Work-in-Progress: Toward Real-Time Cross-ISA Execution on the AMD Embedded+ Architecture

Lukas Neef*, Daniele Ottaviano*, Denis Hoornaert*, Alexander Zuepke*†, Marco Caccamo*, Andrea Bastoni*†
*Technical University of Munich, †Minerva Systems
{lukas.neef, daniele.ottaviano, denis.hoornaert, alex.zuepke, mcaccamo, andrea.bastoni}@tum.de

***Abstract*—Emerging embedded platforms increasingly rely on heterogeneous processing units to address diverse performance and energy requirements. The recently introduced AMD Embedded+ architecture reflects this trend by interconnecting via PCIe on the same motherboard one AMD x86 host processor with one Arm AArch64+FPGA complex. This implementation is another step forward towards a more compact heterogeneous-ISA platform designed with embedded applications in mind. While cross-ISA execution has been explored in the past with a focus on performance, programmability, and energy efficiency, its potential for embedded and predictable real-time workloads remains largely unexplored. In this paper, we start exploring such potential by investigating the real-time capabilities of the first commercial platform based on the AMD Embedded+ architecture: the Sapphire Edge+. We (1) outline key research challenges and real-time use-cases, (2) discuss suitable software architectures for the use-cases and highlight associated trade-offs, and (3) report an initial assessment of the potential of such architectures and use-cases via an experimental evaluation of latency and bandwidth on the real hardware.**

***Index Terms*—real-time system, heterogeneous architectures, cross-ISA execution, multi-core, virtualization**

I. Introduction

Prompted by an increasing demand for efficient computing systems, heterogeneous computing has become an ubiquitous design principle in embedded and cyber-physical systems. In fact, modern platforms combine general-purpose CPU cores, microcontroller-class CPUs, and specialized accelerators such as GPUs, NPUs, or FPGAs. Programming such systems is challenging due to the diverse nature of the processing units composing them. In fact, each unit can have partially overlapping or completely disjoint instruction set architectures (ISAs). In the current computing landscape, the combination of x86-based and Arm AArch64-based processors is an obvious target for manufacturers and system integrators due to their respective focus on raw performance and power efficiency.

Co-designing for such systems creates opportunities for flexible, energy-aware execution, but also raises challenges in programmability, scheduling, and predictability. Over the past decade, a sizeable amount of research has sought to address these challenges through cross-ISA execution. Projects such as Popcorn Linux [1], HEXO [2], and Stramash [3] have demonstrated the benefits of mechanisms such as process migration, VM portability, and cache-coherent execution across different ISAs. Yet, these works focus on power-hungry server-class platforms (interconnected via PCIe [1] or Ethernet [2]) found in cloud infrastructures and optimized exclusively for programmability and throughput.

Marco Caccamo was supported by an Alexander von Humboldt Professorship endowed by the German Federal Ministry of Education and Research.

The recent release of the AMD Embedded+ architecture [4] provides the opportunity to revisit the above mentioned challenges from an *embedded real-time* perspective. The Sapphire Edge+ VPR-4616-MB [5], the first COTS platform based on the Embedded+ architecture, integrates an x86-based AMD Ryzen Embedded CPU and an Arm AArch64-based Versal FPGA+CPU complex on a single motherboard through on-board PCIe (see Fig. 1). It stands out in two ways. First, unlike prior cross-ISA x86–AArch64 platforms limited to datacenter-class systems, it directly addresses size, weight, and power (SWaP) constrained use cases via a compact and low-power embedded form factor. Second, unlike earlier embedded boards that pair x86 with microcontroller-class CPUs (*e.g.*, 32-bit Arm Cortex-M/R and RISC-V), it features fully fledged 64-bit application cores (Cortex-A72) that can host OS/hypervisors and run complex workloads in parallel with accelerators. These qualities make the Embedded+ architecture a perfect fit for mixed-criticality systems such as those found in, *e.g.*, automotive or industrial domains.

In the default configuration, the x86 subsystem acts as the *host* while the Versal is essentially exposed as a subordinate accelerator connected as PCIe *device*, with the Arm cores supporting auxiliary functions. However, we argue that treating both subsystems as first-class computing elements—capable of hosting real-time workloads, supporting task migration, and coordinating accelerator utilization—would help improve system predictability and energy efficiency. This view is also reflected in recent shifts in the, *e.g.*, safety-critical automotive domain, where heterogeneous-ISAs have been highlighted as a promising enabler to consolidate mixed-criticality workloads, while preserving real-time isolation [6].

Heterogeneous-ISA platforms such as the Embedded+ offer a timely opportunity for the real-time systems community to explore suitable execution models. However, they raise questions on system practicality, including bounded migration latency, deterministic accelerator scheduling, and predictable resource isolation. We make three early contributions:

- A framing of the real-time research challenges introduced by cross-ISA platforms in an embedded form factor.
- An overview of suitable software architectures for cross-ISA execution in embedded contexts and their trade-offs.
- Preliminary microbenchmarks of the PCIe connection on the Edge+ quantifying communication bandwidth and latency characteristics relevant to process migration.

II. Related Works

Multikernel Operating Systems. Multikernels run multiple OS kernel instances per core or groups of cores [7], and enable process and thread migration even across heterogeneous ISAs.

Popcorn Linux [1] provides a replicated-kernel OS and a software distributed shared memory (DSM) abstraction for servers connected via off-chip, board-to-board interconnects such as PCIe. It dynamically migrates execution to optimize performance without requiring application source code changes. DAPPER [8] introduces live process rewriting for low-latency cross-ISA migration, emphasizing security through randomized program state. Stramash [3] follows the direction of emerging CXL-based server interconnects and extends the idea of Popcorn by implementing, via full-system emulation, a cache-coherent shared memory across ISAs aimed at reducing message-passing overhead and improving synchronization. However, real-time predictability and isolation guarantees are not the primary concern in both cases.

Virtual Machine and Container Migration. Virtual machine (VM) and container migration have also been investigated as mechanisms to consolidate workloads and improve utilization in heterogeneous environments. HEXO [2] achieves semantic unikernel (see *e.g.*, [9]) VM migration between server-class and embedded Arm boards, typically networked via Ethernet, using lightweight unikernels and memory disaggregation to overcome resource constraints. H-Container [10] enables server-to-edge container migration to heterogeneous ISAs with minimal runtime overhead by transforming checkpoints and recompiling binaries across ISAs without kernel or hypervisor modifications. These systems demonstrate flexible migration of stateful applications, but do not provide or evaluate the bounded worst-case latencies required by real-time workloads.

Existing approaches focus on throughput, average migration overhead, and ease of programming, often assuming datacenter setups interconnected via Ethernet or PCIe. In these works, real-time properties (*e.g.*, bounded communication latency, deterministic scheduling, and mixed-criticality isolation) remain unexplored.

The AMD Embedded+ enables the evaluation of heterogeneous-ISA architectures for real-time use cases.

III. AMD Embedded+ on Sapphire Edge+

Our evaluation platform is the Edge+ [5]. It integrates an AMD Ryzen Embedded SoC (*x86 host*) and a Versal VE2302 Adaptive SoC (*AArch64+FPGA device*), interconnected via a four-lane PCIe Gen3 link (up to 4 GB/s). As shown in Fig. 1, both SoCs can run a full software stack (OS and hypervisor) and exchange data over UART, I^2C, and predominantly PCIe. Versal's accesses are mediated by the AMD PCI Express Multi Queue DMA (QDMA) engine, implemented in programmable logic, which translates PCIe transactions (TLPs) into AXI and supports two modes: a lightweight bridge for control/configuration and a high-throughput DMA mode. In DMA mode, multiple queues, descriptor rings, and interrupts enable efficient bulk transfers with minimal CPU overhead. The QDMA

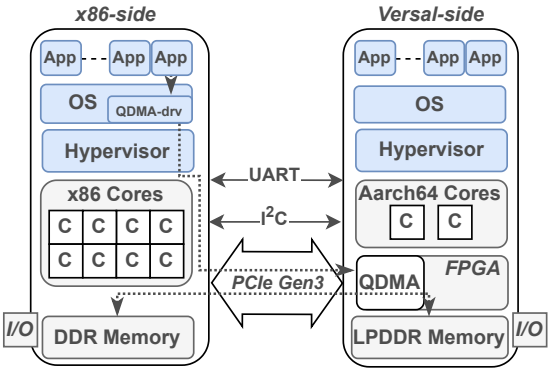


Fig. 1: Architectural view of the Embedded+ platform, showing the x86 *host*, the Versal *device* (AArch64 + FPGA), and PCIe/QDMA as the main communication channel.

is attached to the Versal's programmable Network-on-Chip (NoC), allowing DMA requests to target memory-mapped resources such as LPDDR or AI engines. For data transfers, it fetches descriptors from the host's descriptor rings and issues corresponding read or write requests on the PCIe link. The AMD-provided QDMA-IP driver sets up rings, configures queues, and allocates message-based interrupts (MSI/MSI-X) vectors for completion signaling to the host.

IV. Cross-ISA Real-Time Use-Case Scenarios

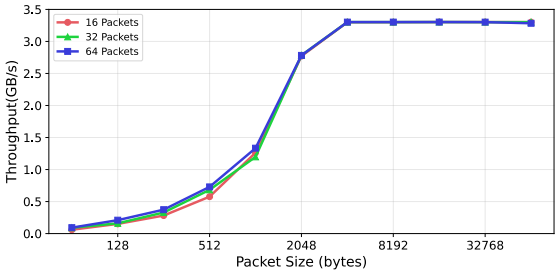
The tight and low-power integration enables new execution models where workloads are dynamically managed across the x86 and Versal-side. We outline two use-cases (UC).

(UC1) energy-efficient task migration: By migrating workloads between high-performance processors and low-power, energy-efficient cores, systems can optimize energy consumption without compromising responsiveness [11]. For example, in UAVs, compute-intensive sensor fusion and control tasks during takeoff and landing may execute on the powerful cores, while during steady-state flight, the workloads can migrate to energy-efficient cores to prolong battery life. Real-time feasibility requires bounded migration latency, predictable interconnect transfers without unbounded blocking or jitter, and schedulability analyses that account for core-dependent WCETs, as previously explored on same-ISA Arm big.LITTLE platforms [11], [12].

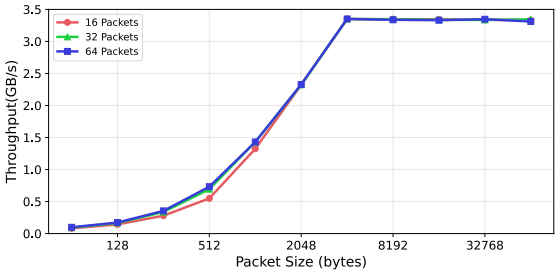
(UC2) accelerator sharing across ISAs: FPGA and hardware accelerators can be dynamically reallocated between x86 and Versal domains. For example, in safety-critical automotive settings, high-priority inference workloads might use accelerators on both domains, with PCIe/DMA arbitration enforcing isolation across mixed-criticality tasks. The bandwidth experienced by the x86 (Versal) domains during data transfers to/from the other domain is a critical factor to characterize the overall real-time guarantees.

V. Cross-ISA SW Architectures for Real-Time

Different system software models can realize these scenarios, each with distinct trade-offs.



(a) x86-to-Versal traffic (write)



(b) Versal-to-x86 traffic (read)

Fig. 2: Throughput measurements on the Edge+. Each curve reports transfer rates for packet sizes between 64 Bi and 64 KiB under different numbers of concurrent DMA transactions, showing the scaling behavior of the PCIe link in both directions.

Partitioning with OpenAMP. The most straightforward approach runs independent OS instances on each side, communicating via open-source standardized frameworks such as OpenAMP (Open Asymmetric Multi-Processing). This setup preserves simplicity and modularity, but task migration (UC1) and accelerator sharing (UC2) must be implemented via ad-hoc middleware in user space, as no OS-level migration is available. This hinders automation and real-time guarantees.

Heterogeneous-ISA Hypervisor with VM Migration. One hypervisor spanning multiple ISAs, as suggested by Omnivisor [13] and recent Xen vision by AMD [6], can enforce isolation while supporting VM migration. This enables dynamic migration of workloads (UC1) and sharing accelerators via *e.g.*, VirtIO [14] and DSM [15] (UC2). However, such architectures are ill-suited for real-time analysis, as bounding the latency of VM migrations and the interference caused by VirtIO-based resource sharing is challenging.

Multikernel OS with Cross-ISA Migration. A multikernel OS (*e.g.*, [1]) supports fine-grained process migration across ISAs, maximizing flexibility and resource utilization. Such systems typically rely on a DSM to maintain a consistent address space across nodes, enabling transparent cross-core memory access. This approach offers transparent accelerator sharing (UC2) and dynamic load balancing (UC1), but increases system complexity complicating real-time evaluations.

VI. Preliminary Experiments

We argue that hypervisor- and multikernel-based designs have greater potential for efficiently supporting real-time workloads on embedded heterogeneous-ISA architectures. In this work, we start exploring their trade-offs by focusing on interconnect performance, which is critical for both VM migration and distributed shared memory (DSM) synchronization (key aspects of hypervisor and multikernel architectures, respectively). Specifically, we measure the achievable PCIe throughput relative to local memory bandwidth on both sides to assess the overhead introduced by a DSM abstraction, and we evaluate transfer latency across varying data sizes, which directly impacts the migration time of a VM of comparable size. To this end, we set up QDMA and NoC such that data traffic targets the local LPDDR on the Versal (FPGA clocked at 250 MHz). On the x86-side, we run Ubuntu 22.04 LTS (Linux 6.8), configured with the `performance` scaling governor and 2 MiB huge pages enabled, and on the Versal-side Petalinux v2024.1 (Linux 6.6.10). PCIe communication relies on AMD's open-source QDMA driver (`v2024.2`). We devise two classes of micro-benchmarks to characterize the Edge+ interconnect: (1) the *maximum channel throughput* of the PCIe Gen3 x4 link and (2) the *fundamental transfer latency* of memory blocks.

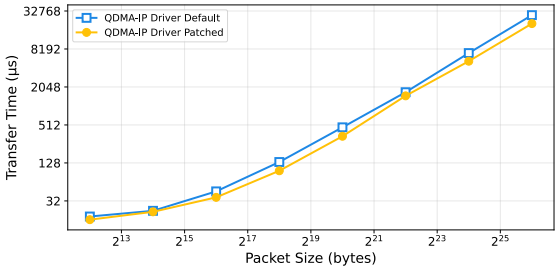
A. Channel Throughput

Using `dma-perf` (an AMD utility to measure DMA performance), we measure data transfer rates between the x86 and Versal subsystems. Fig. 2 shows that throughput scales with packet size and number of outstanding DMA transactions before saturating near 3.3 GB/s in both directions. This corresponds to about 82% of the PCIe Gen3 x4 theoretical peak throughput (4 GB/s), with the remaining gap largely explained by 128b/130b encoding overhead and per-descriptor processing overheads in the QDMA engine. The symmetric results suggest that the bottleneck lies in descriptor management rather than link direction. We plan to further investigate these overhead sources in future work.

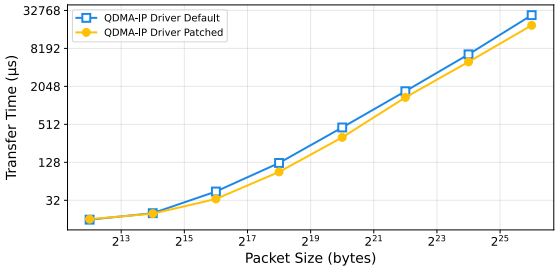
To compare PCIe performance with local memory access, we also measured the bandwidth that x86 and Versal-side can achieve locally. We employed the same memory benchmark used in prior works [16] to analyze DRAM behavior. We set up the benchmark to access a 256 MiB buffer mapped as huge pages. Each iteration performs a *modify* operation that updates part of a cache line, forcing a read-modify-write sequence. Results show that, on average, an x86 core reaches about 10 GB/s on its local DDR memory, while a Cortex-A72 core on the Versal-side attains roughly 3 GB/s on LPDDR.

B. Transfer Latency

To evaluate the cost of migrating memory across ISAs, we measured the transfer time for different buffer sizes, from a 4 KiB page (*i.e.*, fine-grained memory migration) to 64 MiB (*i.e.*, VM state migration). To reduce the overhead of the OS, which is not the main focus of this experiment, we executed the transfers in a hot loop with 1024 iterations. Huge pages (2 MiB) are enabled on the x86-side to allocate physically contiguous and DMA-capable buffers. A closer inspection of the QDMA driver reveals that even when hugepage-backed buffers



(a) x86 DDR to Versal's LPDDR



(b) Versal's LPDDR to x86 DDR

Fig. 3: Blocking transfer time across increasing payload sizes (4 KiB to 64 MiB) for default and patched QDMA drivers.

are physically contiguous, it splits them into 4 KiB chunks, each associated with an individual descriptor. This forces the engine to fetch and process thousands of descriptors for large transfers, introducing substantial overhead. To quantify such overhead, we repeated the experiment with a patched driver forcing one descriptor for each huge page.

Figs. 3a and 3b show the measured blocking transfer times, plotted on a logarithmic scale, for both driver configurations. For payloads smaller than 64 KiB, the system does not reach the peak bandwidth observed in the throughput benchmarks, as fixed costs from driver setup, descriptor posting, and DMA transaction initiation dominate total latency. As the transfer size grows, these fixed overheads are amortized, and the effective bandwidth approaches the one reported earlier. The *"patched"* driver lowers the end-to-end transfer latency, likely thanks to the reduced descriptor count with one descriptor for every 2 MiB huge page instead of one per 4 KiB page, achieving an improvement of around 20% on average.

VII. Discussion and Future Work

The presented early results indicate that the next generation of heterogeneous-ISA platforms, such as the Edge+, narrows the performance gap that previously made cross-ISA migration impractical on embedded systems.

From a migration perspective, prior works on VM migration on embedded platforms relied on Ethernet links and required several seconds to transfer tens of MiBs [2], making the approach useful for energy efficiency but unsuitable for latency-sensitive scenarios. On the Edge+, a 64 MiB transfer completes within tens of milliseconds over PCIe, reducing migration cost by orders of magnitude. This opens the possibility of revisiting cross-ISA migration as a practical mechanism to improve the energy efficiency of real-time systems, provided that residual overheads can be bounded and analyzed. Promisingly, our latency results are comparable to those obtained on same-ISA architectures, which have previously been modeled to exhibit bounded I/O virtualization and inter-VM communication [14].

Furthermore, we observed that the PCIe throughput (3.3 GB/s) is comparable to the average memory bandwidth achieved locally by the Versal-side (3 GB/s), even though it remains below the 10 GB/s measured on the x86 side. These results indicate that a DSM abstraction between x86 and Versal domains could be implemented without excessive performance loss. However, since DSM is not hardware cache-coherent, explicit synchronization and consistency management would incur additional costs that must be modeled for real-time use.

Finally, it is important to consider that the Edge+ represents only the first implementation of the AMD Embedded+ architecture. The current PCIe Gen3 ×4 interconnect is constrained by the Ryzen Embedded processor's limited 16-lane configuration, which is shared with NVMe and other peripherals. Future Embedded+ variants might scale to ×8 or ×16 links, potentially doubling or quadrupling available bandwidth. Such improvements would further strengthen the practicality of cross-ISA migration and DSM-based communication. Future work will focus on implementing and comparing these mechanisms within representative real-time workloads to evaluate their predictability and energy-efficiency benefits.

References

[1] A. Barbalace *et al.*, "Breaking the boundaries in heterogeneous-ISA datacenters," in *ASPLOS*, 2017.
[2] P. Olivier *et al.*, "HEXO: offloading long-running compute- and memory-intensive workloads on low-cost, low-power embedded systems," *IEEE Trans. Cloud Comput.*, vol. 12, no. 4, 2024.
[3] T. Xing *et al.*, "Stramash: A fused-kernel operating system for cache-coherent, heterogeneous-isa platforms," in *ASPLOS*, 2025.
[4] AMD, "AMD Embedded+ Architecture," https://www.amd.com/en/products/embedded/embedded-plus.html.
[5] SAPPHIRE, "EDGE+ VPR-4616-MB," https://www.sapphiretech.com/en/commercial/edge-plus-vpr_4616.
[6] S. Stabellini, "OSS 2025 Talk: The Xen Safety Concept, a Major Milestone Toward Certification," https://ossna2025.sched.com/event/1zfmK.
[7] A. Baumann *et al.*, "The multikernel: a new OS architecture for scalable multicore systems," in *SOSP*, 2009.
[8] A. Bapat *et al.*, "Dapper: A lightweight and extensible framework for live program state rewriting," in *ICDCS*, 2024.
[9] S. Kuenzer *et al.*, "Unikraft: fast, specialized unikernels the easy way," in *EuroSys*, 2021.
[10] T. Xing *et al.*, "H-container: Enabling heterogeneous-ISA container migration in edge computing," *ACM TOCS*, vol. 39, no. 1-4, 2021.
[11] Y. Ma *et al.*, "Online Resource Management for Improving Reliability of Real-Time Systems on "Big-Little" Type MPSoCs," *TCAD*, 2020.
[12] A. Mascitti *et al.*, "Dynamic Partitioned Scheduling of Real-Time DAG Tasks on ARM big.LITTLE Architectures," in *RTNS*, 2021.
[13] D. Ottaviano *et al.*, "The Omnivisor: A real-time static partitioning hypervisor extension for heterogeneous core virtualization over MPSoCs," in *ECRTS*, 2024.
[14] D. Casini *et al.*, "Latency analysis of I/O virtualization techniques in hypervisor-based real-time systems," in *RTAS*, 2021.
[15] H. Chuang *et al.*, "Aggregate VM: why reduce or evict VM's resources when you can borrow them from other nodes?" in *EuroSys*, 2023.
[16] A. Pradhan *et al.*, "Predictable Memory Bandwidth Regulation for DynamIQ Arm Systems," in *RTCSA*, 2025.