principle for computing system design introduces challenges to conventional design processes and methodologies. Meeting these challenges through novel design strategies where time is a first-class design principle while accommodating and integrating conventional design methodologies are crucial to sustaining the tremendous potential benefits that CPS offer and accelerating their deployment and outreach in our lives and society at large.

I am a hardware and software systems researcher whose research primarily looks at the design of efficient computing systems for CPS. Specifically, my research focuses on *designing timing predictable and high-performance compute systems for CPS*. Timing predictability is crucial for correct and safe operation for CPS. Growing complexity of services deployed on CPS and demand for more autonomy via machine learning have raised the high-performance requirements of the computing systems used by CPS [8, 9]. However, designing for timing predictability focuses on and optimizes for *worst-case scenarios*, whereas designing for high-performance focuses on and optimizes for *average-case scenarios*. This makes timing predictability and high-performance *conflicting* design goals. For example, research works have shown that design features in commercially available computing systems that are responsible for fast execution of average-case scenarios are also responsible for significantly slowing down execution of worst-case scenarios, which can compromise safe execution for safety-critical CPS [10]. Disabling these performance features for improved timing predictability or designing solely for timing predictability are shortsighted approaches that ignore the growing integration of machine learning with CPS and the multitude of opportunities that this integration provides. To this end, my research philosophy in CPS is on *designing computing systems that strike a balance between timing predictability and high-performance*. My research approach has two steps: (1) *building timing analysis models* to identify barriers to predictability in current systems design, and (2) using insights derived from the timing models to *propose design methodologies* that balance predictability and high-performance. My dissertation research focused on *data communication mechanisms* between multiple applications, which is a critical component of computing system design [11, 12, 13, 14, 15, 16]. Outside of my dissertation research, I have applied this philosophy to other components of compute system design such as memory controllers [17, 18], cache controllers [19] and more recently, instruction scheduling and execution in graphics processing units (GPUs) [20].

Dissertation Research: Predictable Data Communication

Software applications deployed on CPS such as those from machine learning, computer vision, and scientific computing domains exhibit increasing data and task parallelism [21], growing amounts of data processed and manipulated [22, 23], and frequent data communication [24, 25]. To meet these application requirements in a safety-critical setting, there is a large body of research devoted towards improving the timing predictability of multi-core platforms, GPUs, and multi-processor system-on-chips (MPSoCs), which has helped paved the way for their increasing adoption in CPS [26]. The crux of the low timing predictability of these platforms is the timing interference from *shared hardware resources* such as shared memory units (caches, main-memory) and interconnects that are accessed simultaneously by multiple applications executing on the platform. Data communication between applications executing on different compute units further *compounds* the low timing predictability as data communicated from one application to another must take into account timing interference from shared hardware resource accesses *and the state of the communicated data*.

For my dissertation, I focused on exploring *hardware cache coherence mechanisms* as a solution to facilitate predictable and high-performance data communication in CPS. At a high level, hardware cache coherence enables multiple cores to correctly access the most up-to-date shared data, and allow multiple cores to simultaneously cache shared data in their private cache memories. This hardware mechanism works transparent to the software application, and allows applications

running simultaneously on different compute units to communicate data correctly without duplicating data or imposing constraints on data caching and placement. My dissertation research focused on the following questions: (1) *What are the sources of timing interference when using hardware cache coherence?*, (2) *What design guidelines must be followed to build predictable hardware cache coherence mechanisms?*, and (3) *What are the key design ingredients that impact performance and predictability in hardware cache coherence and how best to reconcile them?*. Exploring these questions, I proposed design methodologies and analyses frameworks [11, 12, 13, 14] and design tools to build predictable cache coherence mechanisms [15, 16]. To the best of my knowledge, the research works constituting my dissertation were the first to explore design methodologies and analyses frameworks for predictable hardware cache coherence, and have fostered further research in this area [27, 28, 29, 30, 31, 32, 33]. Below, I expand on the main components of my dissertation.

Establishing design invariants and guidelines [11, 12]: The first set of works analyzed potential timing issues that can arise from using hardware cache coherence mechanisms and established a set of design guidelines that addressed these issues. Specifically, I analyzed scenarios of *unbounded execution time* that can arise using an abstract cache coherence model. These scenarios were constructed based on the state of communicated data, the access patterns on communicated data (read/write), and the state of the shared resources (interconnects and memory units). The *key contribution* of these works was the *establishment of a set of design invariants that guide the design of a predictable cache coherence mechanism*. The rationale behind establishing design invariants rather than imposing a specific implementation were twofold. First, design invariants offer flexibility on the implementation, which opens up design space exploration based on the requirements and deployment of the CPS. Second, given the implementation details of an existing cache coherence implementation, the design invariants can serve as a checklist to determine the predictability of the implementation. These design invariants laid the foundation for the rest of the works in my dissertation.

Using these design invariants, I developed timing predictable variants of existing high-performance implementations such as predictable modified-shared-invalid (PMSI), predictable modified-exclusive-shared-invalid (PMESI), predictable modified-owned-exclusive-shared-invalid (PMOESI) cache coherence mechanisms. For each of these implementations, I derived the worst-cast data communication latency using formal timing analysis methods. Empirical evaluation with real-world workloads and synthetic workloads that stress the worst-case scenarios showed that the observed worst-case data communication latency were within the computed worst-case bound derived from the formal timing analyses. Performance evaluation showed that predictable cache coherence mechanisms offered significant performance advantages (upto 4×) over existing approaches while still being predictable. In summary, this set of works showed that high-performance and predictable hardware cache coherence mechanisms can be designed to facilitate data communication in CPS.

Balancing predictability and performance [14]: A key concern that arose from the first set of works was that applications under predictable cache coherence mechanisms incurred higher worst-case execution times (WCETs) compared to existing approaches. To put this in perspective, for an 8-core platform, I showed that the data communication latency under predictable cache coherence is 16× higher compared to other approaches (from static timing analyses) with a 5× performance speedup benefit (from empirical evaluation). Despite the stellar performance benefits predictable cache coherence mechanisms offered, this high WCETs threatened any potential for their adoption in CPS.

To this end, I focused on understanding the design ingredients responsible for this high worst-case data communication latency under predictable cache coherence mechanisms and design strategies to reduce this communication latency while guaranteeing performance. The *key contribution* of this work was a systematic approach that defined a formal model of a cache coherence mechanism and the application of this model to identify the key design ingredients that contribute to high worst-case communication latency. A novelty of the formal model was a timing analyses based on *worst-case asymptotic latency* where the communication latency is expressed as a function of core count. Such a timing analyses is beneficial to capture first-order behavior of a mechanism without requiring detailed information about the implementation. Using the insights derived from the formal model, I explored one design technique that focused on changes to the underlying cache coherence protocol state machines; I explored another design technique to achieve the same with my co-authors in [34]. The resulting predictable cache coherence mechanisms, referred to as *linear cache coherence mechanisms*, traded minor performance to achieve the same worst-case communication latency as existing approaches. Empirical evaluation showed that linear cache coherence mechanisms sacrificed at most 13% performance compared to the original predictable cache coherence mechanisms for a 94% reduction in worst-case latency for an 8-core compute system model. In summary, I showed that a systematic approach to predictable cache coherence design can result in minimizing the worst-case communication latency and improve their prospects for adoption and implementation in computing systems for CPS.

Designing for mixed-criticality CPS [13]: CPS such as advanced driver-assistance systems deploy a multitude of applications on the compute platform that have *varying requirements on timing predictability and performance* [35]. For instance, these applications range from media applications such as in-car entertainment that have low requirements on timing predictability or *low-criticality* but benefit from high-performance to applications such as object avoidance and tracking that have strict predictability requirements or *high-criticality*. Handling predictable data communication on mixed-criticality CPS introduced an additional challenge wherein data communication between mixed-criticality applications must not result in timing interference to the high-criticality applications [36, 37]. To this end, I developed CARP,

a novel criticality-aware predictable cache coherence mechanism that facilitated data communication between different criticality levels while satisfying this mandate. The *key design novelty* of CARP was that coherence state transitions were triggered on both the operation type (read/write) *and the criticality levels of the applications performing the operations*. Infusing the cache coherence protocol state machine with information on criticality levels required micro-architectural extensions and state machine changes. CARP prioritized data communication from high-criticality applications, and hence ensured that latency of communication initiated by high-criticality applications was independent of the communication from low-criticality applications.

Design tools, proof-of-concepts, and open-source artifacts [15, 16]: Designing a hardware cache coherence mechanism for CPS is a complex design exercise as it requires a designer to carefully reason about correctness, predictability and high-performance. To ease this complex design exercise, I consolidated the design insights and formal models proposed in my research so far into `Synthia`, an automated tool to construct predictable and high-performance hardware cache coherence protocol state machines. The *key objective* of `Synthia` was to improve designer productivity and facilitate design space exploration of predictable cache coherence protocol state machines. `Synthia` took as input a simple specification of protocol state machine devoid of any predictability and performance features, and produced a correct, predictable, and high-performance cache coherence protocol state machine. To put into perspective `Synthia`'s benefits, `Synthia` took a specification of a cache coherence protocol deployed in a commercial multi-core processor that consisted of 25 state transitions and generated a predictable and high-performance protocol state machine with 118 state transitions (more than 4× the number of the transitions). I evaluated `Synthia` with several cache coherence protocol specifications, and the corresponding output protocol state machines were subjected to verification for correctness and empirical evaluation for predictability and performance. I also co-developed `MapleBoard` [34], a predictable and *open-sourced* RISC-V multi-core hardware test-bed for deploying and evaluating predictable cache coherence mechanisms. The key objective of `MapleBoard` was to provide researchers a FPGA-based test bed for rapid exploration, implementation, and evaluation of predictable cache coherence mechanisms.

A core tenet of my research philosophy is making *all* of my research open-source and available for public use. I have open-sourced research artifacts such as the micro-architectural simulator used to implement and evaluate proposed mechanisms, benchmarks, and tools such as `Synthia` and `MapleBoard`. Making research artifacts available for public use helps disseminate research ideas and methodologies, encourage further contributions that expand the research area, and opens up opportunities for discussions with manufacturers and stakeholders regarding the adoption and integration of proposed designs in future compute platform offerings.

Future Research Agenda

Exploring memory models for CPS: CPS exhibit diverse data communication patterns and scenarios between deployed applications [24]. This diversity arises from data placement, data and task dependencies, criticality levels, and task execution constraints. Abstracting away this diversity, data communication fundamentally reduces to *reads and writes on data that reside in shared memory*. One area of future research focuses on exploring *memory consistency models* for CPS that specify the accepted behavior of tasks operating on shared data. The main motivation for this research focus is that existing approaches to handling data communication in CPS must work within the confines of the memory model defined by the underlying computing system. As with most micro-architectural performance features on existing computing systems, these memory models are also designed with performance in mind to allow flexible reordering of reads and writes to shared data [38]. A memory consistency model appropriate for CPS can offer some relaxations in ordering of reads and writes to shared memory based on certain properties of the shared data and task. Such a memory consistency model for CPS has the potential to open up several design opportunities for data communication. In the future, I will look into exploring the applicability of existing relaxed memory consistency models for CPS, and explore new memory models that better suit the data communication patterns in CPS. It is important to note that designing a memory consistency model has ramifications to the micro-architecture and instruction set architecture (ISA) in the form additional instructions to enforce ordering and visibility of updates to shared memory. Hence, a key objective of this exploration will be to design memory models on which existing software architectures and methodologies can be applied as-is and at the same time, explore alternate software methodologies that make use of the proposed memory model features.

Designing predictable MPSoCs: Heterogeneous MPSoCs integrate multiple compute cores, GPUs, programmable FPGAs and custom machine learning compute engines onto a single compute platform [39]. There is growing adoption of such platforms in CPS as they cater to a myriad of different application deployed on CPS [40, 41]. However, heterogeneous MPSoCs bring unique challenges to predictability. Specifically, I am interested in two research areas. First, GPUs on heterogeneous MPSoCs have been shown to exhibit low timing predictability making their adoption a serious concern [42, 43]. The low timing predictability arises from certain implementation details arising from single-instruction-multiple-data instruction execution [20] along with the presence of shared hardware resources such as cache memories, interconnects, and shared memory [44]. There is rich body of research work that focuses on improving the timing predictability of COTS

GPUs through reverse-engineering micro-architectural details [45, 46] and software approaches through programming models and driver intervention [47]. A recent work with my co-authors looked at improving the predictability of GPUs through a combination of micro-architectural extensions and compiler optimizations [20]. In this work, we looked at the impact of *branch-divergence* on WCET and proposed micro-architectural extensions to reduce their impact to WCET. Branch divergence happens when a group of threads executing in lockstep encounters conditional statements that result in converting the lockstep execution into a serialized execution. Motivated by this work, in the future, I will continue look into exploring and improving the predictability of GPUs. My previous research on GPU software design [48, 49, 50, 51] and industrial experience in compiler design for high-performance GPUs make me well suited for research in this area.

Second, the heterogeneous processors on an MPSoC share the main-memory and applications running on different processors can communicate data with each other. Heterogeneous processors implement different cache hierarchy designs and have different sensitivities to communication latency and throughput [52, 53, 54]. As a result, predictable cache coherence and memory consistency models need to be designed that take into account this heterogeneity. In the future, I am interested in building on the design insights derived from my dissertation towards building novel predictable cache coherence models and memory consistency models for predictable heterogeneous MPSoCs.

Optimizing on-chip data movement: A common application characteristic underlying both safety-critical applications and high-performance compute applications is the increasing volumes of data accessed and operated on by compute agents. This increasing volume of data stresses the on-chip interconnects and cache hierarchies resulting increased power consumption and performance penalties. This application trend requires novel micro-architectural solutions in the memory hierarchy that reduces data movement and improves the utility of the cache hierarchy for these applications with large data footprint. During my dissertation, I worked on a novel hardware data-dependent prefetcher for graph analytics called Gretch [55]. Data accesses in graph analytics comprise of data-dependent accesses and these accesses have low spatial and temporal reuse [56]. Gretch was a novel hardware data-dependent prefetcher that identified data-dependent accesses and scheduled prefetches for data-dependent accesses into the cache hierarchy. In the future, I will look into opportunities for designing micro-architectural extensions that improve on-chip data movement and cache usage for emerging workloads in the area of machine learning and data analytics. In the machine learning domain, I am keen to apply my experience with improving the performance of graph analytics on graph neural networks (GNNs). Specifically, I am interested in looking at leveraging application knowledge to infuse semantics of data operations into the memory hierarchy. This information can be leveraged by a programmable micro-architectural extension in the memory hierarchy to better manage on-chip data movement.

Conclusion

Design of efficient hardware and software compute systems is vital to sustain the current machine learning revolution. Each component of the compute stack starting from the high-level language design, compiler framework, operating system, and micro-architecture is crucial towards sustaining this revolution. I have been fortunate to perform research in compute systems spanning *breadth* in terms of different areas of compute systems deployment (CPS and high-performance compute systems) and *depth* in terms of different components of the compute stack (software frameworks, compiler design, micro-architecture). My aim through my research is to contribute to this revolution by exploring design methodologies to build efficient hardware and software compute systems, and in the process, help build a future where better and efficient computing systems can solve the big challenges facing humanity.

References

[1] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, et al. Challenges and research directions in medical cyber–physical systems. *Proceedings of the IEEE*, 100(1):75–90, 2011.

[2] Nilanjan Dey, Amira S Ashour, Fuqian Shi, Simon James Fong, and João Manuel RS Tavares. Medical cyber-physical systems: A survey. *Journal of medical systems*, 42:1–13, 2018.

[3] W An, D Wu, S Ci, H Luo, V Adamchuk, and Z Xu. Agriculture cyber-physical systems. In *Cyber-physical systems*, pages 399–417. Elsevier, 2017.

[4] Krishna Sampigethaya and Radha Poovendran. Aviation cyber–physical systems: Foundations for future aircraft and air transport. *Proceedings of the IEEE*, 101(8):1834–1855, 2013.

[5] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 25(8):425–466, 2008.

[6] Oliver Inderwildi, Chuan Zhang, Xiaonan Wang, and Markus Kraft. The impact of intelligent cyber-physical systems on the decarbonization of energy. *Energy & Environmental Science*, 13(3):744–771, 2020.

[7] Zhaohui Wang, Houbing Song, David W Watkins, Keat Ghee Ong, Pengfei Xue, Qing Yang, and Xianming Shi. Cyber-physical systems for water sustainability: challenges and opportunities. *IEEE Communications Magazine*, 53(5):216–222, 2015.

[8] ARM Technology. ARM Expects Vehicle Compute Performance to Increase 100x in Next Decade, 2015.

[9] Leela Karumbunathan. NVIDIA Jetson AGX Orin Series, Technical Brief, 2022.

[10] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, et al. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):1–53, 2008.

[11] Mohamed Hassan, **Kaushik, Anirudh M.**, and Hiren Patel. Predictable Cache Coherence for Multi-core Real-Time Systems. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 235–246, 2017.

[12] **Kaushik, Anirudh Mohan**, Mohamed Hassan, and Hiren Patel. Designing Predictable Cache Coherence Protocols for Multi-Core Real-Time Systems. *IEEE Transactions on Computers*, 70(12):2098–2111, 2021.

[13] **Kaushik, Anirudh Mohan**, Paulos Tegegn, Zhuanhao Wu, and Hiren Patel. CARP: A Data Communication Mechanism for Multi-core Mixed-Criticality Systems. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 419–432, 2019.

[14] **Kaushik, Anirudh Mohan** and Hiren Patel. A Systematic Approach to Achieving Tight Worst-Case Latency and High-Performance Under Predictable Cache Coherence. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 105–117, 2021.

[15] **Kaushik, Anirudh Mohan** and Hiren Patel. Automated Synthesis of Predictable and High-Performance Cache Coherence Protocols. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 816–821, 2021.

[16] **Kaushik, Anirudh Mohan** and Hiren Patel. Automatic Construction of Predictable and High-Performance Cache Coherence Protocols for Multicore Real-Time Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(10):3318–3331, 2022.

[17] Mohamed Hassan, **Kaushik, Anirudh M.**, and Hiren Patel. Reverse-engineering embedded memory controllers through latency-based analysis. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 297–306, 2015.

[18] Mohamed Hassan, **Kaushik, Anirudh M.**, and Hiren Patel. Exposing Implementation Details of Embedded DRAM Memory Controllers through Latency-Based Analysis. *ACM Transactions on Embedded Computing Systems*, 17(5), 2018.

[19] Zhuanhao Wu, **Kaushik, Anirudh**, and Hiren Patel. ZeroCost-LLC: Shared LLCs at No Cost to WCL. In *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 249–261, 2023.

[20] Artem Klashtorny, Zhuanhao Wu, **Kaushik, Anirudh Mohan**, and Hiren Patel. Predictable GPU Wavefront Splitting for Safety-Critical Systems. *ACM Transactions on Embedded Computing Systems*, 22(5s), 2023.

[21] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 751–766, 2018.

[22] Meng Xu, Linh Thi, Xuan Phan, Hyon-Young Choi, and Insup Lee. vCAT: Dynamic Cache Management Using CAT Virtualization. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 211–222, 2017.

[23] Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu. The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory. In *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, page 62–75, New York, NY, USA, 2015. Association for Computing Machinery.

[24] Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, and Falk Wurst. Communication centric design in complex automotive embedded systems. In *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[25] Lukas Osinski, Tobias Langer, Ralph Mader, and Jürgen Mottok. Challenges and Opportunities with Embedded Multicore Platforms: Spotlight on Real-Time and Dependability Concepts. In *9th European Congress Embedded Real Time Software and Systems (ERTS 2018)*, 2018.

[26] Tamara Lugo, Santiago Lozano, Javier Fernández, and Jesus Carretero. A survey of techniques for reducing interference in real-time applications on multicore platforms. *IEEE Access*, 10:21853–21882, 2022.

[27] Nathanaël Sensfelder, Julien Brunel, and Claire Pagetti. On How to Identify Cache Coherence: Case of the NXP QorIQ T4240. In Marcus Völp, editor, *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, volume 165 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:22, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[28] Nathanaël Sensfelder, Julien Brunel, and Claire Pagetti. Modeling Cache Coherence to Expose. In *ECRTS 2019*, 2019.

[29] Ayoosh Bansal, Jayati Singh, Yifan Hao, Jen-Yang Wen, Renato Mancuso, and Marco Caccamo. Reconciling predictability and coherent caching. In *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–6. IEEE, 2020.

[30] Roger Pujol, Hamid Tabani, Jaume Abella, Mohamed Hassan, and Francisco J Cazorla. Empirical Evidence for MPSoCs in Critical Systems: The Case of NXP's T2080 Cache Coherence. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1162–1165. IEEE, 2021.

[31] Roger Pujol, Mohamed Hassan, Hamid Tabani, Jaume Abella, and Francisco Javier Cazorla. Tracking Coherence-Related Contention Delays in Real-Time Multicore Systems. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 461–470, 2023.

[32] Zhuanhao Wu, Marat Bekmyrza, Nachiket Kapre, and Hiren Patel. Ditty: Directory-based Cache Coherence for Multicore Safety-critical Systems. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

[33] Arun S. Nair, Aboli Vijayan Pai, Biju K. Raveendran, and Geeta Patil. MOESIL: A Cache Coherency Protocol for Locked Mixed Criticality L1 Data Cache. In *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, 2021.

[34] Zhuanhao Wu, **Kaushik, Anirudh Mohan**, Paulos Tegegn, and Hiren Patel. A Hardware Platform for Exploring Predictable Cache Coherence Protocols for Real-time Multicores. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 92–104, 2021.

[35] Alan Burns and Robert I Davis. A survey of research into mixed criticality systems. *ACM Computing Surveys (CSUR)*, 50(6):1–37, 2017.

[36] International Organization for Standardization. ISO 26262, 2011.

[37] Simon Fürst, Jürgen Mössinger, Stefan Bunzel, Thomas Weber, Frank Kirschke-Biller, Peter Heitkämper, Gerulf Kinkelin, Kenji Nishikawa, and Klaus Lange. AUTOSAR–A Worldwide Standard is on the Road. In *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, volume 62, page 5, 2009.

[38] Vijay Nagarajan, Daniel J Sorin, Mark D Hill, and David A Wood. *A primer on memory consistency and cache coherence*. Springer Nature, 2020.

[39] Mohamed Hassan. Heterogeneous MPSoCs for mixed criticality systems: Challenges and opportunities. *arXiv preprint arXiv:1706.07429*, 2017.

[40] E. Talpes, D. Sarma, G. Venkataramanan, P. Bannon, B. McGee, B. Floering, A. Jalote, C. Hsiong, S. Arora, A. Gorti, and G. S. Sachdev. Compute Solution for Tesla's Full Self-Driving Computer. *IEEE Micro*, 40(02):25–35, 2020.

[41] Vamsi Boppana, Sagheer Ahmad, Ilya Ganusov, Vinod Kathail, Vidya Rajagopalan, and Ralph Wittig. UltraScale+ MPSoC and FPGA families. In *2015 IEEE Hot Chips 27 Symposium (HCS)*, pages 1–37. IEEE, 2015.

[42] Nathan Otterness, Ming Yang, Sarah Rust, Eunbyung Park, James H. Anderson, F. Donelson Smith, Alex Berg, and Shige Wang. An Evaluation of the NVIDIA TX1 for Supporting Real-Time Computer-Vision Workloads. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 353–364, 2017.

[43] Glenn A Elliott, Bryan C Ward, and James H Anderson. GPUSync: A framework for real-time GPU management. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 33–44. IEEE, 2013.

[44] Nicola Capodieci, Roberto Cavicchioli, Ignacio Sañudo Olmedo, Marco Solieri, and Marko Bertogna. Contending memory in heterogeneous SoCs: Evolution in NVIDIA Tegra embedded platforms. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10. IEEE, 2020.

[45] Tanya Amert, Nathan Otterness, Ming Yang, James H. Anderson, and F. Donelson Smith. GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 104–115, 2017.

[46] Ming Yang. Avoiding pitfalls when using NVIDIA GPUs for real-time tasks in autonomous systems. In *Proceedings of the 30th Euromicro Conference on Real-Time Systems*, 2018.

[47] Björn Forsberg, Luca Benini, and Andrea Marongiu. HePREM: A predictable execution model for GPU-based heterogeneous SoCs. *IEEE Transactions on Computers*, 70(1):17–29, 2020.

[48] **Kaushik, Anirudh Mohan**, Ashwin M. Aji, Muhammad Amber Hassaan, Noel Chalmers, Noah Wolfe, Scott Moe, Sooraj Puthoor, and Bradford M. Beckmann. Optimizing Hyperplane Sweep Operations Using Asynchronous Multi-grain GPU Tasks. In *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pages 59–69, 2019.

[49] **Kaushik, Anirudh** and Hiren D. Patel. Systemc-clang: An open-source framework for analyzing mixed-abstraction SystemC models. In *Proceedings of the 2013 Forum on specification and Design Languages (FDL)*, pages 1–8, 2013.

[50] Valeria Bertacco, Debapriya Chatterjee, Nicola Bombieri, Franco Fummi, Sara Vinco, A. M. Kaushik, and Hiren D. Patel. On the use of GP-GPUs for accelerating compute-intensive EDA applications. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1357–1366, 2013.

[51] Mahesh Nanjundappa, **Kaushik, Anirudh**, Hiren D. Patel, and Sandeep K. Shukla. Accelerating SystemC simulations using GPUs. In *2012 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pages 132–139, 2012.

[52] Jason Power, Arkaprava Basu, Junli Gu, Sooraj Puthoor, Bradford M Beckmann, Mark D Hill, Steven K Reinhardt, and David A Wood. Heterogeneous system coherence for integrated CPU-GPU systems. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 457–467, 2013.

[53] Johnathan Alsop, Matthew Sinclair, and Sarita Adve. Spandex: A flexible interface for efficient heterogeneous coherence. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 261–274. IEEE, 2018.

[54] Susmita Tadepalli, Zhuanhao Wu, and Hiren Patel. PASoC: A Predictable Accelerator-rich SoC. In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, pages 325–330. 2023.

[55] **Kaushik, Anirudh Mohan**, Gennady Pekhimenko, and Hiren Patel. Gretch: A Hardware Prefetcher for Graph Analytics. *ACM Transactions on Architecture and Code Optimization*, 18(2), 2021.

[56] Abanti Basak, Shuangchen Li, Xing Hu, Sang Min Oh, Xinfeng Xie, Li Zhao, Xiaowei Jiang, and Yuan Xie. Analysis and Optimization of the Memory Hierarchy for Graph Processing Workloads. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 373–386, 2019.