Providing QoS policies for mixed-criticality applications on RISC-V based MPSoCs

Raúl de la Cruz[1]*, Gonzalo Salinas[1], Alejandro Garcia[1]

[1]Connected & Real-Time Systems Group, Collins Aerospace Applied Research & Technology, Ireland

Abstract

More integrated systems come at the price of harder predictability and determinism. This is specially true on MPSoCs with myriad shared resources that can behave as interference channels thus creating contention and non-deterministic behaviour. Variability makes MPSoCs very hard and expensive to certify for safety-critical systems. In order to ease the certification process and guarantee tasks' timeliness, drastic measures are traditionally enforced in the system configuration. Some of these include the deactivation of resources to avoid contention scenarios, synchronization of core's partitions using ARINC-653 schedulers and long provisioning of budget times increasing deadlines. These mitigation strategies impact drastically on the performance of the system, either underutilizing most of the advanced MPSoCs capabilities or reducing the integration level of mixed-criticality applications. This work proposes novel QoS strategies to enforce determinism of mixed-criticality applications while performance is preserved. The infrastructure has been successfully evaluated on a customized quad-core RISC-V RocketChip architecture using high assurance level applications with stringent deadlines.

Introduction

Aerospace industry is evolving towards better SWaP-C factor computing systems due to the trend of more electric-powered autonomous and intelligent systems. These systems will require additional computing power to run simultaneously multiple, complex and AI-based applications on the same integration system [1]. Semiconductor manufacturers are facing performance limitations through parallelization strategies with the integration of heterogeneous architectures in complex Multiprocessors Systems on Chip (MPSoCs). In addition, manufacturers are also ceasing the single-core production and hence there is a current deprecation and discontinuation of these type of systems. This trend will necessarily require the adaptation of the current safety-critical systems (SCS) to MPSoCs to ensure long-term operation and maintenance.

Multicore architectures are well-known to leverage the computing power for general purpose applications and high performance computing. However, when it comes to critical systems their predictability is at risk due to the appearance of interference channels and contention in the form of execution delays. Interference channels appear when shared resources do not provide enough resources to handle concurrent requests from different masters initiators (e.g. cores, DMAs or PCIe devices). Contention occurs when requests have to be serialized, queued or stored in order to be served. In this context, masters may need to wait contenders' requests to be completed before their own get served.

Objectives

This work aims at proposing QoS strategies on MPSoCs hosting mixed-criticality applications to mitigate its non-deterministic behavior and guarding the system performance. This objective is achieved by two cornerstone efforts. Firstly, the integration using Chipyard of a custom HW infrastructure on a RISC-V RocketChip architecture that measures accurately the contention produced in the TileLink interconnect [2]. And secondly, deploying QoS policies at SW level orchestrating the custom HW to enable a flexible scheduling scheme and avoiding the traditional and rigid ARINC-653 scheme. Our full QoS stack accommodates mixed-criticality applications with different periodicities and assurance levels on the same SoC. Although the QoS mechanism has been tested on a 4-core RISC-V bare-metal implementation, the approach is fully agnostic and could be deployed on top of any HyperVisor/RTOS.

QoS stack to mitigate contention

Contention is detrimental to the determinism of the system leading to very pessimistic WCETs for each executed task. In a mixed-criticality environment, an unanticipated contention from a low assurance level application can produce a high critical application to overrun its WCET and miss its deadline. Hence, it is imperative to create methodologies to measure and and limit the tasks' contention at runtime whenever safe bounds are exceeded. The methodology applied

*Corresponding author: `raul.delacruz@collins.com` to derive our QoS infrastructure is based on:

- Identify the interference channels. (HW)
- Deploy contention assessment modules. (HW)
- Assign contention quotas based on tasks' profiles and priorities. (SW)
- Devise QoS policies for exhausted quotas according to the applications' criticality level. (SW)

Identifying interference channels. First, the interference channels that can create contention need to be assessed. Full visibility and understanding of the microarchitecture's behavior is a must. This is usually challenging when COTS are used as many IPs are kept undocumented by manufacturers and NDAs need to be signed to overcome this problem. In this regard, the use of RocketChip architecture and open standards such as RISC-V grant us full visibility and control of the whole hardware architecture.

The assessment of our custom microarchitecture drove our attention on the contention produced accessing the main memory, composed of a scratchpad. This scratchpad is accessed by cores using a TileLink interconnect that act as a crossbar and therefore as a potential bottleneck to concurrent requests. After obtaining all the microarchitectural information and how requests are managed through the TileLink standard, the next step was to design a HW module able to monitor and signal when contention happens, which master is producing it (contender) and who needs to wait to complete the transaction (contended).

Contention assessment module. A specific contention assessment module (CSS) was designed in Verilog and integrated in Chipyard to sniff the transfers going through the interconnect and derive the contention cycles according to the information provided by TileLink's signals.

The module is connected to each core tile acting as master device, sniffing core requests and creating N^2 contention signals (Cx_y) being N the number of masters. Our microarchitecture includes 4 cores, so a total of 16 signals named according to the cores producing contention exist. Hence, $C1\_2$ represents contention produced over core 1 due to core 2, whereas $C0\_0$ is the contention of the core zero over itself due to a sustained burst of transactions.

Once that the CCS module was integrated, a specialized IP to count the contention events and provide quota reservations was required. To reduce the implementation effort, an open source module called SafeSU [3, 4] and developed in the SELENE project [5] was used for this purpose. This is a generic statistical module designed for measuring different types of events, supporting both contention assessment signals and PMU events. The SafeSU provides the following HW capabilities:

- A programmable crossbar that routes desired contention signals to event counters.
- Set of weighted counters that track the occurrences of user-defined events on every cycle.
- MCCUs that allow to assign group quotas and trigger interrupts when expired to the PLIC.

QoS policies. Finally, the whole system is orchestrated by the SW layer that configures and manages the QoS policies for our mechanism. This task is performed by a dedicated manager core that arranges several quotas in the SafeSU module based on offline timing analysis of the applications being executed.

Our SW layer is in charge of several tasks. a) Configure the SafeSU crossbar and MCCUs to account for the cycles that each Core has been contended (provided by CSS) and raise an interrupt given a combined set of quotas. b) Reroute external MCCUs' interrupts to specific cores to take countermeasures based on mitigation policies. c) Issue IPIs to resume cores' execution at each quantum slice. d) And finally, implement strategies for the renewal of quotas reconfiguring SafeSU module periodically.

Evaluation

Our QoS infrastructure has been successfully evaluated on three different use cases, demonstrating that the combination of the CCS and SafeSU modules and our QoS policies are a feasible and resilient approach to provide deterministic behaviour in a mixed-criticality system with very stringent deadlines. This approach certainly proved not only that mitigates the contention but also that can provide safety measures for complex scheduling configurations with tasks running at different periods, thus avoiding the strict ARINC-653 scheme.

References

[1] Woodrow Bellamy. "Avionics Industry Advances Toward DAL A Multicore Adoption". In: *aviationtoday.com* (2020).

[2] SiFive. *TileLink specification.* `https://starfivetech.com/uploads/tilelink_spec_1.8.1.pdf`.

[3] Guillem Cabo et al. "SafeSU: an Extended Statistics Unit for Multicore Timing Interference". In: *2021 IEEE European Test Symposium (ETS).* 2021, pp. 1–4. DOI: `10.1109/ETS50041.2021.9465444`.

[4] Pablo Andreu et al. *End-to-End QoS for the Open Source Safety-Relevant RISC-V SELENE Platform.* 2022. DOI: `10.48550/ARXIV.2210.04683`.

[5] H2020 SELENE consortium. *SELENE RISC-V open source hardware platform.* `https://gitlab.com/selene-riscv-platform`. 2021.