



Project: ISOLDE: customizable Instruction Sets and Open Leveraged Designs of Embedded riscv processors

Reference number: 101112274

Project duration: 01.05.2023 - 30.04.2026

Work Package: WP2: Open-Source Foundation Cores

Deliverable **D2.1**

Title **Initial report on foundational IP cores**

Type of deliverable: Report

Deadline: 31.10.2023

Creation date: 31.10.2023

Authors: Nils Wessman, GSL
Rafael Tornero, UPV
Carles Hernández, UPV
André Sintzoff, TDIS
Holger Blasum, SYSGO
Esther Soriano, FEN
Mihai Munteanu, Honorius Galmeanu (XPERI)

Involved grant recipients: Frontgrade Gaisler AB (GSL)
Universitat Politècnica de València (UPV)
Thales DIS France SAS (TDIS)
SYSGO GmbH (SYSGO)
Fent Innovative Software Solutions S.L. (FEN)
FotoNation SRL by XPERI (XPERI)

Contacts: Nils Wessman, GSL, nisse@gaisler.com

# Table of Contents

# 1 Executive Summary

This document describes the work performed within ISOLDE WP2 Open-Source Foundation Cores. WP2 will develop IPs that will be delivered through internal repositories during the work and finally through the ISOLDE virtual repository. WP2 progress will be reported through updates of this report that will be issued as document deliverables D2.2 (M18) and D2.3 (M33).

The requirements and specifications for the work to be performed is established in WP1. At the time of issuing this report, much of the work to be carried out in WP2 is still in the specification or planning stage.

# 2 Introduction

## 2.1 Scope

This document provides a description and progress report on the work performed in WP2. The development in WP2 is based on the specifications and requirements generated in WP1. Some parts of the technical descriptions from document D1.2 [AD02] are reused in the work descriptions in this report to make this document self-contained.

The document organised with main headings with the same names and in the same order as the tasks within WP2 [AD01]. The current development status is provided as subsections for each task with an overview of the work to be performed and the current status.

Progress will continue to be reported in updates of this document that will be issued as the D2.2 and D2.3 deliverables.

### 2.1.1 Applicable documents

| Internal code / DRL | Reference | Issue | Rev. | Title |
|---|---|---|---|---|
| AD01 | | | | Grant Agreement Project 1011112274 - ISOLDE |
| AD02 | D1.2 | 1 | 0 | Requirements and specifications on architecture, hardware and software modules and IPs |

### 2.1.2 Reference documents

| Internal code / DRL | Reference | Issue/Rev. | Date | Title |
|---|---|---|---|---|
| 1. | | | | |

### 2.1.3 Definitions and Acronyms

| Acronym | Definition |
|---|---|
| AHB | Advanced High-performance Bus |
| APB | Advanced Peripheral Bus |
| AXI | Advanced eXtensible Interface |
| CFI | Control Flow Integrity |
| CLIC | Core-Local Interrupt Controller |
| COP | Call-Oriented Programming |
| CSR | Control and Status Register |
| CVA6 | Core-V Application 6 stages processor |
| CV-XIF | Core-V eXtension InterFace |
| EDA | Electronic Design Automation |
| JOP | Jump-Oriented Programming |
| L2C | Level 2 Cache |
| NoC | Network on Chip |
| NOEL-V | (Not an acronym) |
| PWM | Pulse Width Modulation |
| ROP | Return-Oriented Programming |
| RTL | Register-Transfer Level |
| SoC | System on Chip |

# 3 Processor IP: Performance analysis, safety, and improvements

In this section, we describe the work on processor cores improvements, including extensions (e.g. for safety and security purposes) and core configurability (to support the various demonstrator needs). For each task, a short description and the status about the work already done is provided.

## 3.1 NOEL-V processor extensions (GSL)

The NOEL-V processor will be extended with security features defined in the Control-Flow Integrity (CFI) RISC-V extension. Work will also be performed towards supporting a trusted execution environment and the cryptography Extensions.

### 3.1.1 Support for shadow stack (Zicfiss) and Landing pad (Zicfilp) RISC-V extensions.

The RISC-V CFI extension has been identified as a desirable security feature. This extension helps defend against both Return-Oriented Programming (ROP) and Call/Jump-Oriented Programming (COP/JOP) attacks.

**Current status**

The NOEL-V implementation of these extensions is up to date with the current version of the specification. The specification is not yet ratified, so further updates are anticipated.

### 3.1.2 Support for RISC-V Cryptography extensions.

To support the increased demand for security, the RISC-V Cryptographic extension has been identified as a suitable feature for the NOEL-V processor. In this task this extension will be implemented in NOEL-V.

**Current status**

Currently no work has yet been done for this task.

### 3.1.3 Extend NOEL-V processor system to support trusted execution environment.

In the context of the ISOLDE project the RISC-V standardization of “World Guard” will be followed and evaluated. Following this, NOEL-V should be updated to support this.

**Current status**

Currently no work has yet been done for this task.

### 3.1.4 Generate IP-XACT description for the NOEL-V subsystem.

To support integration of the NOEL-V processor system into the demonstrator design an IP-XACT description will be generated. This will make it easier to integrate the processor using EDA tools like Xilinx Vivado.

**Current status**

The work is in its initial state and how this description should be generated is being decided.

## 3.2 CVA6 processor (TDIS)

The CVA6 processor will be extended to support post quantum cryptography extensions. To fulfil the different requirements expressed by the ISOLDE demonstrators, the CVA6 RTL will be configurable by enable/disable features.

### 3.2.1 Support for post quantum cryptography extensions

To support the increased demand for security due to quantum computing threat, support of post quantum cryptography is a must for the future. Additional instructions will be useful to support PQC algorithms standardised by NIST.

**Current status**

At this time, a working group at RISC-V International is starting to specify such instructions. Once specified, they will be implemented accordingly in CVA6.

### 3.2.2 Configurable RTL

The CVA6 RTL configurability is a mean to have cores adapted to the demonstrator needs. It will be possible to enable/disable and/or configure different features: e.g., privilege modes, extensions, cache configuration. By disabling some features, the

**Current status**

At this time, the CVA6 RTL configurability work is started, and some features are already made configurable.

### 3.3 Testing Design Parameters for CVA6 (UZL)

Creation of CVA6 of different multicore architectures by systems-on-chip generator tools. For this purpose, various design parameters are systematically created according to the specifications of T1.3/T1.4 and compared with each other using various design metrics (computing power, power consumption, etc.) and the suitable design parameters are identified.

#### 3.3.1 Setup of SoC Generator and CVA6 Configuration

The CVA6 will be embedded in a systems-on-chip design, which will generate by specialized tools. We examine different FPGA target platforms, based on the requirements of the demonstrators.

**Current status**

At this time, the target platform are chosen and the SoC-Generator Configuration is adapted accordingly.

#### 3.3.2 Choose of design parameter and benchmarking

Various design parameters for the CVA6 will be chosen and tested according to the specifications of T1.3/T1.4. The benchmarking compares different configuration with each other using various design metrics (computing power, power consumption, etc.) and the suitable design parameters are identified.

**Current status**

Currently no work has yet been done for this task, because it bases on the task III.1.

# 4 Peripheral and interconnect IP cores

Here, we provide the description of the main IP cores, including their firmware, that compose the catalogue of components that are usually required to build a System-on-Chip (SoC) successfully. This catalogue contains infrastructure IP cores mainly. Thus, compute power (processor IPs) is beyond the scope of it. We have structured the catalogue in two groups: peripherals and interconnects.

### 4.1 Peripherals

#### 4.1.1 GRLIB peripheral IPs (GSL)

GSL will provide the GPL version of the GRLIB IP library to support building the demonstrator SoCs. Required updates or extensions of the IP cores necessary for the demonstrators will be assessed in terms of feasibility and handled in support of the demonstrator platforms developed within ISOLDE project.

**Current status**

Latest release of the GPL version of GRLIB is available.

### 4.1.2 Improved L2C-lite (UPV)

GRLIB library from GSL includes a share-level cache module (L2C-lite) licensed as GPL to improve the performance of the open-source multicore architectures that can be built with this library. This cache module has an AHB interface as the front-end (to connect the cores) and supports both AXI and AHB as a back end to interconnect with memory or upper cache levels.

In the context of ISOLDE, the L2C-lite module will be improved to increase the level of parallelism supported. Thus, increasing the performance of the open-source multicores that can be built with GRLIB. At the same time, we will also extend the replacement policies of the L2C-lite module to support cache partitioning. This additional support will benefit those use-case needing to have time-predictable behaviour since it removes the inter-task interferences that will otherwise occur in the L2C-lite module.

**Current status**

First prototype in progress.

### 4.1.3 Timer (IFX)

The timer module shall support the following features

1. Watchdog
2. Clock
3. Time interval measurement
4. PWM signal generation

The interface may be alternatively via memory mapped registers or via CSRs or both. In that sense the timer may be an own IP-module or a RISC-V sub-module, later potentially merged with existing performance counter functionality.

Memory mapped registers may be accessible via AHB or via APB (decision pending). The provisioning – e.g. multiplexing – of control signals shall be done outside the timer.

The timer should have several timer channels, each with a size constrained counter, an optional number of same size capture compare units (CCUs) and a control module. Mapping of counter, CCU and control bit-fields to registers is not yet defined.

**Current status**

Requirements capture and concept for timer started. First prototype in progress.

### 4.1.4 Interrupt Controller (IFX)

The interrupt controller shall support priorities, priority groups and the passing of the active/newly requesting interrupt number to the RISC-V core.

The interface may be alternatively via memory mapped registers or via CSRs or both. In that sense the timer may be an own IP-module or a RISC-V sub-module, later potentially merged with RISCV CLIC.

Memory mapped registers may be accessible via AHB or via APB (decision pending). The provisioning – e.g. multiplexing – of control signals shall be done outside the timer.

The number of interrupts supported – and thus implicitly the maximum number of priorities is under discussion. Mapping of configuration bit-fields and the interrupt status bitfields to registers is not yet defined. Direct provisioning of interrupt number to exception handler is in discussion, whereas the CLIC way of doing is preferred.

**Current status**

Requirements capture and concept for timer started. First prototype in progress.

## 4.2 Interconnects

### 4.2.1 Wormhole NoC (UPV)

For SoC designs based on a Network-on-Chip (NoC) we envision a Tile-oriented design approach. Here, a Tile is an abstraction that holds one or more Intellectual Properties (IPs) and provides a common access interface and protocol to them. Through the NoC, the components located at different tiles will exchange messages in a unified manner. These IPs will interface to the NoC by means of a Network Interface (NI), which will expose different types of signal interfaces to satisfy the typical connections between two given components, like Initiator and Target. Thus, the NI will comply with AXI interfaces to connect external IP Cores to the network. The NoC router will support 2D mesh topologies. Overall, the NoC will provide the next features:

- Scalable on-chip architecture.
- Deadlock avoidance guarantee.
- Freedom of data losses.
- Broadcast and point-to-point communication primitives.
- Traffic partitioning.
- Quality of Service.
- High throughput.

**Current status**

We have completed the design of the architecture of the NI and the NoC Router.

### 4.2.2 AXI traffic sniffer (UPV)

To enable traffic monitoring in a wide variety of SoCs, we provide and integrate a hardware module that monitors the network activity in AXI interconnects. This module is written in VHDL, and its function is to leverage source information in the network packets to obtain contention information between sources. Thus, this module tracks the timing contention between the different traffic sources on an AXI network for each given destination.

In some cases, such as traversing a cache, initiator information is obscured on the AXI network, thus making fine-grained contention monitoring impossible. To solve this issue, the AXI traffic monitor relies on request initiator information propagated using the 4 bits available in the AXI QoS field.

The contention tracked by this module is propagated with specific interfaces to a timing interference hardware monitor using the safeSU's CCS signalling mechanism developed by BSC in the context of WP3.

The AXI4 contention monitoring infrastructure monitors read and write contention by blaming the head of its respective channel queues for the delay it causes to the tails of the queues. It can also monitor cross-channel contention when one channel's requests are blocked by another channel's requests being processed.

**Current status**

A baseline module is already available. Verification of this module to be initiated soon.

### 4.2.3 Context-Aware BUS (CA-BUS) - TRT

The context-aware bus (CA-BUS) extends existing AXI-buses so they can integrate the context information associated to the requests for their processing by the context-aware performance monitoring counters (CA-PMCs) developed in WP3.

Furthermore, as part of the CA-BUS development, this work will study the transmission of the context-aware augmented requests through IPs in the path between buses, like bridges and caches that can transfer requests between buses.

Figure 1 shows an integration example of the CA-BUS alongside the other related IPs for the context-aware monitoring developed in WP2 (CA-BUS, CA-CORE) and WP3 (CA-PMC, CA-PMC-IF). In Figure 1 the CA-BUS replaces the system bus, and a second dedicated bus to access to the CA-PMC-IF to access the monitors deployed in the system. An alternative design could suppress the dedicated bus and use the CA-BUS to also access the monitors.

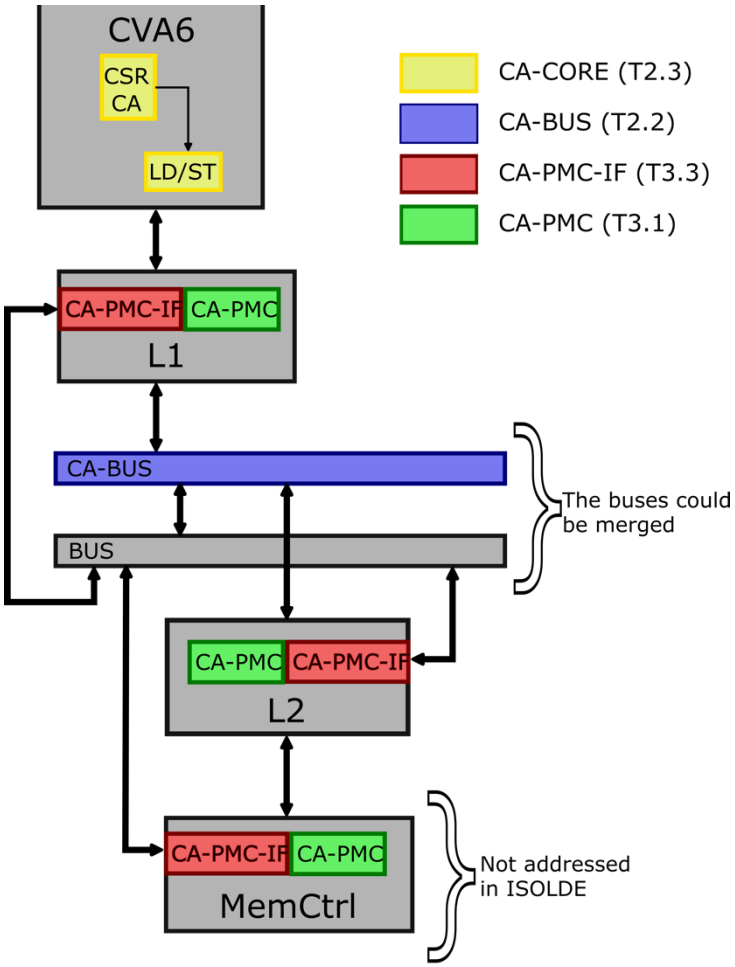


**Figure 1: context-aware monitoring infrastructure**

**Current status**

No progress has been reported for this task.

### 4.2.4 AHB Bridge (IFX)

The AHB crossbar prioritizes and multiplexes different initiators' and responders' read and write requests. Scalable address sizes and privileged mode are under discussion. Bust transfer should not be supported.

*Please note “master” and “slave” has been widely used, fortunately more and more these terms are replaced. “initiator” and “responder” are used here instead. As these terms are not fully aligned, we strive for Isolde consistent names.*

**Current status**

Requirement and concept work ongoing. No substantial implementations.

# 5 Common extension interfaces

In this section we describe the work on extending the processor cores to integration accelerators and support context aware performance counters. A short description on the different tasks will be provided together with an update on the progress currently made.

## 5.1 RISC-V accelerator interface for RISC-V core (UNITBV)

**Current status**

Work on this task has not yet started.

## 5.2 Memory bank accelerator interface for RISC-V core - XPERI

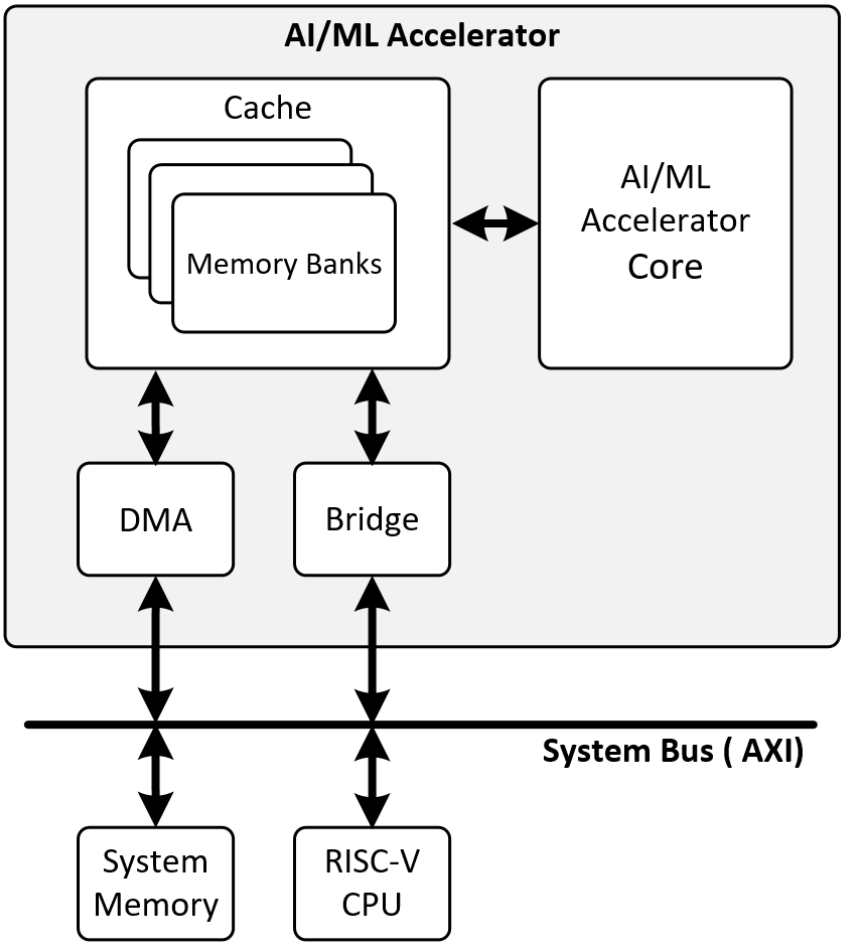


**Figure 2: Memory bank accelerator interface**

The purpose of the AXI to Memory banks bridge is to bypass costly transactions to and from the system memory. Many operations could be performed by RISC-V accessing directly the memory banks, without the need to transfer the data through the system memory.

For example, we consider a SoftMax operation that must be applied on the results of a previous operation. This result resides in the memory banks.

1. Without the bridge processing has the following steps:
    a. Use DMA to transfer data to the system memory. Transfers to and from the system memory can be very expensive operations in clock cycles and power, especially if an off-chip DDR memory is used.
    b. The RISC-V processes the data from the system memory and writes the results back to the system memory.
    c. Data is transferred back to the memory banks using DMA, so the accelerator can process it further.
2. The optimized approach uses the bridge and requires only two transfers (Memory Banks → RISC-V and then RISC-V → Memory Banks), instead of four transfers used previously.
    a. The CPU reads data directly from the memory bank and processes it.
    b. The CPU writes the results back to the memory banks.

**Current status**

Requirements were defined, design refinement and implementation activities are pending.

## 5.3 Context-Aware Core Extension (CA-CORE) - TRT

The context-aware core extension (CA-CORE) will upgrade the RISC-V CVA6 core to provide the context information required for the exploitation of the context-aware performance monitor counters (CA-PMC) developed in WP3.

The CA-PMCs will enable fine-grained monitoring controlled by the software, typically the OS or the Hypervisor, which will be capable of programming the CA-PMCs to monitor the incoming request based on their context. The CA-CORE extension extends the RISC-V CVA6 with the required mechanisms to define the context of the software currently executing in the core.

The CA-CORE extension adds a dedicated CSR to keep the context of the software currently executing in the core. The OS or hypervisor will change the value of the context CSR depending on the running application or VM. The CA-CORE extension uses the value in the CSR context to associate the context value in the CSR with all the load/store requests that core will issue. For that purpose, we modify load/store-unit to create requests with the context information. These context-aware requests are transmitted through the system with the CA-BUS (see Section “Peripheral and interconnect IP cores” - CA-BUS) to enable their processing in the CA-PMCs developed in WP3.

Figure 1 (see section 4.2.3 “Context-Aware BUS (CA-BUS) - TRT”) shows an integration example of the CA-CORE alongside the other related IPs for the context-aware monitoring developed in WP2 (CA-BUS, CA-CORE) and WP3 (CA-PMC, CA-PMC-IF).

## 5.4 Integration and test of accelerator cores with NOEL-V (GSL)

The NOEL-V processor will be updated to support an extension (co-processor/accelerator) interface. In the context of the ISOLDE project, standard extension interfaces will be evaluated (in collaboration with OpenHW group and the TRISTAN project) and finally demonstrated the integration of the ETHZ vector accelerator developed within WP3 (and potentially other accelerators).

During the project the OpenHW group CV-X-IF interface will be evaluated for integration with NOEL-V. A test implementation will be made to determine if some aspects of the specification should be updated. We will work towards an updated specification more suitable to be implemented in different processor architecture supporting a variety of accelerators.

The final stage in this development is to integrate the vector accelerator form ETHZ with the NOEL-V processor and show vector performance improvements.

**Current status**

Currently the collaboration with OpenHW group and partners from the TRISTAN project has been started.

### 5.5 Common coprocessor interface for CVA6 (TDIS)

The current implementation of the CV-X-IF on CVA6 processor compliant to version 1.0 will be revisited to support CV-X-IF version 2.0. The specification of this version will be the outcome of a joint work with OpenHW Group members and TRISTAN project partners. This collaboration has been started.

CVA6 modifications will be evaluated during the CV-X-IF specification development to ensure the needs of the different possible coprocessors (e.g., post quantum cryptography coprocessors, vector coprocessor, security coprocessor) will be met.

The CV-X-IF on CVA6 processor will be verified using UVM.

**Current status**

Currently the collaboration with OpenHW group and partners from the TRISTAN project has been started.

# 6 Software interfaces to general purpose cores

### 6.1 XNG RISC-V BSP to support new NOEL-V features (FEN)

The XNG (Xtratum Next Generation) RISC-V BSP (Board Support Package) will be updated to support the new standards of RISC-V architecture implemented by NOEL-V. In the context of the ISOLDE project, this new XNG RISC-V BSP will also support the new interrupt controller version that will be developed inside the scope of this project.

**Current status**

Currently the analysis of the requirements of the new features which are going to be supported has been started.

### 6.2 NOEL-V software tools (GSL)

GSL will update and provide bare-metal and Linux RISC-V tool chains for NOEL-V. Work will also be performed to extend NOEL-V software support.

**Current status**

Current versions of the NOEL-V tool chains are available for download.

### 6.3 System software support (SYSGO)

Test hardware developments; produce open-source hardware demonstrations / device drivers.

**Current status**

We are working on defining validation environment, e.g. investigate to use OpenPiton or similar to synthesize multicore environments.