Investigating Performance and Real-Time Trade-offs in Out-of-Order Processors

Wenhao Sun, Yuhui Hao, Yiyang Huang
Institute of Computing Technologies, Chinese Academy of Sciences, Beijing China
Corresponding: sunwenhao23s@ict.ac.cn
Accepted by ISCA 2025 Workshop: Workshop on Architecture Support for Embodied AI Systems

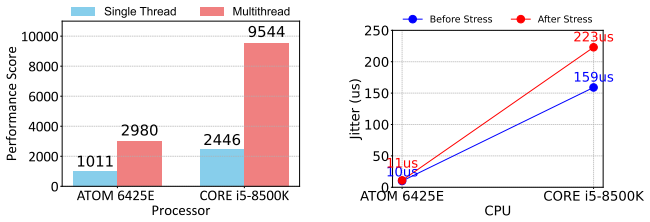
***Abstract*—Embodied AI systems necessitate both high computational performance for complex perception and planning tasks and stringent real-time control for reliable physical interaction. This paper investigates the inherent trade-offs between maximizing performance throughput and guaranteeing deterministic, low-latency responses in out-of-order processors commonly employed in embodied AI systems. Drawing insights from industrial automation, we analyze the performance and real-time characteristics of representative high-performance and embedded processors. We examine how key microarchitectural features—out-of-order execution, branch prediction, and the last level cache—impact on performance and real-time. Our analysis indicates that while complex microarchitectures enhance average performance, they often increase real-time variability, underscoring the need for balanced processor designs in embodied AI applications.**

I. Introduction

Sophisticated computational capabilities are fundamental to embodied AI systems, such as humanoid robots and autonomous agents, enabling them to perceive the environment, make complex decisions, and interact with the physical world. [1]. Embodied AI systems demand both high computational performance for tasks like model inference and stringent real-time performance for precise and deterministic control of actuators and effectors [2]. This dual requirement creates a significant challenge for processor design, as traditional performance optimizations often conflict with the need for predictable and low-latency execution necessary for real-time guarantees. This study theorily investigates the fundamental influence and trade-off in out-of-order processors for embodied AI systems, by analyzing the impact of microarchitectural features on performance and real-time properties, while identifying directions for future research.

II. Motivation

Embodied AI systems necessitate high computational performance for tasks such as perception and planning (e.g., VLA model inference) and stringent real-time control for physical interaction (e.g., precise robotic actuation). On power and space-limited edge devices, these computationally intensive inference workloads often fall to the CPU. Real-time systems are defined by their ability to respond to events within strict time constraints. For hard real-time tasks in embodied AI, such as collision avoidance, missing a deadline is unacceptable and can lead to catastrophic failures.



(a) Performance under PassMark. (b) Jitter before and after stress.

Fig. 1: Comparison between ATOM and Core I5.

In embodied AI tasks, performance focuses on computational throughput, measured in terms of IPC or benchmark scores like CoreMark. However, performance optimization often conflicts with real-time guarantees, requiring careful processor design.

In contrast, real-time performance is quantified by latency and jitter. Latency is the time difference between event occurrence (t_e) and system response completion (t_r):

$$\text{Latency} = t_r - t_e \tag{1}$$

Jitter is defined as the scientific average of latency measurements, where L_imax is the i-th cycle max latency measurement, L_imin is the i-th cycle minimum latency, and n is the number of measurements:

$$\text{Jitter} = \frac{1}{n}\sum_{i=1}^{n}(L_imax - L_imin) \tag{2}$$

This tension is evident when comparing processors. High-performance processors (e.g., Intel Core i5) maximize throughput but introduce variability, while low-power embedded processors (e.g., Intel Atom 6425E) prioritize predictable execution but may lack throughput for demanding AI tasks. As Figure 1 illustrates, an Atom x6425E achieves low jitter (50 μs) for motor control but limited IPC performance, whereas the Core i5-8500K offers high throughput but significantly higher jitter (exceeding 200 μs), potentially resulting in real-time failures. **This highlights the critical need to balance performance and real-time guarantees for embodied AI platforms.**

III. Microarchitectural Factors and Their Impact

To understand the performance and real-time differences between processors like the Intel Core i5 and Atom, examining

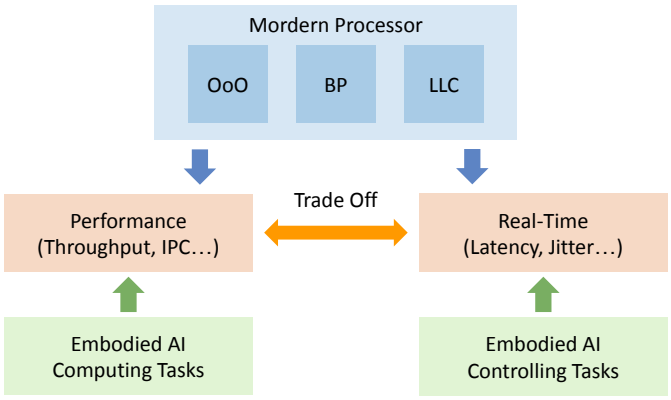


Fig. 2: Influence of modern features on embodied AI tasks.

their microarchitectures is necessary. Although both are out-of-order designs, they have key differences prioritizing distinct objectives. As Figure 2 illustrates, these modern features include out-of-order execution (OoO), branch prediction (BP), and the last-level cache (LLC).

A. Out-of-Order Execution

Out-of-order (OoO) execution is a fundamental technique in modern processors that improves performance by allowing instructions to execute in an order different from the program sequence while respecting data dependencies. Using mechanisms like reorder buffers (ROBs) and instruction queues, it exploits instruction-level parallelism to increase throughput. For computationally intensive tasks in embodied intelligence, effective OoO execution can significantly reduce overall execution time.

However, the dynamic nature of OoO execution poses challenges for real-time systems. The actual completion time of an instruction or task becomes dependent on the availability of execution units, the state of the ROBs, and complex interactions with other instructions in flight. This variability makes it difficult to precisely predict the Worst-Case Execution Time (WCET), a critical metric for guaranteeing real-time deadlines. Processors with larger ROBs and wider issue widths, typical of high-performance designs can exploit more parallelism but may also exhibit greater variability in execution times compared to simpler designs like the Atom, which features a smaller ROB and narrower pipeline.

B. Branch Prediction

Branch prediction (BP) is a key performance technique in pipelined processors. It predicts conditional branch outcomes, allowing speculative execution along anticipated paths to minimize stalls. Accurate BP is vital for maintaining high throughput in code with complex control flow, common in perception and planning algorithms.

Nevertheless, branch prediction errors incur a significant penalty: the pipeline must be flushed, and the correct instructions fetched, leading to wasted cycles and increased execution time. This penalty is amplified in processors with deeper and wider pipelines, as more speculative work is discarded. For real-time tasks, branch mispredictions introduce unpredictable delays, increasing jitter and making it harder to guarantee timely responses. While high-performance processors like the Core i5 employ sophisticated branch predictors with larger brunch target buffer (BTB) to improve accuracy on average, even rare mispredictions can have a significant impact on WCET. Simpler, more predictable branch predictors, even if less accurate on average, might be preferable for hard real-time components.

C. Last Level Cache

The Last Level Cache (LLC), typically the L3 cache, is a shared resource designed to reduce average memory access latency and enhance performance by storing frequently accessed data. Larger caches improve hit rates, significantly boosting throughput for data-intensive embodied AI tasks. For instance, the Intel Core i5-8500's 9MB L3 cache contributes to higher performance compared to the Intel Atom x6425RE's smaller 1.5MB L2 cache (its LLC).

However, LLC behavior introduces complexities for real-time systems. Accessing data not present in the cache (a cache miss) results in a significantly longer latency to fetch data from main memory. The unpredictable nature of cache misses, influenced by access patterns and contention in multi-core systems, introduces variability in instruction execution times. Furthermore, in multi-core or multi-threaded scenarios, different tasks or threads compete for space within the shared LLC. This cache contention can lead to cache lines belonging to a real-time task being evicted by a non-real-time task, causing subsequent cache misses and increased latency for the real-time task. This resource competition makes it challenging to precisely estimate the WCET and increases jitter, negatively impacting real-time guarantees. Processors with larger, more complex cache hierarchies, while offering higher average performance, can exhibit greater real-time unpredictability compared to those with simpler cache designs.

IV. Conclusion

Embodied AI systems require processors balancing high performance for intelligent tasks with robust real-time control. Our analysis highlights the inherent influence and trade-offs in out-of-order design: features like OoO, BP, and LLC boost throughput but complicate real-time guarantees by introducing variability. Performance-optimized processors often show greater real-time unpredictability than those prioritizing efficiency. Future research should explore with detail data that how these features affect performance and real-time, with optimizing processor architectures and system-level techniques to mitigate these trade-offs, enabling balanced designs for the next generation of embodied AI systems.

References

[1] W. Sun, S. Hou, Z. Wang, B. Yu, S. Liu, X. Yang, S. Liang, Y. Gan, and Y. Han, "Dadu-e: Rethinking the role of large language model in robotic computing pipeline," *arXiv preprint arXiv:2412.01663*, 2024.

[2] Y. Huang, Y. Hao, B. Yu, F. Yan, Y. Yang, F. Min, Y. Han, L. Ma, S. Liu, Q. Liu *et al.*, "Corki: Enabling real-time embodied ai robots via algorithm-architecture co-design," *arXiv preprint arXiv:2407.04292*, 2024.