04/26/2024

KIIT University

PROJECT GUIDE

Prof. Dr.
Banchhanidhi Dash

PRESENTED BY

Manash Sangam
(20051731)

Sandesh Ghimire
(20051753)

Performance Analysis of Last level cache using Heterogeneity Aware Replacement of heterogeneous Multi-core ARM based processor architecture to expedite the read performance for IOT based system

Table of Contents

I Introduction

This paper delves into the fundamental design concepts, inherent challenges, and evolving trends in ARM-based heterogeneous multi-core processors, illuminating their potential to revolutionize the landscape of computing systems, particularly in the context of IoT applications.

The Figure illustrates the architecture of a scalable System-on-Chip (SoC) featuring ARM's big.LITTLE configuration, a groundbreaking approach in processor design. At its core are four ARM Cortex-A76 processors, representing the "big" cores optimized for high-performance computing tasks. Complementing these are four ARM Cortex-A55 processors, the "LITTLE" cores designed for energy-efficient operations. These cores are interconnected in a ring architecture, enabling efficient communication and workload distribution. This heterogeneous multi-core setup allows the SoC to dynamically allocate tasks based on their computational requirements, optimizing both performance and power consumption.

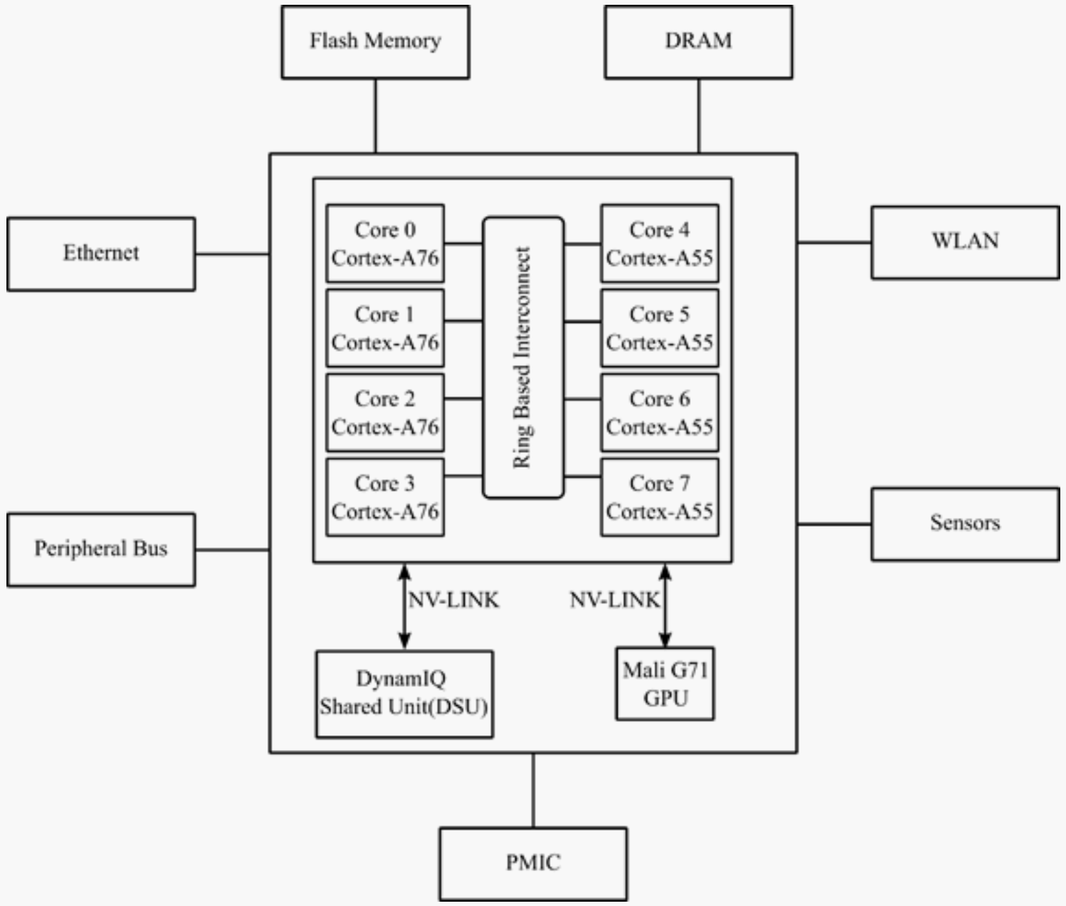


FIGURE: SCALABLE ARM BASED MULTI-CORE SOC

I Introuduction

SoC Components

- ARM Cortex A76
- ARM Cortex A55
- Mali G71 GPU
- DynamIQ Shared Unit(DSU) Cluster

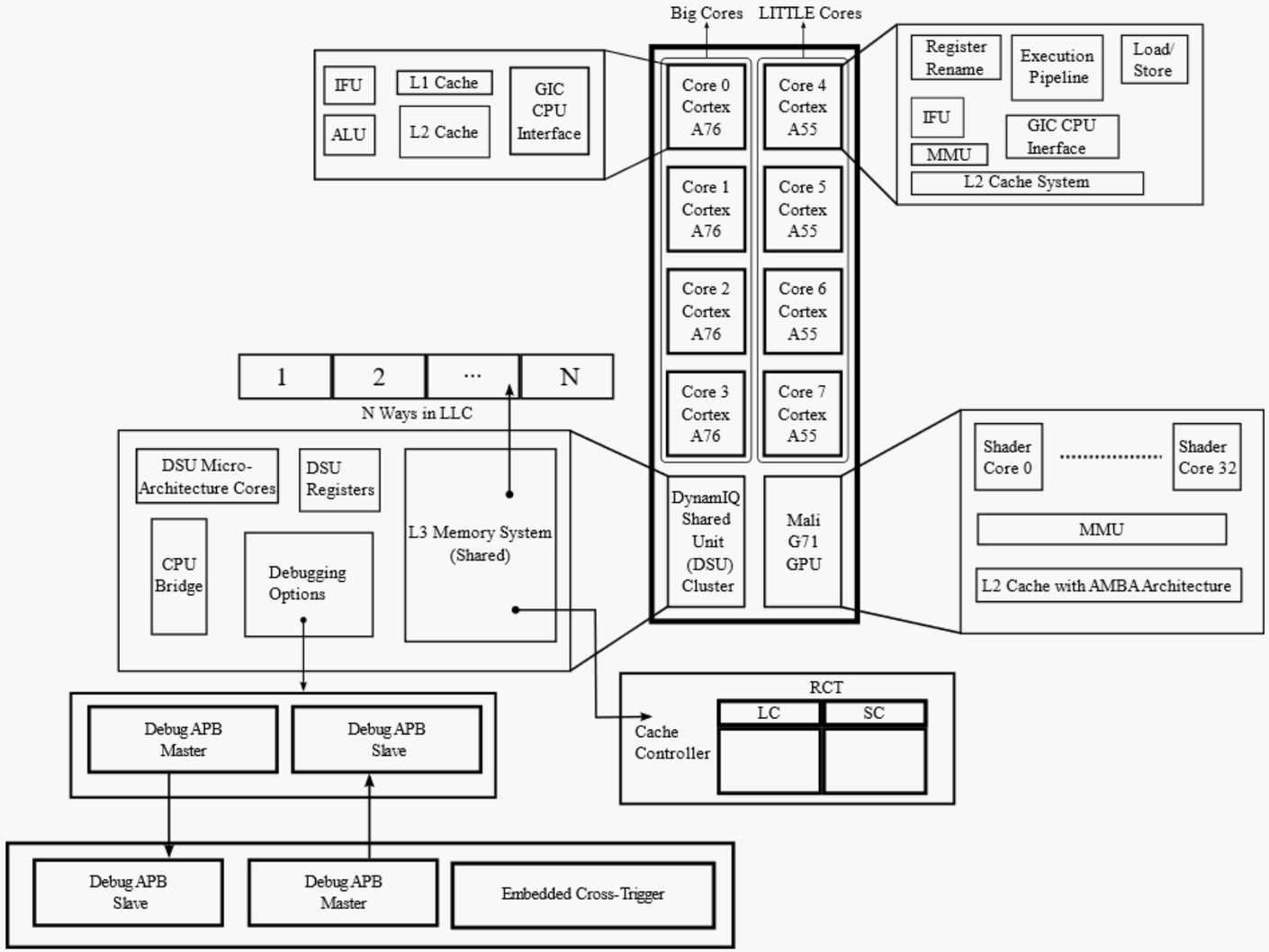


Figure : The processor configuration in detail

II Research Background & Motivation

We have identified following objectives for our thesis based on previous works to maximize the utilization of the shared LLC in our SoC model configuration of ARM big.LITTLE architecture ISA.

Objective-1: To introduce Heterogeneity Sensible Replacement for Partitioned Cache(HSRPC) in shared LLC to expedite cache utilization and system performance over scalable ARM based Heterogeneous multi-core processor.

III Simulation Environment

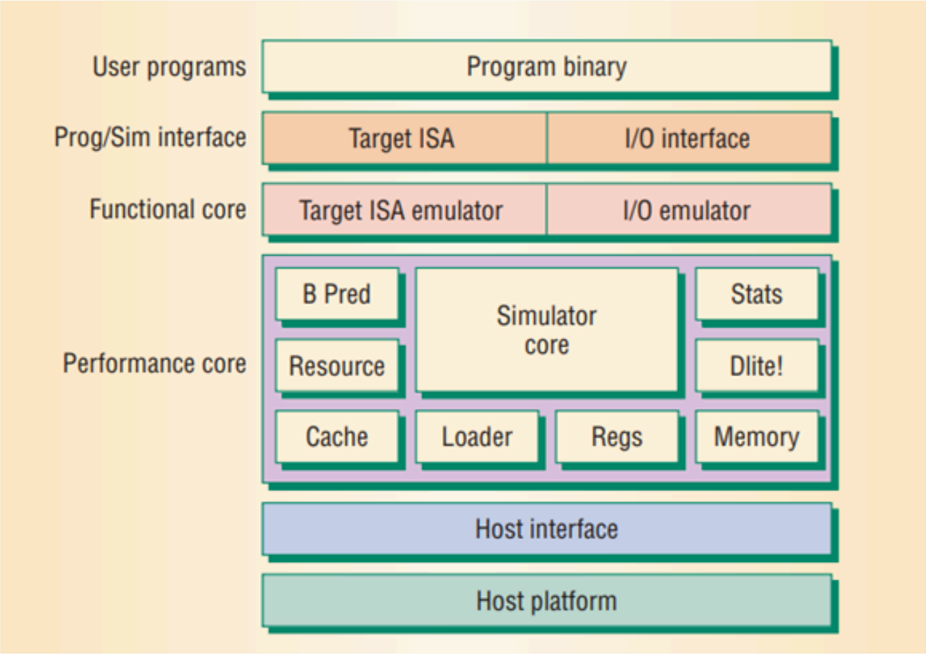


Figure : SimpleScalar Architecture

SimpleScalar Framework

In the domain of processor simulation, the SimpleScalar framework is highly regarded for its efficacy, especially in assessing ARM processors. It provides a comprehensive array of tools and libraries designed meticulously to streamline simulation processes, allowing researchers and developers to delve into performance evaluation, architectural analysis, and experimental inquiries with notable efficiency.

The figure shows the SimpleScalar architecture. SimpleScalar offers a comprehensive framework for processor simulation, integrating essential components like instruction set interpreters, memory hierarchy simulation, branch prediction mechanisms, and speculative execution. The core of the framework lies in its instruction set interpreter, which deciphers and executes instructions sourced from trace files. This interpreter interacts seamlessly with other critical modules responsible for managing processor states, executing memory operations, and handling branch instructions.

III Simulation Environmnent

Benchmark Type	Workload Mix	
Small Scale Benchmark	w1	Blackscholes, Freqmine, X264
	w2	Vips, Streamcluster, Freqmine
	w3	Canneal, Ferret, Fluidanimate
	w4	Swaption, Kmeans, Blackscholes
	w5	Vips, Canneal, Fluidanimate
Medium Scale Benchmark	w6	Fluidanimate, Streamcluster, X264
	w7	Ferret, Canneal, X264
	w8	Kmeans, Freqmine, Vips
	w9	Streamcluster, Ferret, Kmeans
	w10	Blackscholes, Fluidanimate, Swaption

Benchmarks and Workloads

In this simulation, we have combined the benchmarks and provided them to our configuration using Pin Tool. The workloads are achieved by combining the Parsec 3.0 and Splash 2x benchmarks, considering the benchmark size, computation utilization, and memory utilization. We have divided the workloads into:

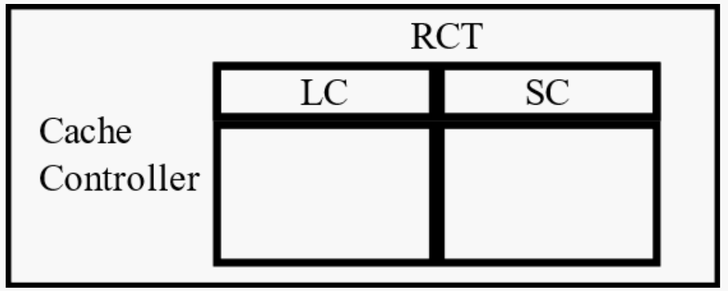
Table outlines the composition of small scale and medium scale benchmarks, each denoted by a workload mix identifier (w1 to w10). In the small scale benchmarks, workloads are grouped based on their application types and computational characteristics. For instance, w1 combines Blackscholes, Freqmine, and X264, while w2 includes Vips, Streamcluster, and Freqmine. Medium scale benchmarks follow a similar pattern, grouping different combinations of benchmarks to create workloads suitable for medium-scale simulations. Each workload mix represents a unique combination of benchmarks tailored to assess specific aspects of system performance.

IV Implementation and Results

Heterogeneity Sensible Replacement for Partitioned Cache(HSRPC)

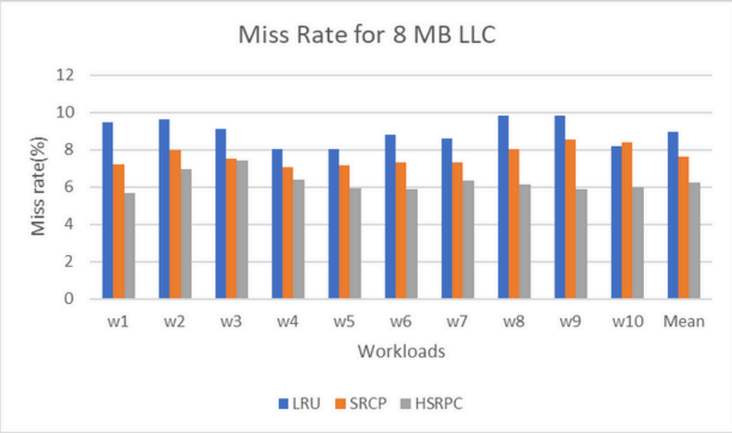
The HSRPC policy considers the access frequency of cache blocks and their reuse status tracked by the RCT. Cache blocks with higher reuse counts are given higher priority to be retained in the LLC. Lower priority blocks are selected for eviction based on their access frequency, optimizing cache utilization and performance. In the heterogeneous-aware partition cache replacement policy (HSRPC), the objective is to maintain the reuse count table (RCT) in a manner that accounts for the memory access behavior characteristics of the heterogeneous cores in an asymmetric multi-core architecture. Specifically, this policy aims to prioritize cache blocks reused by big cores over those reused by little cores, given the performance differences between the two types of cores.

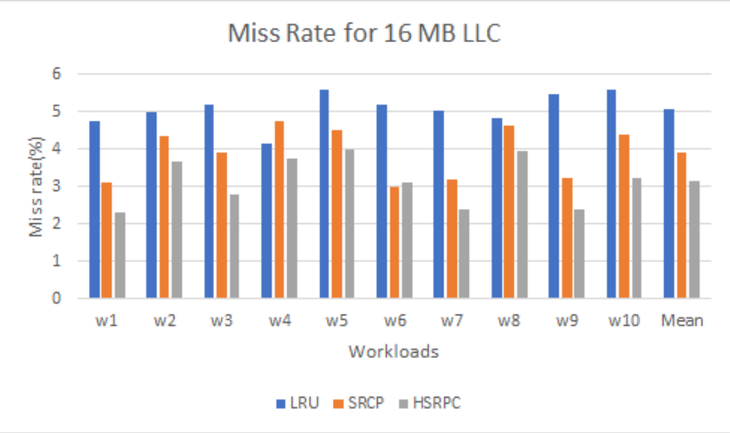
A Reuse Counting Table (RCT) is maintained for the LLC to track the historical reuse information of each cache block. The RCT records two main metrics: LC (Local Count) and SC (Shared Count). LC records the reuse count caused by local core fetch requests hitting the cache block, while SC records the reuse count caused by shared core memory fetch requests hitting the cache block.

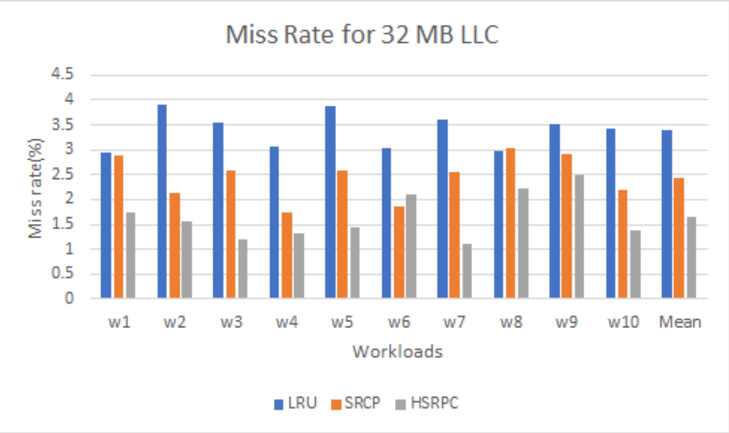


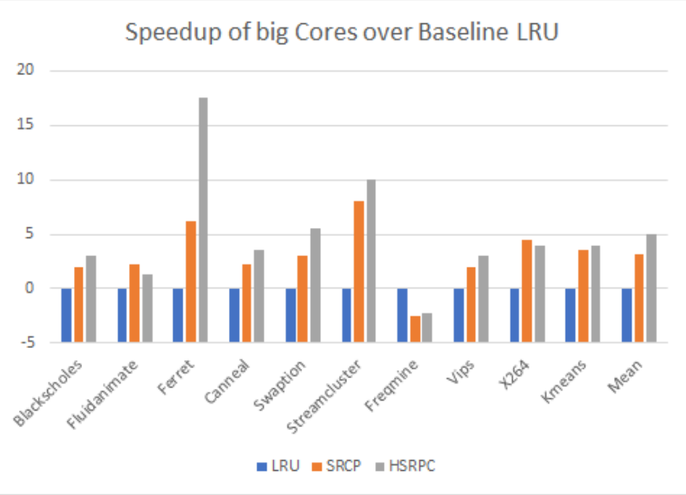
IV Implementation and Results

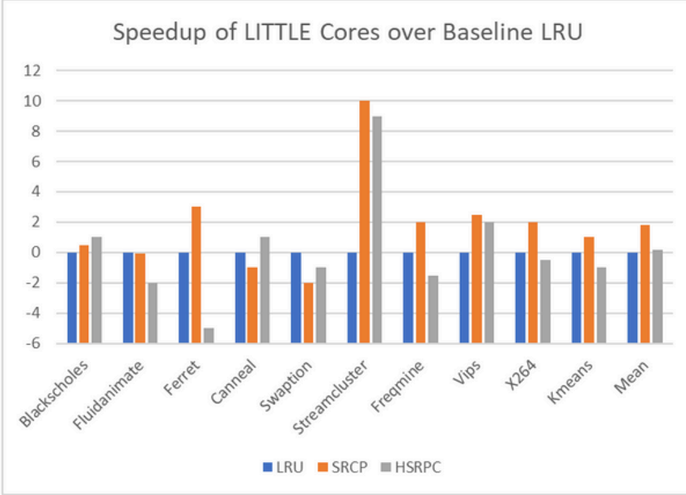
Results











IV Implementation and Results

Analysis

Observing the results across all LLC sizes, it is evident that HSRPC consistently outperforms both LRU and SRCP in terms of reducing cache miss rates. This improvement is particularly pronounced in smaller LLC sizes, where the cache capacity is more constrained. For instance, at the 8MB LLC size, HSRPC exhibits significantly lower miss rates compared to LRU and SRCP across all workload mixes, indicating its effectiveness in optimizing cache utilization and reducing cache thrashing.

Analyzing the results further, we observe that as the LLC size increases from 8MB to 32MB, the gap in miss rates between HSRPC and the other policies diminishes. This trend suggests that with larger cache capacities, the performance differences between the cache replacement policies become less pronounced. However, even at the largest LLC size of 32MB, HSRPC consistently maintains lower miss rates compared to LRU and SRCP, albeit with a smaller margin.

V Conclusion & Future Work

The evolution of ARM-based heterogeneous multi-core processors, coupled with innovative cache management schemes like Heterogeneity Sensible Replacement for Partitioned Cache (HSRPC), presents a paradigm shift in processor design, offering unparalleled flexibility and efficiency for modern computing applications. The big.LITTLE configuration, dynamic CPU clustering, and advanced cache management techniques enable these processors to seamlessly adapt to varying workload demands while minimizing energy consumption and improving system performance.

Expanding this study to mobile and embedded technologies is the next step. With the increasing overlap between mobile and IoT ecosystems, optimizing processor architectures becomes crucial. Adapting the proposed cache management scheme, HSRPC, to mobile and embedded systems could significantly enhance efficiency and performance. This future work aims to address evolving challenges and unlock new opportunities in mobile and embedded computing

References

[1] Arm® DynamIQ™ Shared Unit Technical Reference Manual, https://developer.arm.com/documentation/100453/latest/

[2] Arm® Cortex®-A55 Core Technical Reference Manual, https://developer.arm.com/documentation/100442/latest/

[3] Arm® Cortex® A76 Core Technical Reference Manual, https://developer.arm.com/documentation/100798/latest/

[4] Cortex-M Processors and the Internet of Things (IoT), https://community.arm.com/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-21-42/White-Paper_2D00_Cortex_2D00_M-Processors-_2600_-the-IoT.pdf

[5] Arm® Mali™-G71 Performance Counters Reference Guide, https://developer.arm.com/documentation/102641/0105

[6] Asghar, M. N. (2020). A review of ARM processor architecture history, progress and applications. Journal of Applied and Emerging Sciences, 10(2), pp-171.

[7] Zahid, Y., Khurshid, H., & Memon, Z. A. (2018). On improving efficiency and utilization of last level cache in multicore systems. Information Technology and Control, 47(3), 588-607.

[8] Mascitti, A., Cucinotta, T., & Marinoni, M. (2020). An adaptive, utilization-based approach to schedule real-time tasks for ARM big. LITTLE architectures. ACM SIGBED Review, 17(1), 18-23.

[9] Gollapudi, R. T., Yuksek, G., & Ghose, K. (2019, April). Cache-aware dynamic classification and scheduling for Linux. In 2019 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS) (pp. 1-3). IEEE.

[10] Cronin, P., Gao, X., Wang, H., & Cotton, C. (2021, December). An exploration of ARM system-level cache and GPU side channels. In Proceedings of the 37th Annual Computer Security Applications Conference (pp. 784-795).

[11] Zahran, M. (2007). Cache replacement policy revisited. WDDD held in conjunction with ISCA, 1-8.

[12] Aswathy, N. S., Raj, R. R., Jose, J., & Josna, V. R. (2017). Implementation and Analysis of Adaptive Packet Throttling in Mesh NoCs. Procedia computer science, 115, 626-634.

[13] Coutinho, D. A. M. (2021). Performance-energy trade-offs prediction and runtime selection for parallel applications on heterogeneous multiprocessing systems.

[14] Cortex-M Processors and the Internet of Things (IoT), https://community.arm.com/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-21-42/8233.White-Paper_2D00_Cortex_2D00_M-Processors-_2600_-the-IoT.pdf

[15] Qureshi, M. K., & Patt, Y. N. (2006, December). Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06) (pp. 423-432). IEEE.

[16] Jaleel, A., Theobald, K. B., Steely Jr, S. C., & Emer, J. (2010). High performance cache replacement using re-reference interval prediction (RRIP). ACM SIGARCH computer architecture news, 38(3), 60-71.

[17] Ghosh, S. N., Bhargava, L., & Sahula, V. (2021). SRCP: sharing and reuse-aware replacement policy for the partitioned cache in multicore systems. Design Automation for Embedded Systems, 25(3), 193-211.

[18] Fang, J., Kong, H., Yang, H., Xu, Y., & Cai, M. (2022). A Heterogeneity-Aware Replacement Policy for the Partitioned Cache on Asymmetric Multi-Core Architectures. Micromachines, 13(11), 2014.

[19] Zhan, X., Bao, Y., Bienia, C., & Li, K. (2017). PARSEC3. 0: A multicore benchmark suite with network stacks and SPLASH-2X. ACM SIGARCH Computer Architecture News, 44(5), 1-16.

[20] Luk, C. K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., ... & Hazelwood, K. (2005). Pin: building customized program analysis tools with dynamic instrumentation. Acm sigplan notices, 40(6), 190-200.

[21] Austin, T., Larson, E., & Ernst, D. (2002). SimpleScalar: An infrastructure for computer system modeling. Computer, 35(2), 59-67.

Thank you for your time!