Open Source RISC-V Advanced Interrupt Architecture (AIA) IP

Francisco Costa*, Manuel Rodríguez, Bruno Sá, and Sandro Pinto

Centro ALGORITMI/LASI, Universidade do Minho, Portugal

Abstract

This work describes the design and implementation of an open-source Advanced Interrupt Architecture (AIA) IP compliant with the RISC-V AIA specification (v1.0-RC2). We have designed and implemented the core extensions, the Advanced Platform Level Interrupt Controller (APLIC), and the Incoming Message-Signalled Interrupt Controller (IMSIC) IPs. These IPs being integrated into a RISC-V CVA6-based (64-bit) SoC. We conduct a preliminary evaluation of the system and present a hardware report. Our work showcases the feasibility of implementing RISC-V AIA and establishes a base for future research and development. We will open-source our IP to foster collaboration among the RISC-V community.

Introduction

The Platform Level Interrupt Controller (PLIC) is the current interrupt controller in RISC-V systems, but it has limitations in scalability and feature-richness. These limitations are: (i) large physical address space usage; (ii) sharing of M-mode and S-mode global registers; (iii) no support for Message Signal Interrupts (MSI); (iv) no support for interrupt line sensing configuration; and (v) no support for virtualization, leading to increased interrupt latency for VMs [1, 2].

The RISC-V Advanced Interrupt Architecture (AIA) [3] is the novel reference specification for interrupt-handling functionality. In this work, we report the ongoing design, implementation, and validation of an AIA IP compliant with the ratified specification. The AIA IP will be open-sourced, contributing this way to the RISC-V community.

RISC-V AIA in a nutshell

The AIA specification, currently in version 1.0-RC2, is composed of three distinct components: (i) extended local interrupts (AIA CSRs); (ii) Incoming Message-Signalled Interrupt Controller (IMSIC); and (iii) Advanced Platform Level Interrupt Controller (APLIC).

The APLIC consists of a set of interrupt domains. Each domain has its memory-mapped control region. APLICs convert wired interrupts into MSIs when harts implement IMSICs, serving as a replacement for the PLIC in their absence.

The IMSIC supports MSIs with a set of interrupt files, that are composed of arrays to track and enable interrupts. It also provides for virtualization support, enabling direct injection of interrupts into VMs.

*Corresponding author: pg47200@alunos.uminho.pt

Design and Implementation

In this section, we briefly discuss the design and implementation of the AIA in general, and the multiple AIA components (i.e., APLIC, IMSIC) in particular. The target design goals are scalability and modularity.

Design

APLIC. An APLIC domain comprises three modules: (i) gateway, (ii) notifier, and (iii) register control. The gateway module receives the interrupt source and evaluates if it can become pending. The notifier implementation depends on the delivery mode supported by the APLIC. For direct mode, the notifier finds the highest pending and enabled interrupt and notifies the hart. For MSI mode, the notifier sends newly pending and enabled interrupts to a hart IMSIC. The register controller manages the APLIC registers.

IMSIC. Access to the IMSIC occurs through the CSRs. Creating a channel for communication between the CSR module and the IMSIC results in explicit isolation between them. This allows for easy integration of the IMSIC module into other projects.

Implementation

The implementation of the AIA IP was carried out in SystemVerilog HDL. Regarding the APLIC implementation, were created six modules, culminating into 2124 source lines of code (SLoC). For the IMSIC implementation, were developed two modules, encompassing a total of 332 SLoC. Using multidimensional arrays, it is simpler to extend the number of interrupt files that a given IMSIC implement by simply modifying the NR_INTP_FILES parameter of the module.

These IPs were then integrated in a CVA6-based

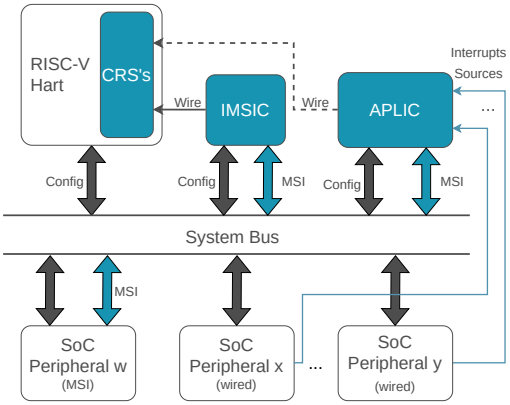

Figure 1: *Simplified view of a generic SoC architecture with AIA related components (highlighted in blue).*

SoC [4]. Originally, the SoC had a RISC-V PLIC, which we replaced with the newly designed APLIC. Then, we implemented the AIA core extensions and finally the IMSIC. Figure 1 shows, highlighted in blue, the AIA components in a generic SoC. At this point, the CVA6 system bus is used to send the MSIs.

Preliminary Evaluation

Functional Validation

The functional validation process involved the use of openSBI, the Linux operating system, and the Bao hypervisor [5]. We successfully run Linux atop of Bao in the CVA6-based SoC with the explored AIA IP.

Hardware Results

Table 1 summarizes the FPGA resources used for various SoC configurations on the Genesys2 board. The first row represents the original SoC with PLIC, while the second row is an SoC configuration where PLIC is replaced by APLIC. It can be observed that the new functionalities of APLIC have increased the LUTs utilization by 3.40 % and the FF by 0.71 %. The third row shows the hardware utilizations for the same SoC, but now with a full AIA, including APLIC, IMSIC with 3 interrupt files (M, S and VS), and the core extensions Smaia, and Ssaia. Compared to the SoC configuration with APLIC, there is a slight increase of 1.54 % in LUTs and 0.34 % in FF, indicating that the core extensions and IMSIC IP do not use much hardware compared to APLIC.

Roadmap

Next, we plan to functionally validate the AIA IP with other system virtualization-based software stacks, such as KVM and XVisor, to ensure compatibility with other software. We also plan to conduct optimizations to the APLIC design, such as using a single APLIC domain, which can reduce hardware cost. Additionally, we plan to explore different design approaches, such as creating a dedicated bus to send MSIs, and implementing the APLIC and IMSIC as a single module placed near the hart.

Table 1: *Hardware resources used in a Genesys2 by different SoCs configurations. (1) Original SoC with PLIC; (2) APLIC replaces the PLIC; (3) Complete AIA IP implementation w/ 1VS-File.*

SoC Configuration	Resource	Utilization	Utilization (%)
PLIC (1)	LUT	74541/203800	36.58
	FF	51446/407600	12.62
APLIC (2)	LUT	81480/203800	39.98
	FF	54334/407600	13.33
AIA (3)	LUT	84610/203800	41.52
	FF	55735/407600	13.67

Conclusion

This work presents the design and implementation of an open-source AIA IP. It will be made public so that it can be used by the RISC-V community. Future activities will mainly focus on implementation, aiming at reducing hardware costs.

Acknowledgments

This work has been supported by Technology Innovation Institute (TII) and FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope UIDB/00319/2020 and Scholarships Project Scope SFRH/BD/07707/2021.

References

[1] Bruno Sá, Jose Martins, and Sandro Emanuel Salgado Pinto. "A First Look at RISC-V Virtualization from an Embedded Systems Perspective". In: *IEEE Transactions on Computers* (2021).

[2] Bruno Sá et al. *CVA6 RISC-V Virtualization: Architecture, Microarchitecture, and Design Space Exploration*. 2023.

[3] *RISC-V Advanced Interrupt Architecture (AIA)*. RISC-V. Mar. 2022. URL: `https://github.com/riscv/riscv-aia`.

[4] Florian Zaruba and Luca Benini. "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2019).

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