AIA User Priority Mask Extension
Minimizing Critical Sections Side-Effects on Real-Time Automotive Systems

Sandro Pinto, José Martins, Manuel Rodriguez, Kajetan Nuernberger*, and Thomas Roecker*

Centro ALGORITMI - UMinho, OSYX Technologies, *Infineon AG

Abstract

Critical sections are widely used in many real-time automotive scenarios; however, if not adequately supported at the ISA level, it can lead to unintended performance impact. Studies within the AUTOSAR community point to nearly 30% CPU-load. In this work, we advocate that the RISC-V ISA (and related specifications) cannot efficiently support critical sections for real-time automotive systems. To address that, we propose a novel AIA extension. We implemented the proposed extension on a CVA6-based SoC endowed with an Advanced Interrupt Architecture (AIA) IP and functionally validated the intended behavior. We are now deriving quantitative evidence/statistics to support the discussion of the current proposal at RISC-V International. We are going to open-source all artifacts to promote collaboration within the RISC-V community.

Introduction

RISC-V is gaining incredible momentum in the automotive industry. Quintauris was jointly funded by six leading automotive-related companies with the goal of advancing the adoption of RISC-V in the automotive market, and at the same time, the European Union is financing millions through multiple R&D projects targeting RISC-V automotive (e.g. RIGOLETTO). Furthermore, RISC-V Automotive SIG was also restarted and increasing synergies are being developed in related working groups (e.g., Safety SIG).

In this work, we highlight the existing gap in the RISC-V ISA (and related specifications) to efficiently support critical sections. To address that, we propose a novel AIA extension - `User Priority Mask` - to minimize the side effects of critical sections in the context of real-time automotive systems.

RISC-V AIA in a nutshell

The AIA specification [1] encompasses (1) an extension to the privileged ISA, the (2) Advanced Platform-Level Interrupt Controller (APLIC), and the (3) Incoming Message-Signaled Interrupt Controller (IMSIC). The newly introduced CSRs, via **Smaia**/**Ssaia**, implement a simplified interface for seamless interaction with the IMSIC. The **IMSIC** consists of a set of interrupt files to support MSIs. For each privilege level and virtual hardware hart capable of receiving MSIs, the IMSIC hart contains an interrupt file. The interrupt file comprises two arrays: one to track pending interrupts and another to enable the interrupt.

Motivation

Critical Sections: Automotive Use Cases. Critical sections are required in various real-time automotive scenarios, such as (i) reconfiguration of memory protection systems, (ii) management of local datasets, and (iii) replacement of key information, to name a few. Although these operations are short (a few nano to milliseconds), they are frequently invoked. Every time a critical section is enforced, a pair of software routines for enabling/disabling interrupts is invoked. In the context of OSEK, the AUTOSAR Classic architecture refers to these routines as *Suspend (OS/All) Interrupts* and *Resume (OS/All) Interrupts*; hereinafter, we refer to these routines as *SR-pair*. While duration of critical sections are limited, i.e., minimizing *off-time*, they are invoked relatively often, e.g., during OS/application rescheduling. Thus, the SR-pair should have the shortest possible runtime to minimize overhead. In some of today's automotive systems, the SR-pair is referenced hundreds of thousands to millions of times, scaling with application complexity/size. Even assuming that only a tiny fraction of these calls (∼1-10%) is actually executed, the SR-pair execution time can notably contribute to the harts workload, as shown in [2] (nearly 30% CPU-load).

Critical Sections: RISC-V ISA Gaps. Considering a privileged architecture like RISC-V, where the Machine-mode software is typically kept very minimal, the majority of IRQs are vectored to Supervisor/Hypervisor privilege for further servicing. Traditionally, if a User-mode task would need to establish a critical section, it would come at the cost of at least two

system calls for switching to Supervisor/Hypervisor-mode, including the corresponding context saving and restoring. Global IRQ disabling is given by the *sstatus* and *sie* CSRs, while the effective interface for source-selective disabling is provided by the interrupt architecture. The Advanced Interrupt Architecture (AIA) is the state-of-the-art reference architecture for interrupt handling functionality in RISC-V. Source-selective disabling is supported via interrupt files implemented in the (core-local) IMSIC, providing a minor identity per IRQ-source (mapped to the respective hart), ultimately determining the preemption scheme's priority. The AIA endows each interrupt file within the IMSIC with a priority-based masking mechanism via a configurable threshold for the IRQ presentation. When the threshold is a nonzero value P, interrupt identities P and higher (i.e., interrupts with lower priority) do not contribute to signaling interrupts to the target hart. However, a key limitation of the AIA, particularly in the real-time domain, is that User-mode lacks a direct interface to the IMSIC for disabling/suppressing interrupts when establishing a critical section. Instead, the software must trap into Supervisor/Hypervisor mode to access the AIA CSRs, introducing non-negligible overhead due to context switching.

AIA User Priority Mask

The AIA User Priority Mask Extension is aimed at giving restrained control to user mode over interrupt priority to allow user mode applications to disable interrupts up to a certain priority level in, for example, time-sensitive critical sections. We introduce new CSRs as follows:

User Threshold (ueithreshold). The *ueithreshold* CSR is a WLRL register that determines the minimum interrupt priority (i.e., maximum interrupt identity) allowing an interrupt to be signaled from the attached S-mode interrupt file, so that it appears as a pending external interrupt in bits *mip.SEIP* and *sip.SEIP*. *ueithreshold* must be capable of holding values between 0 and N, inclusive.

Supervisor-defined Threshold (sminueithreshold). The *sminueithreshold* CSR is a WLRL register that determines the maximum interrupt priority (i.e., minimum interrupt identity) that can be written into *ueithreshold*. Independently of *sminueithreshold*, the value 0 can always be written to *ueithreshold*. *sminueithreshold* must be capable of holding values between 0 and N, inclusive. N is reset-value –> Original behavior.

Threshold Computation (ufueithreshold). The *ufueithreshold* CSR is a RO register with the final threshold value used to determine if a pending-and-enabled S-mode interrupt can be signaled to the attached hart, according to the register values presented in the table 1.

Table 1: *Final threshold computation based on ueithreshold, min_ eithreshold and eithreshold values.*

ueithreshold	sminueithreshold	ufueithreshold
> 0	> 0	min(ueithreshold, eithreshold)
> 0	0	ueithreshold
'dont care'	N	eithreshold

Use-model With the AIA User Priority Mask Extension, user applications can mask interrupts up to a certain level defined by the OS. In edge cases, the OS can set *sminueithreshold* to N, which means that the user cannot mask any interrupts (corresponding to the original behavior), or set *sminueithreshold* to 0, enabling the user code to mask all interrupts. When *final threshold* is a nonzero value P, interrupt identities P and higher are not signaled to the hart (as though those identities were not enabled) regardless of the settings of their corresponding interrupt-enable bits in the *eie* array.

Status, Roadmap, and Conclusion

To date, we have implemented the proposed extension in a CVA6-based SoC endowed with an Advanced Interrupt Architecture (AIA) IP [3]. We have also functionally validated the intended behavior by running a set of low-level benchmarks and micro-operations. We are now proceeding towards the quantitative evaluation, which is two-fold. Firstly, we want to quantify PPA-impact on a reference implementation of the AIA. From initial estimates, this is considered negligible. Secondly, we are going to quantify the benefit for real-time performance. For that, we are preparing a software stack based on the Bao hypervisor [4] and application benchmarks. Once we complete the empirical evaluation, we will engage in discussions among the related RISC-V working groups. We intend to open-source all artifacts to promote collaboration within the RISC-V community.

References

[1] *RISC-V Advanced Interrupt Architecture (AIA)*. RISC-V. Jan. 2025. URL: https://github.com/riscv/riscv-aia.

[2] *Why AUTOSAR fails so often*. 2025. URL: https://www.gliwa.com/media/download/15th_aoc_gliwa_why_autosar_fails_opt.pdf.

[3] Francisco Costa et al. " Open Source RISC-V Advanced Interrupt Architecture (AIA) IP ". In: *RISC-V Summit EU*. 2023.

[4] José Martins et al. "Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems". In: *NG-RES Workshop*. 2020.