Explorative surveying RISC-V open hardware and specifications for mixed-critical systems

Samuel Ardaya-Lieb[1], Holger Blasum[2], Enkhtuvshin Janchivnyambuu[2], Florian Krebs[2], Jan Reinhard[2], Darshak Sheladiya[2], and George Violettas[2]

[1]Consolinno Energy GmbH
[2]SYSGO GmbH

Abstract

Mixed-critical systems are essential in modern computing, where applications of different criticality levels share the same hardware platform(s) and/or resources. This paper explores the robustness and implementation of mechanisms for such mixed-critical systems utilizing RISC-V specifications and available open hardware. In particular, the CVA6 processor and OpenPiton System on Chip (SoC) are used. We also research the mixed-criticality suitability of core-to-core communication, resource management, and specific components like performance counters, memory, and caches. Furthermore, we discuss future developments of mixed-critical systems utilizing open hardware.

1 Mixed-critical systems and hardware

In mixed-critical(ity) systems, applications of different criticality (typically with a focus on safety, but possibly including security requirements [1]) , run on the same hardware platform [2, 3]. Mixed-criticality can also be realized by physically separating hardware (at a slightly larger cost), often during production planning and design, where the exact size of the critical part(s) is not yet known, and a software solution is more flexible. Mixed-critical systems have many use-cases and applications in the modern computerized world [4, 5]. The simplest case for such a system is where one critical and one less critical application rely on the same platform (Fig. 1).

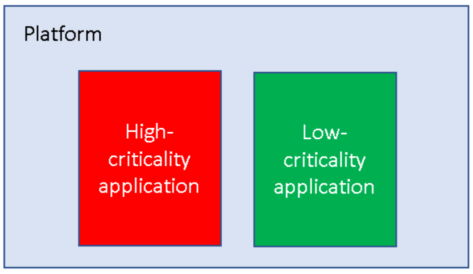

Figure 1: *Mixed-criticality Platform view*

1.1 Multicore considerations

Traditionally, the 60s' time-sharing systems worked out on the notion of individual processes concurrently running on the same processor [6].

However, nowadays, with application processors having multiple cores, a quite common way of separation is to assign one or several cores to each protection domain. So, when mixed critical systems are represented on multicore hardware, a typical setup looks like the one in Fig. 2, where cores are sharing multiple resources, such as L2 caches, DRAM and PCI controller(s). When configuring such shared resources, it needs to be ensured that a high-criticality application cannot be blocked by a lower-criticality one.

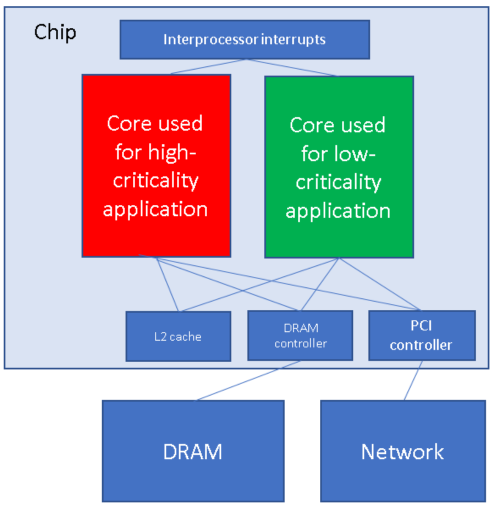
Chip
Interprocessor interrupts
Core used for high-criticality application
Core used for low-criticality application
L2 cache
DRAM controller
PCI controller
DRAM
Network

Figure 2: *Chip view with two cores, including shared resources*

Table 1 is a non-exhaustive list of referencing papers where resources are discussed by authors Fuchsen (F) [7], Agirre et al. (A) [8], or Cerrolaza et al. (C) [9].

An individual core may consist of multiple components, such as a processing pipeline, Translation Look-aside Buffers (TLBs), or L1 caches. It could even have tightly-coupled memory assigned, as depicted in Fig. 3. However, those resources are usually

Table 1: *Discussion of shared resources in the literature*

Description	Author		
Cache coherency/sharing	F	A	
Memory bus	F	A	
PCI bus	F		C
I/O devices	F		
Interrupts	F		
Interconnect		A (CoreNet)	
Memory controller		A	
IO MMU		A (PAMU)	
Scratch memory			C

exclusive to a core and thus cannot be the source of resource-sharing conflicts.

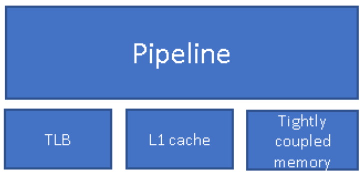

Figure 3: *Core view: exclusively allocated resources*

1.2 Hardware implementations

For mixed-critical systems, viewed as a hardware platform, it is possible to choose systems that are completely non-configurable. For example, the AAMP7 processor was such a design [10]. Also, an early RISC-V multicore processor optimized for determinism was developed in the T-Crest project [11]. As custom designs are expensive to maintain, there seems not to be a wide general adoption of such systems (i.e., AAMP7 and T-Crest). Recently, RISC-V is gaining more momentum, and mechanisms relevant to mixed-criticality are gradually being standardized for it via RISC-V International. The survey at hand looks at current RISC-V specifications and general-purpose open-source RISC-V hardware for what is available for resource management.

With regards to verifying separation claims, an additional feature of RISC-V, it not only provides an openly available instruction set architecture (ISA), but also there exists an ecosystem of open-source hardware implementations [12, 13], whose openly inspectable design allows in principle direct verification of non-interference, which is generally not so easy in typical "closed-source" hardware where register-level design is a trade secret.

2 RISC-V components for mixed-critical systems

Our work is to collect information on resource management or other mechanisms for mixed-critical systems in the RISC-V specifications, and other open hardware resources. So far, we have looked at the CVA6 open core (RV64IMAC) and System on Chip (SoC) demonstration based on OpenPiton [14], which is an open-source, general-purpose, multithreaded, manycore processor and framework, with CVA6. This particular SoC has been built for performance rather than for mixed-criticality, but we are not aware yet of any open hardware SoC explicitly targeting mixed-criticality. We are using this SoC as a running example for mixed-criticality mechanisms discussed below, and for each component, a general state in the RISC-V domain will be followed by the discussion of this SoC.

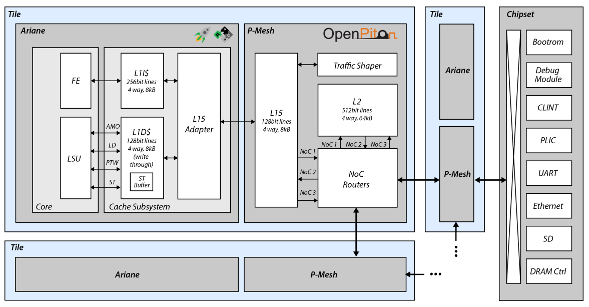

Figure 4: *OpenPiton CVA6, from [14]*

2.1 Performance counters

Quality of Service (QoS) performance counters can be used for partitioning resources, e.g., counting memory accesses [7]. Performance counters are defined in the RISC-V Privileged Specification [15] Sect. 3.1.10 and in [16, 17], using RCIDs (Resource Control IDs) and MCIDs (Monitor Counter IDs). On the OpenPiton SoC, we have CVA6 implementing Control Status Registers, i.e., CSR-based performance counters [18], including the standard 64-bit clock cycle counter `mcycle`, the retired instruction counter `mstret` as well as the six generic 64-bit event counters, corresponding event selectors, which can be enabled/disabled via the `mcount` inhibit CSR. The supervisor and user access of performance counters are allowed through enabling the `mcounteren` and `scounteren` CSRs.

2.2 Memory

Dynamic Random Access Memory (DRAM) can be assigned directly to cores, through asymmetric multiprocessing (AMP), or be shared among cores by symmetric multiprocessing (SMP). In the latter case, sources of contention can be a shared memory controller and a shared memory bus. This kind of contention on the memory bus has been observed in [7, 8]. The usage of the shared memory bus as a side channel has been demonstrated in [19]. SMP also means that we need the protection of synchronization mechanisms (e.g., spinlocks or synchronization instructions such as memory barriers) by access control, e.g., bound to `hart`[1] ID. Whenever memory is shared, regardless of AMP or SMP, `Rowhammer` [22] attacks may be used to inspect adjacent memory cells. The analyzed OpenPiton platform has shared DRAM. However, critical task DRAM access can be shielded from interference by non-critical tasks DRAM access using the MMU (Memory Management Unit).

The DRAM Protection from Input/Output (IO) devices, is described in [8] as the control of interactions coming from IO devices by the IO MMU (Memory Management Unit) or the IO PMP (I/O Physical Memory Protection). RISC-V has IO PMP for this [23]. The analyzed OpenPiton setup does not have this yet though.

A Tightly-Coupled Memory / TCM is a per-core memory and typically a core-exclusive resource, hence there are no side effects. We observe that TCM access is currently not standardized and having standard control and discovery mechanisms for TCM would be desirable towards building more portable system software. Where TCM is indeed shared on a chip between cores, then access control is needed. The analyzed OpenPiton setup does not have TCM.

Caches store data from memory closer to the CPU, removing the need for memory lookups. For example, a cache can be on-chip whereas the memory is DRAM. L2/L3 caches are usually shared between different cores, to make memory coherency simpler. The author in [7] has observed significant slow-downs of performance due to cache sharing of L2 caches as well as due to the coherency protocol. Similarly, authors in [19] have demonstrated covert channels by caching effects. Although caches can be partitioned, such a feature is not yet standardized in RISC-V [24]. If cache partitioning is supported, the redundancy (i.e., number of "ways") in a shared cache is typically a multiple of the number of `harts` sharing that level of cache, so that a way can uniquely be assigned to this `hart`. The translation TLB is usually not affected, as it is usually CPU-local. In the analyzed OpenPiton setup, the SoC cache hierarchy consists of three different cache levels: L1 and L1.5 are part of the CVA6 tile. The L2 cache is not partitioned, which would provide an interference channel. Communication between L1.5 and L2 is done via SoC using a distributed, directory-based cache coherency protocol. For mixed critical systems, such a system demands higher levels of verification such that protocol should handle memory errors (e.g., single-bit flips) and for system reliability, it is important to have fault-tolerance mechanisms. Additionally, it is important to look at side-channel attacks (e.g., cache-based timing attacks) as it is important to ensure cache coherence latency should not affect the real-time behavior of the system.

2.3 Synchronization domains

For controlling TLB flushes, the RISC-V provides the privileged instructions HFENCE.VVMA for use in hypervisor mode and SFENCE.VMA for use in supervisor mode [15], which guarantee that any previous stores already visible to the current RISC-V `hart` are ordered before subsequent instructions in that `hart` (also implicitly) reference the memory management data structures. However, if multiple domains are to be kept apart (e.g., several operating system instances running on a hypervisor), then it is more useful to be able to ensure that memory synchronization (e.g., by `fence` instruction) is restricted to selected actors (such as operating systems, hypervisors), called "shareability domains" on ARM [25], and this seems not have yet been specified on RISC-V. Synchronization domains are also not implemented on the CVA6 yet.

2.4 Buses

Interconnects are connecting components and many architectures are shared among cores. In principle, of course, it is possible to have dedicated bus lines per core, but the more common usage principle is to have a hardware bus arbiter, and allow the OS control of that arbiter, permitting the assignment of bandwidth quota.

Regarding Inter-Core buses, state-of-the-art ranges from undocumented behavior [8] to hardware-configurable exclusive bus access (at the cost of redundant hardware structures) [26]. In the OpenPiton demonstration SoC the building blocks are tiles, where each tile consists of a core, caches (L1.5/L2), a floating-point unit (FPU), a CPU-Cache Crossbar (CCX) arbiter, a Memory Inter-arrival Time Traffic Shaper (MITTS), and networks-on-a chip routers (NoCs). These three NoCs are physical networks and packets are routed using dimension-ordered wormhole routing. To ensure deadlock-free operation, the L1.5 cache, L2 cache, and memory controller give different priorities to different NoC channels. However, there is no evidence that this SoC provides deterministic com-

[1] A hart is defined as a hardware thread, either one per core without hardware multithreading or multiple harts per core with hardware multithreading. The concept of a hart is a hardware resource abstraction representing an independently advancing RISC-V execution context with its own instruction fetching [20, 21].

munication where the timing of messages would be predetermined and guaranteed, which is an important safety-critical application.

2.5 Microarchitecture

Side-effects of speculative execution have been used for the Meltdown [27] and Spectre [28] attacks. A mixed-criticality-friendly microarchitecture could allow to `fence off` state of speculative execution, i.e., a reset of the speculative execution state has been proposed as a `fence.t` instruction [29]. In the analyzed OpenPiton setup, the CVA6 core is a 6-stage RISC-V compatible processor core that supports an efficient out-of-order execution, hence, `fence.t` support in CVA6 is benefical if cores are scheduled to run multiple security domains, and/or resource allocations would change dynamically.

2.6 Positioning of the analyzed SoC

In the previous sections, different approaches to stronger isolation on RISC-V platforms and their state of specification have been demonstrated. In this section, we illustrate how to build such systems in general. Those approaches are not specific to RISC-V, but they illustrate the usefulness of the aforementioned components.

- A small system with hardware-dedicated resources, such as a small monitor or safety application, can use dedicated cores with tightly-coupled memory.
- If tightly coupled memory is not available, e.g., another common technique for building mixed-critical systems is to ensure that the entire memory fits into the core-local L1 cache.
- Larger systems using HW/SW performance monitoring are used in cases where the usage of shared resources is unavoidable. The most common techniques comprise of partitioning some shared resources by hardware-enforced access control (e.g., cache partitioning) and other resources by OS-controlled accesses, typically relying on performance counters [7].

The OpenPiton demonstration SoC discussed above as an example can be used for the second and third approaches.

3 Results and further steps

We have outlined some ingredients that RISC-V cores need to provide for mixed-critical systems and concretely looked at the preconfigured OpenPiton setup. Our looking at concrete RISC-V implementations for support for mixed-critical systems is similar to [3], looking at interrupts and also doing benchmarks on Noel-V, whereas we are looking at a OpenPiton demonstration and survey for building blocks such as performance counters, and bus partitioning both on this platform as specification-level developments. We aspire that our survey will support future designers towards building future-generic, yet mixed-criticality-friendly open hardware platforms. A further (obvious) step could be to specify a more mixed-criticality-oriented demonstration SoC. Moreover, for a reliability assessment, in the ISOLDE project [30], we are currently polling partners for certification plans of their components ("precertification checklist"). In terms of formal analysis, on open-source components, in principle, a rigorous analysis of non-interference by hardware information flow tracking could be performed, e.g., proof-carrying hardware [31].

Acknowledgment

This work was supported by the European Union projects TRISTAN (RIA Chips-JU, GA 101095947), and ISOLDE (RIA Chips-JU, GA 101112274). Insights were drawn from Jérôme Quévremont and Rudolf Fuchsen, and from discussions at the RISC-V functional safety special interest group [32].

References

[1] Stéphane Paul. "On the meaning of security for safety (S4S)". Opatija, May 2015, pp. 379–389.

[2] Alan Burns and Robert Ian Davis. "Mixed Criticality Systems - A Review : (13th Edition)". *White Rose - Research Online* (Feb. 2022).

[3] Ralf Ramsauer et al. "Static hardware partitioning on RISC-V: Shortcomings, limitations, and prospects". *8th World Forum on Internet of Things (WF-IoT)*. IEEE. 2022, pp. 1–6.

[4] Simon Barner et al. "DREAMS Toolchain: Model-Driven Engineering of Mixed-Criticality Systems". *ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Austin, TX: IEEE, Sept. 2017, pp. 259–269.

[5] Junghwan Lee and Myungjun Kim. "Real-Time Scheduling for Mixed-Criticality Systems in the Automotive Industry". *Journal of Computing Science and Engineering* 14.1 (Mar. 2020), pp. 9–18.

[6] Jerome H. Saltzer and Michael D. Schroeder. "The protection of information in computer systems". *Proceedings of the IEEE,* 63.9 (1975), pp. 1278–1308.

[7] Rudolf Fuchsen. "How to address certification for multi-core based IMA platforms: Current status and potential solutions". *Digital avionics systems conference (DASC), 2010 IEEE/AIAA 29th.* Oct. 2010, 5.E.3–1 – 5.E.3–11.

[8] Irune Agirre et al. "On the tailoring of CAST-32A certification guidance to real COTS multicore architectures". *12th IEEE International Symposium on Industrial Embedded Systems (SIES).* Toulouse, June 2017, pp. 1–8.

[9] Jon Perez Cerrolaza et al. "Multi-core Devices for Safety-critical Systems: A Survey". *ACM Computing Surveys* 53.4 (July 2021), pp. 1–38.

[10] David Greve, Raymond Richards, and Matthew Wilding. "A Summary of Intrinsic Partitioning Verification". *5th International Workshop on the ACL2 Theorem Prover and Its Applications (ACL2 2004), Austin, TX.* 2004.

[11] Martin Schoeberl et al. "T-CREST: Time-predictable multi-core architecture for embedded systems". *Journal of Systems Architecture* 61.9 (Oct. 2015), pp. 449–471.

[12] RISC-V foundation. *RISC-V Cores and SoC Overview.* Tech. rep. `https://github.com/riscvarchive/riscv-cores-list`. Dec. 2023.

[13] OpenHW Group. *CORE-V Family of Open-Source RISC-V Cores.* Tech. rep. `https://github.com/openhwgroup/core-v-cores`. Dec. 2023.

[14] Jonathan Balkind et al. "OpenPiton: An open source manycore research framework". *ACM SIGPLAN Notices* 51.4 (2016), pp. 217–232.

[15] Andrew Waterman, Krste Asanovic, and John Hauser. "The RISC-V instruction set manual, volume II: Privileged architecture, document version 20211203". *RISC-V Int., Dec* (2021).

[16] Ved Shanbhogue. "Quality of Service in a RISC-V SoC". *RISC-V Summit.* Santa Clara, CA., Nov. 2023.

[17] Kersten Richter. *RISC-V Capacity and Bandwidth Controller QoS Register Interface.* Tech. rep. `https://github.com/riscv-non-isa/riscv-cbqri`. July 2023.

[18] CVA6 documentation. *CSR performance counters control.* Tech. rep. `https://docs.openhwgroup.org/projects/cva6-user-manual/01_cva6_user/CSR_Performance_Counters.html`. 2024.

[19] Don Kuzhiyelil and Sergey Tverdyshev. "Timing covert channel analysis on partitioned systems". *Proceedings of ESCAR Europe, Hamburg 18-19 Nov 2014.*

[20] Heidi Pan, Benjamin Hindman, and Krste Asanovic. "Lithe: Enabling efficient composition of parallel libraries". *Proc. of HotPar* 9 (2009).

[21] Kersten Richter. *RISC-V glossary.* Tech. rep. `https://github.com/riscv/riscv-glossary`. RISC-V, May 2024.

[22] Yoongu Kim et al. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors". *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA).* ISSN: 1063-6897. June 2014, pp. 361–372.

[23] RISC-V International. *RISC-V IOPMP Specification.* Tech. rep. `https://github.com/riscv-non-isa/iopmp-spec`. Oct. 2023.

[24] Allen Baum et al. "Cache Management Operations for RISC-V" (2022). `https://github.com/riscv/riscv-CMOs/blob/master/specifications/cmobase-v1.0.1.pdf`.

[25] ARM. *Architecture Reference Manual for A-profile architecture.* Tech. rep. `https://developer.arm.com/documentation/ddi0487/latest/`. 2023.

[26] Helmut Kurth. "Security Evaluation of a RISC-V-based SoC". *RISC-V Summit Europe* (2023).

[27] Moritz Lipp et al. "Meltdown: Reading kernel memory from user space". *Communications of the ACM* 63.6 (2020), pp. 46–56.

[28] Paul Kocher et al. "Spectre Attacks: Exploiting Speculative Execution". *Communications of the ACM* 63.7 (2020), pp. 93–101.

[29] Ronan Lashermes. "Fence.t, a RISC-V extension proposal". *Microarchitecture Side Channels Special Interest Group (uSC SIG)* (2024). `https://github.com/riscv-admin/uarch-side-channels/blob/main/fence_t/fence_t.adoc`.

[30] ISOLDE project. *RISC-V processing systems and platforms.* funded by the EU's HE KDT programme, Gr. Agr.: 101112274. `https://www.isolde-project.eu/`. 2023.

[31] Wei Hu, Armaiti Ardeshiricham, and Ryan Kastner. "Hardware Information Flow Tracking". *ACM Computing Surveys* 54.4 (May 2022), pp. 1–39.

[32] Jerome Quévremont. "Introduction to RISC-V Functional Safety special interest group". *Spring 2022 RISC-V Week, Paris* (2022).