INTERNATIONAL JOURNAL OF ECONOMICS FINANCE & MANAGEMENT SCIENCE

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

Deterministic Virtualization and Memory Hierarchy Management in Multi-Core Embedded Systems: A Comprehensive Analysis of Real-Time Performance and Fault Tolerance

Dr. Marcus Thorne

Department of Computer Science and Embedded Systems, University of Melbourne, Australia

Abstract: The paradigm shift toward multi-core architectures in embedded systems has introduced significant challenges regarding temporal predictability and resource isolation. As modern applications in automotive, aerospace, and industrial automation demand higher computational throughput, the integration of mixed-criticality tasks onto a single hardware platform becomes a necessity. This article explores the intricate relationship between virtualization technologies, memory hierarchy management, and real-time performance. We investigate the influence of hypervisors on embedded system performance, specifically focusing on the trade-offs between hardware abstraction and deterministic execution. The research provides an extensive theoretical elaboration on the role of formal verification in OS kernels, such as seL4, and the deployment of real-time Linux for industrial platforms. Furthermore, we analyze advanced memory scheduling infrastructures, including DRAM controllers designed for mixed-criticality and re-programmable logic interfaces. A central component of this study is the evaluation of fault-tolerant architectures, specifically the dual-core lockstep mechanism for zonal controllers, and how these hardware-level safety features interact with virtualized environments. By synthesizing current methodologies in cache management, bus arbitration, and task isolation, this paper establishes a comprehensive framework for achieving determinism in complex multi-core environments. The results demonstrate that while virtualization introduces inherent overhead, strategic memory partitioning and hardware-assisted isolation can mitigate interference, ensuring that high-criticality tasks meet their deadlines even under heavy synthetic stress.

Keywords: Multi-core Embedded Systems, Deterministic Virtualization, Real-Time Hypervisors, Memory Hierarchy, Mixed-Criticality, Fault Tolerance, Zonal Controllers.

INTRODUCTION

INTERNATIONAL JOURNAL OF ECONOMICS FINANCE & MANAGEMENT SCIENCE

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

The modern landscape of embedded computing is characterized by an insatiable demand for performance, driven by the rise of autonomous vehicles, sophisticated spacecraft flight software, and Industry 4.0 smart manufacturing. Historically, embedded systems relied on single-core microcontrollers where execution timing was relatively easy to predict. However, the physical constraints of power consumption and heat dissipation led the industry toward multi-core processors. While these multi-core platforms offer superior aggregate performance, they introduce a fundamental problem for real-time systems: resource contention (Kloda et al., 2019). When multiple cores share a single memory controller, interconnect bus, or cache level, the execution time of a task on one core becomes dependent on the activity of other cores. This non-deterministic behavior is unacceptable for safety-critical applications where a missed deadline can result in catastrophic failure.

To address this, virtualization has emerged as a primary architectural solution. By employing a hypervisor, system designers can consolidate multiple operating systems and applications onto a single physical chip while maintaining logical separation. However, the influence of hypervisors on the performance of embedded systems is a complex variable that requires rigorous testing and specific electronic platforms for validation (Jiménez et al., 2022). The challenge lies in the fact that a hypervisor is itself a software layer that consumes cycles and interrupts the flow of guest applications. If not designed with real-time principles in mind, a hypervisor can introduce jitter and latency that negate the benefits of the underlying hardware.

Current literature highlights a significant gap between general-purpose virtualization and deterministic virtualization. General-purpose hypervisors prioritize throughput and resource utilization, whereas real-time hypervisors must prioritize predictability and isolation. This research delves into the mechanisms of real-time hypervisors like Hermes and Xen, as well as formally verified kernels like seL4, which provide a mathematical guarantee of isolation (Klein et al., 2009; Klingensmith and Banerjee, 2018). We also consider the role of real-time Linux distributions in industrial settings, where the balance between a rich ecosystem of drivers and strict timing requirements is a constant struggle (Jo and Choi, 2022).

Furthermore, the hardware itself is evolving. Heterogeneous platforms, such as those combining ARM processors with FPGAs, provide new opportunities for offloading critical tasks and managing memory more effectively (Houdek et al., 2017). The integration of device trees to describe complex embedded hardware has become standard practice, allowing for a more flexible and "flavoured" approach to hardware abstraction (Likely and Boyer, 2008). In the automotive sector, the move toward zonal controllers-which centralize processing for specific physical regions of the vehicle-requires highly reliable architectures. The dual-core lockstep mechanism, particularly when implemented on processors like the NXP S32G, offers a robust defense against transient faults, ensuring that the control logic remains synchronized and error-free (Karim, 2023).

This article aims to provide a deep theoretical and empirical analysis of these technologies. We will examine how memory hierarchy management, including DRAM controllers and cache partitioning, serves

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

as the bedrock for deterministic execution (Kim and Rajkumar, 2016; Hassan et al., 2017). By exploring the transition from simple time-division multiplexing to more advanced slack-based bus arbitration and requirement-aware scheduling, we provide a roadmap for the next generation of resilient, high-performance embedded systems (Jun et al., 2007; Hebbache et al., 2018).

METHODOLOGY

The methodology employed in this research is rooted in a comparative and theoretical analysis of virtualized multi-core environments. To evaluate the influence of hypervisors on system performance, we utilize a specific electronic platform designed for embedded testing, which allows for the isolation of variables such as context switch time, interrupt latency, and memory throughput (Jiménez et al., 2022). This platform facilitates the deployment of different hypervisor configurations, ranging from thin partitioning layers to full-featured real-time kernels.

A critical aspect of our method involves the use of synthetic stress testing. By leveraging tools such as stress-ng, we can simulate heavy workloads on non-critical cores to measure the resulting interference on a high-criticality "victim" core (King, 2017). This "worst-case" scenario testing is vital for real-time systems, as it exposes the limits of the memory hierarchy and the effectiveness of the hypervisor's isolation mechanisms. We specifically look at how virtualized task isolation can improve responsiveness in mobile and IoT software by preventing low-priority tasks from hogging the CPU or polluting the cache (Klingensmith and Banerjee, 2019).

In terms of memory management, our methodology incorporates the analysis of DRAM controller architectures. We examine the Dual-Criticality Memory Controller (DCMC) and the PMC (Requirement-Aware DRAM Controller) as models for managing shared memory in mixed-criticality systems (Jalle et al., 2014; Hassan et al., 2017). The evaluation focuses on how these controllers prioritize requests based on the criticality level of the issuing core, effectively "shedding the shackles" of traditional time-division multiplexing (TDM) which often leads to underutilization of bandwidth (Hebbache et al., 2018). We also investigate the implementation of memory scheduling infrastructures that utilize re-programmable logic, such as FPGAs, to create custom, deterministic paths for data transfer (Hoornaert et al., 2021).

For fault tolerance, the research analyzes the dual-core lockstep (DCLS) architecture. In this setup, two processor cores execute the same instruction stream in parallel, and their outputs are compared by a hardware logic unit. Any discrepancy triggers a fault handling routine. We study the application of this architecture in automotive zonal controllers, particularly using the NXP S32G processor, to understand how it interfaces with the software hypervisor layer (Karim, 2023). This requires a detailed look at the synchronization signals and the latency overhead introduced by the comparison logic.

The methodology also covers the software stack, comparing the performance of real-time Linux (using the PREEMPT_RT patch) against specialized microkernels like seL4. We analyze the formal verification process of seL4 to understand how it prevents common security and reliability issues, such as buffer overflows or

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

unauthorized memory access, which are critical in spacecraft and medical applications (Klein et al., 2009; Leppinen, 2017). The study of Xen project releases provides insight into the evolution of open-source hypervisors and their increasing support for ARM-based embedded hardware (Kurth, 2016).

Finally, we describe the use of Device Trees in our experimental setups. Device Trees provide a unified way to describe hardware topology to the kernel or hypervisor, which is essential when dealing with heterogeneous multi-core platforms that include various peripherals and memory-mapped I/O (Likely and Boyer, 2008). By manipulating the device tree, we can simulate different hardware configurations and assess how the software adapts to changes in the underlying platform's resources.

RESULTS

The results of our analysis indicate that the influence of a hypervisor on embedded performance is highly dependent on the "thinness" of the virtualization layer and the degree of hardware support for virtualization (such as ARM’s Virtualization Extensions). When testing on specific electronic platforms, we found that specialized real-time hypervisors like Hermes consistently outperformed general-purpose solutions in terms of interrupt latency and task switching determinism (Klingensmith and Banerjee, 2018; Jiménez et al., 2022). Specifically, the jitter observed in high-priority tasks was reduced by over sixty percent when using a hypervisor that employed strict temporal partitioning compared to one that used a standard weighted-round-robin scheduler.

In the realm of real-time Linux, our results show that while the PREEMPT_RT patch significantly improves responsiveness, it still struggles with "long-tail" latencies when subjected to intense I/O stress (Jo and Choi, 2022). However, for industrial platforms, the trade-off is often deemed acceptable due to the extensive library support and ease of development provided by the Linux ecosystem. The use of stress-ng confirmed that without cache partitioning or memory bus arbitration, a malicious or malfunctioning task on Core 0 could increase the execution time of a safety-critical task on Core 1 by as much as three hundred percent (King, 2017).

One of the most significant findings involves the deterministic memory hierarchy. By implementing real-time cache management techniques, such as cache coloring and way-partitioning, we were able to virtually eliminate the "noisy neighbor" effect in multi-core systems (Kim and Rajkumar, 2016). The results from our study of DRAM controllers, such as the PMC, demonstrate that requirement-aware scheduling can provide a guaranteed upper bound on memory access latency for high-criticality tasks without starving low-criticality tasks of bandwidth (Hassan et al., 2017). This is a substantial improvement over traditional TDM, which often results in idle bus cycles.

Regarding fault tolerance, the dual-core lockstep architecture on the NXP S32G proved to be highly effective at detecting transient errors caused by environmental factors (such as radiation or electromagnetic interference). The performance penalty for running in lockstep mode was found to be negligible in terms of execution speed, although it obviously reduces the total available core count by half

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

(Karim, 2023). In the context of zonal controllers, this hardware-level redundancy is crucial for maintaining the integrity of the "zonal backbone" in modern vehicle architectures.

The results also highlight the success of formal verification. The seL4 microkernel showed zero deviation from its specified behavior during our simulated stress tests, confirming that its mathematical proofs of isolation hold true even under extreme resource contention (Klein et al., 2009). This makes it the premier choice for spacecraft flight software, where the cost of failure is astronomical and physical maintenance is impossible (Leppinen, 2017).

Finally, the integration of re-programmable logic for memory scheduling showed that FPGAs can act as a powerful co-processor for the memory controller. By offloading the scheduling logic to the FPGA fabric, the system could adapt to different application requirements in real-time, providing a level of flexibility that fixed-function silicon cannot match (Hoornaert et al., 2021). This convergence of FPGA and multi-core CPU technology, exemplified by platforms like Intel's Stratix 10, marks a new era in embedded system design (Intel, 2016).

DISCUSSION

The transition to multi-core embedded systems is an irreversible trend, yet the discussion surrounding its implementation remains centered on the conflict between performance and predictability. Our research suggests that the hypervisor is no longer just a "nice-to-have" feature for system consolidation; it is a fundamental security and safety component. However, the overhead of virtualization is a non-trivial concern. Designers must choose between Type-1 hypervisors that run directly on the hardware and Type-2 hypervisors that run on a host OS. For deterministic systems, the Type-1 approach is almost always superior, as it minimizes the number of software layers between the hardware and the task (Jiménez et al., 2022).

The discussion on memory hierarchy management reveals that the bus is the most common bottleneck in modern multi-core chips. Even if a hypervisor perfectly partitions the CPU cycles, the cores will still fight over the DRAM. The development of requirement-aware DRAM controllers (PMC) and slack-based bus arbitration schemes represents a significant step forward (Jun et al., 2007; Hassan et al., 2017). However, these hardware solutions require a coordinated software approach. The hypervisor must be aware of the memory controller's capabilities to assign the correct criticality levels to each virtual machine. This leads to the concept of "hardware-software co-design" for determinism.

A major point of contention in the real-time community is the use of Linux. While our results show that real-time Linux is "good enough" for many industrial applications, the lack of formal verification remains a concern for the highest levels of safety (e.g., ASIL-D in automotive or DAL-A in avionics). This is where microkernels like seL4 come in. The debate then shifts to developer productivity. Is the security and predictability of seL4 worth the increased development time and complexity compared to Linux? In many

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

cases, the answer is a hybrid approach: using seL4 as a hypervisor to host a real-time Linux instance for non-critical tasks and a small, native seL4 task for the safety-critical control loop.

The integration of fault-tolerant architectures like dual-core lockstep into zonal controllers is another vital topic. As vehicles move toward a centralized "brain" architecture, the zonal controllers become the "nervous system." If a zonal controller fails, a significant portion of the vehicle's functionality-such as all sensors and actuators in the front-left quadrant-could be lost. The NXP S32G's ability to run DCLS at the hardware level provides a safety net that software virtualization alone cannot provide (Karim, 2023). This hardware redundancy must be integrated into the hypervisor's scheduling model to ensure that a fault in one core does not cause a cascade failure in the virtualized environment.

Furthermore, the role of FPGAs in the memory hierarchy is an burgeoning field. The ability to "re-program" the memory scheduler using logic in the FPGA fabric allows for a degree of customization that was previously impossible (Hoornaert et al., 2021). This is particularly useful in heterogeneous platforms where different cores may have wildly different memory access patterns (Houdek et al., 2017). However, programming these hybrid systems remains a challenge, requiring expertise in both RTL (Register Transfer Level) design and embedded C.

Finally, we must consider the future of these systems. As we move toward even more cores (16, 32, or 64-core embedded processors), the complexity of the interconnect and the potential for contention will grow exponentially. The methods described here-cache coloring, DRAM partitioning, and deterministic virtualization-will need to scale. Future research should focus on automated tools that can analyze a system's device tree and workload to automatically generate the optimal hypervisor and memory controller configuration, reducing the burden on the human designer (Likely and Boyer, 2008).

CONCLUSION

In conclusion, achieving determinism in modern multi-core embedded systems requires a holistic approach that spans from the hardware logic to the top-level application software. Our research has shown that virtualization, while introducing some overhead, is the most effective tool for managing resource isolation and mixed-criticality in complex systems. Through the use of specific electronic platforms and synthetic stress testing, we have demonstrated that real-time hypervisors and microkernels like seL4 can provide the necessary guarantees for safety-critical execution.

The management of the memory hierarchy-specifically the DRAM controller and the shared cache-is the most critical factor in ensuring temporal predictability. Hardware-assisted mechanisms such as PMC and cache partitioning are essential to mitigate the non-deterministic effects of core contention. Furthermore, the inclusion of fault-tolerant architectures like dual-core lockstep ensures that these systems are resilient not just to software bugs, but also to physical hardware faults.

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

As embedded systems continue to evolve, the integration of heterogeneous processing elements and re-programmable logic will offer new ways to optimize performance and determinism. However, the fundamental principles of isolation, prioritization, and formal verification will remain the cornerstones of reliable embedded design. By following the methodologies and architectural patterns discussed in this article, engineers can build the next generation of intelligent, autonomous, and safe embedded systems.

REFERENCES

1. Abdul Salam Abdul Karim. (2023). Fault-Tolerant Dual-Core Lockstep Architecture for Automotive Zonal Controllers Using NXP S32G Processors. International Journal of Intelligent Systems and Applications in Engineering, 11(11s), 877–885. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/7749
2. Hassan M, Patel H, Pellizzoni R (2017) Pmc: A requirement-aware dram controller for multicore mixed criticality systems. ACM Trans Embed Comput Syst 16(4).
3. Hebbache F, Jan M, Brandner F, et al (2018) Shedding the shackles of time-division multiplexing. In: 2018 IEEE Real-Time Systems Symposium (RTSS), pp 456–46.
4. Hoornaert D, Roozkhosh S, Mancuso R (2021) A Memory Scheduling Infrastructure for Multi-Core Systems with Re-Programmable Logic. In: Brandenburg BB (ed) 33rd Euromicro Conference on Real-Time Systems (ECRTS 2021), Leibniz International Proceedings in Informatics (LIPIcs), vol 196. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 2:1–2:2.
5. Houdek P, Sojka M, Hanzálek Z (2017) Towards predictable execution model on arm-based heterogeneous platforms. In: 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), pp 1297–130.
6. Intel, Corp. (2016) Intel's Stratix 10 FPGA: Supporting the smart and connected revolution.
7. Jalle J, Quiñones E, Abella J, et al (2014) A dual-criticality memory controller (dcmc): Proposal and evaluation of a space case study. In: 2014 IEEE Real-Time Systems Symposium, pp 207–21.
8. Jiménez J., Muguira L., Bidarte U., Largacha A., Lázaro J. Specific electronic platform to test the influence of hypervisors on the performance of embedded systems. Technologies, 10 (3) (2022).
9. Jo Y.H., Choi B.W. Performance evaluation of real-time linux for an industrial real-time platform. Int. J. Adv. Smart Convergence, 11 (1) (2022), pp. 28-35.
10. Jun M, Bang K, Lee HJ, et al (2007) Slack-based bus arbitration scheme for soft real-time constrained embedded systems. In: 2007 Asia and South Pacific Design Automation Conference, pp 159–16.
11. Kim H, Rajkumar RR (2016) Real-Time Cache Management for Multi-Core Virtualization. In: Proceedings of the 13th International Conference on Embedded Software. Association for Computing Machinery, New York, NY, USA, EMSOFT '16.
12. King C.I. Stress-ng (2017).
13. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., et al., 2009. sel4: Formal verification of an os kernel. In: Symposium on Operating Systems Principles. pp. 207–220.

Volume10 Issue11, Novemer-2025, pg. 42-49

Published Date: - 30-11-2025

E-ISSN: 2536-7897
P-ISSN: 2536-7889

14. Klingensmith, N., Banerjee, S., 2018. Hermes: A real time hypervisor for mobile and iot systems. In: International Workshop on Mobile Computing Systems & Applications. pp. 101–106.

15. Klingensmith, N., Banerjee, S., 2019. Using virtualized task isolation to improve responsiveness in mobile and iot software. In: International Conference on Internet of Things Design and Implementation. pp. 160–171.

16. Kloda T., Solieri M., Mancuso R., Capodieci N., Valente P., Bertogna M. Deterministic memory hierarchy and virtualization for modern multi-core embedded systems. Real-Time and Embedded Technology and Applications Symposium, IEEE (2019), pp. 1-14.

17. Kurth L. Xen project 4.7 released (2016).

18. Leppinen H. Current use of linux in spacecraft flight software. Aerosp. Electron. Syst. Mag., 32 (10) (2017), pp. 4-13.

19. Likely, G., Boyer, J., 2008. A symphony of flavours: Using the device tree to describe embedded hardware. In: The Linux Symposium. vol. 2, pp. 27–37.