FRACTAL

Deliverable

D3.5 Final Fractal hardware node and support

Deliverable Id:	D3.5
Deliverable Name:	Preliminary FRACTAL hardware node and support
Status:	Completed
Dissemination Level:	Public
Due date of deliverable:	31. August 2022
Actual submission date:	31. August 2022
Work Package:	WP3
Organization name of lead contractor for this deliverable:	ETH Zurich
Author(s):	Frank K. Gürkaynak, ETH Zürich
Partner(s) contributing:	Jérôme Quévremont, Thales Alexander Flick, PLC2 Jaume Abella, Ramon Canal, BSC Edurne Palacio, Carles Estrada, IKER Carles Hernandez, UPV Bekim Chilku, Siemens Iñaki Paz, LKS Tania Di Masco, Giacomo Valente, UNIVAQ Stefano Delucchi, AITEK Juan Garcia Enamorado, Qualigon Marco Cappella, Mario De Biase, Modis Paolo Burgio, Giacomo Brilli, UNIMORE Michael Gautschi, ACP

Abstract:
This deliverable updates D3.1/D3.3 and describes the different hardware nodes of the FRACTAL project that will be used to in the technical WPs to develop different services of FRACTAL, explain how they are being utilized for different Use Cases and present an up to date status.

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) and co-funded by the Horizon 2020 Programme of the European Union under grant agreement No 877056

Contents

History

Version	Date	Modification reason	Modified by
v0.1	07.July.2022	Initial skeleton	Frank Gürkaynak
v0.2	20.July 2022	Updates with new 'big picture'	Frank Gürkaynak
V0.5	22.Aug.2022	Consolidated for review	Frank Gürkaynak
V1.0	29.Aug.2022	Finalized version after review	Frank Gürkaynak

1 Summary

The main objective of the FRACTAL project is to "*create a cognitive edge node enabling a fractal Edge that can be qualified to work under different safety-related domains*". Furthermore, it is stated in the DoA that "*This computing node will be the basic building block of intelligent, scalable and non-ergodic IoT*". As such the hardware node is a central part of the FRACTAL project around which 28 partners collaborate, investigate and industrial partners develop their use cases.

This deliverable (D3.5) is the final deliverable and follows up on the "Preliminary FRACTAL HW node and support" (D3.1) from M12 and "Intermediate FRACTAL hardware node and support" (D3.3) from M18 and describes work done within the FRACTAL project on the hardware of the FRACTAL node. These three deliverables are also paired with the "software node and services" deliverables D3.2, D3.4 and D3.6.

The FRACTAL project brings together a large number of partners (28) both from the industry and academia, working on varied and challenging topics as well as eight industrial use cases. It was already a challenging task to provide a set of solutions for the hardware node in this context and combined with restrictions around COVID and worldwide supply disruptions for electronic components, partners in WP3 had to face additional challenges.

In the original plan two main options for the hardware node were foreseen:

- **Commercial node** based around the Xilinx VERSAL ACAP (Adaptable Compute Acceleration Platform)
- **Customizable mode** based around the open-source RISC-V based PULP platform

These two main nodes continue to form the backbone of the developments and implementations for the FRACTAL project, but to accommodate practical needs and requirements of project partners additional (related) platforms were also leveraged when necessary.

The organization of the deliverable is as follows. Section 2 provides a general introduction to the hardware nodes and following the discussions around D2.1 "Platform Specification (a)" clarifies the role of hardware platforms in FRACTAL for use cases. Setcion 3, then summarizes the current state of the hardware platforms, and Section 0 describes how the technical developments in work packages WP4/5/6 make use of the hardware nodes being described in this deliverable. Section 0 lists the plans for the use case providers and the hardware nodes described in their use cases and finally Section 6 provides conclusions for the WP3 activities of FRACTAL on Hardware Node development.

2 Introduction

FRACTAL is an ambitious project to design a cognitive edge node that is capable of learning how to improve its performance against the uncertainty of the environment. In the project proposal, we had identified four strategic objectives of FRACTAL to reach this goal:

1 **Objective 1**: Design and Implement an Open-Safe-Reliable Platform to Build Cognitive Edge Nodes of Variable Complexity. This part is mainly being addressed as part of WP3.

2 **Objective 2**: Guarantee extra-functional properties (dependability, security, timeliness and energy-efficiency) of FRACTAL nodes and systems built using FRACTAL nodes (i.e., FRACTAL systems), which has determined the tasks of WP4

3 **Objective 3**: Evaluate and validate the analytics approach by means of AI to help the identification of the largest set of working conditions still preserving safe and secure operational behaviours, which is the topic of WP5

4 **Objective 4**: To integrate fractal communication and remote management features into FRACTAL nodes, which will be covered by WP6.

While the development of both the hardware and software of the node architecture is the primary goal of WP3 that addresses Objective 1 it can be seen that the node plays an essential part of the developments of the other objectives as the hardware nodes are meant to be the vehicles where technical developments in WP4/5/6 are expected to be demonstrated and their contributions validated as part of WP7/8.

As can be seen in Figure 1 the FRACTAL system can be viewed in three layers, the Application, Orchestration and the Node layer. The FRACTAL features are supported by all these layers and give the FRACTAL system its characteristics.

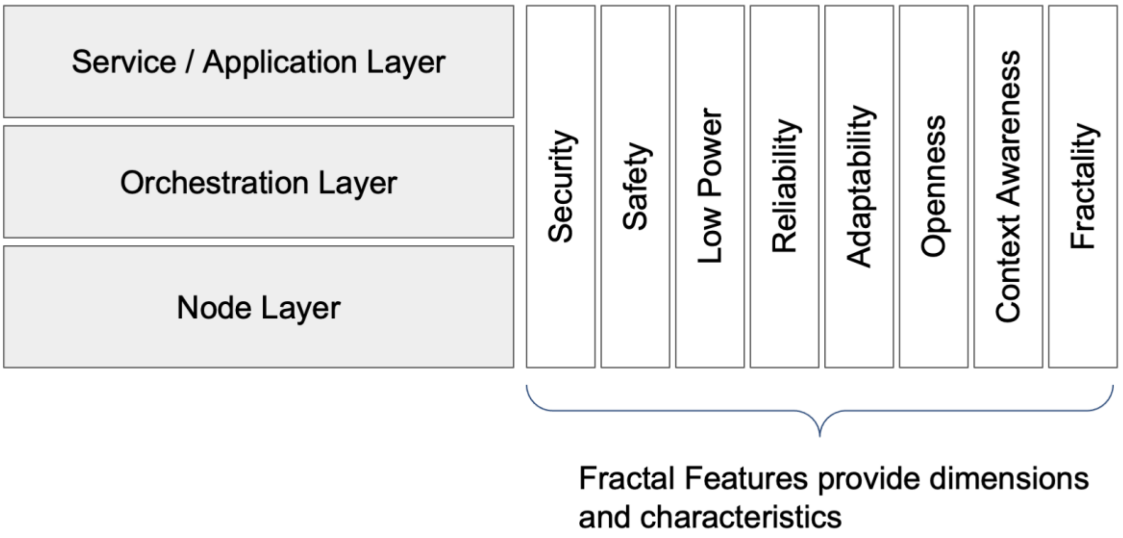


Figure 1 High-level organization of FRACTAL systems

Copyright © FRACTAL Project Consortium

This deliverable deals with the hardware node that is part of the node layer and from the beginning of the project a key aspect was to identify a hardware platform that could lead to commercialization within time and cost limitations of the project. It was important to offer a mature platform to end-users for the integration and assessment of their use cases already at the start of the project, as well as a relatively short path towards commercialization of the FRACTAL approach. Project partners had already identified Xilinx VERSAL platform supported through FRACTAL partner PLC2 as the most suitable SoC system as the **commercial hardware node**. As a commercial system with significant resources, Xilinx VERSAL is able to fulfil the performance requirements of even the most challenging use case and project considered within FRACTAL while offering a mature development environment based around an industry standard design flow.

While the commercial platform offers many advantages, especially in short-term commercialization efforts, such a platform presents some well-established solutions, and the customization options are limited within the processing system (known as PS within a Xilinx MPSoC system). In order to explore, practically without any additional constraints, a **customizable node** was added. Designed around the open RISC-V instruction set architecture (ISA) and based on the open-source PULP platform (maintained by partner ETHZ) the customizable node allows FRACTAL partners a powerful and flexible starting point for the development of the custom node and a viable path for longer-term product development without an early commitment to a proprietary ISA and platform. The customizable node was also offered in an extremely flexible FPGA-based development platform where resources in the node, as well as their organization, can be adapted as needed to enable a larger range of trade-offs.

The status of both of these platforms will be discussed in Section 3 with some detail. Moreover, at the beginning of the project, especially during the work done for D2.1 “Platform specification (a)”, it became clear that especially in the initial stages of the project the official FRACTAL nodes had to be augmented by other complementary platforms as well. There are two main reasons for this:

- Supply chain issues partially as a result of global response to COVID related restrictions, has limited the availability of Xilinx VERSAL platforms, with few partners having access to the node within the first year. This is of course a temporary issue and all FRACTAL partners that need access to a Xilinx VERSAL platform are expected to get one sooner than later.
- While the customizable node offered an interesting alternative, especially as it supported a RISC-V based open-source solution, some of the use case requirements were geared towards higher end solutions that exceeded the capabilities of the IoT based platform made available. Some partners realized they could use other RISC-V based platforms and still remain compatible with the FRACTAL platforms in the longer term. We will describe these platforms and their rationale in Section 0.

Following the first project review of FRACTAL and feedback from the Industrial Advisory Board, it was decided to provide a big picture of FRACTAL components and highlight the developments reported in deliverables in relation to this big picture as part of Section 2.1 of this deliverable.

The FRACTAL consortium has generated a comprehensive list of components that will be contributed by all partners as part of the WP2 activity. This deliverable has been updated (where applicable) with references to the components listed in D2.3.

Table 1 The FRACTAL components (according to D2.3) related to WP3. Some components will be described in D3.6 (and one in D7.3), for others the section in this deliverable is given

Component ID	(Sub)Component / Development Name	Partner	Deliv.	Section
WP3-AI	**AI accelerator (hardware and software support)**			
WP3T32-01	HW accelerator (SIEFRACC)	SIEM	D3.5	5
WP3T32-05	ML inference demo PULPissimo	ETHZ	D7.3	
WP3T32-07	Age and Gender identifier at the edge	UNIVAQ	D3.5	4.4
WP3T32-10	VERSAL accelerator building-blocks	IKER	D3.5	3.1
WP3T34-03	Versal Model deployment layer	PLC2	D3.5	3.1
WP3T35-01	SW driver for HW accelerator	SIEM	D3.6	
WP3T35-02	Accelerator Adaptation to AI library	UPV	D3.5	3.3.1
WP3T35-03	LEDEL (Low Energy EDDL)	SML	D3.6	
WP3T35-04	Deep learning based automatic iris diagnosis	MODIS	D3.6	
WP3T35-05	Idiom Recognition	UNIGE	D3.6	
WP3-CPU/OS	**CPU and OS support**			
WP3T32-02	PULPissimo platform for IoT applications	ETHZ	D3.5	3.2
WP3T32-02b	Ariane for Linux capable RISC-V platform	ETHZ	D3.5	3.3.2
WP3T32-03	PULP trainings	ETHZ	D3.5	3.2
WP3T32-04	FreeRTOS port to PULP	ETHZ	D3.6	
WP3T32-08	Real-time aware caches	ACP	D3.5	4.2
WP3T32-11	Smart Interrupt distribution system	ACP	D3.5	4.2
WP3T32-12	Security services - TL2AXI adapter	ACP	D3.5	5
WP3T33-03	CVA6 (former Ariane) RISC-V core	THA	D3.5	3.3.2
WP3T36-01	Linux for CVA6 (former Ariane)	THA	D3.6	
WP3T36-02	Load Balancing Module	MODIS	D3.6	
WP3T36-03	Nuttx on PULP	OFFC	D3.6	
WP3-Safety	**Safety and security features for CPU**			
WP3T31-01	Edge-oriented monitoring unit	BSC	D3.5	4.2
WP3T31-02	Interconnect to support Accelerators integration	UPV	D3.5	3.3.1
WP3T31-03	Safety and security hardware support	UPV	D3.5	4.2
WP3T32-06	Redundant Acceleration Scheme	UPV	D3.5	3.3.1
WP3T32-09	Runtime Bandwidth Regulator	UNIMORE + UNIVAQ	D3.5	4.2
WP3T34-01	Driver for the edge-oriented monitoring unit	BSC	D3.6	
WP3T34-02	Drivers for the SW diverse redundancy library	BSC	D3.6	

Table 1 shows all the components associated with WP3 and highlights those components that are described within D3.5. The majority of the components not covered in this deliverable will be part of D3.6.

The goal of WP3 is to provide the HW nodes, and how the FRACTAL hardware nodes would be demonstrated in use cases will be discussed in the Section 2.2. Later in the document Section 5 will provide an up-to-date description of individual solutions for hardware nodes for use cases.

2.1 The hardware nodes as part of the big picture for FRACTAL

The *big picture* for FRACTAL was developed as part of the second technical workshop of FRACTAL in February 2022. This diagram shown in Figure 2 captures all main components and their interactions within FRACTAL.

From top to bottom the figure captures three main aspects

- FRACTAL services in the cloud
- Software components of FRACTAL on edge nodes (SW Edge)
- Hardware platforms used in FRACTAL (HW Edge)

The diagram has customizations for individual hardware nodes (Versal, Noel-V, CVA6, PULPissimo) as well as individual use cases. This section will describe the 'Hardware platforms used in FRACTAL' that is marked as HW Edge in the diagram.

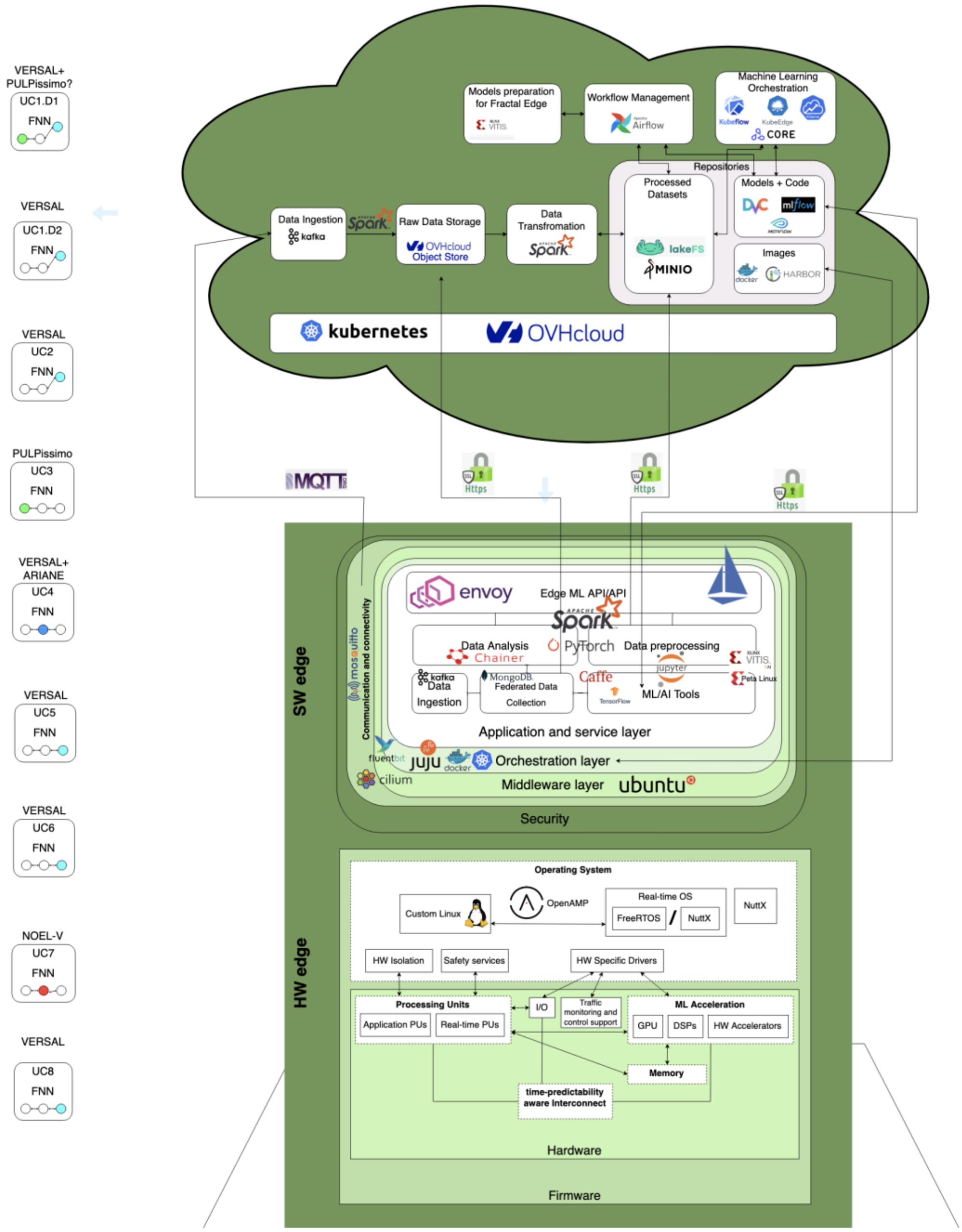


Figure 2 The FRACTAL big picture that was developed part of the 2nd technical workshop of FRACTAL

Figure 3 provides a closer look at the HW edge part of the diagram. In the following, we will briefly explain the individual components and the individualized version of the big picture for hardware nodes will be used in Section 3.

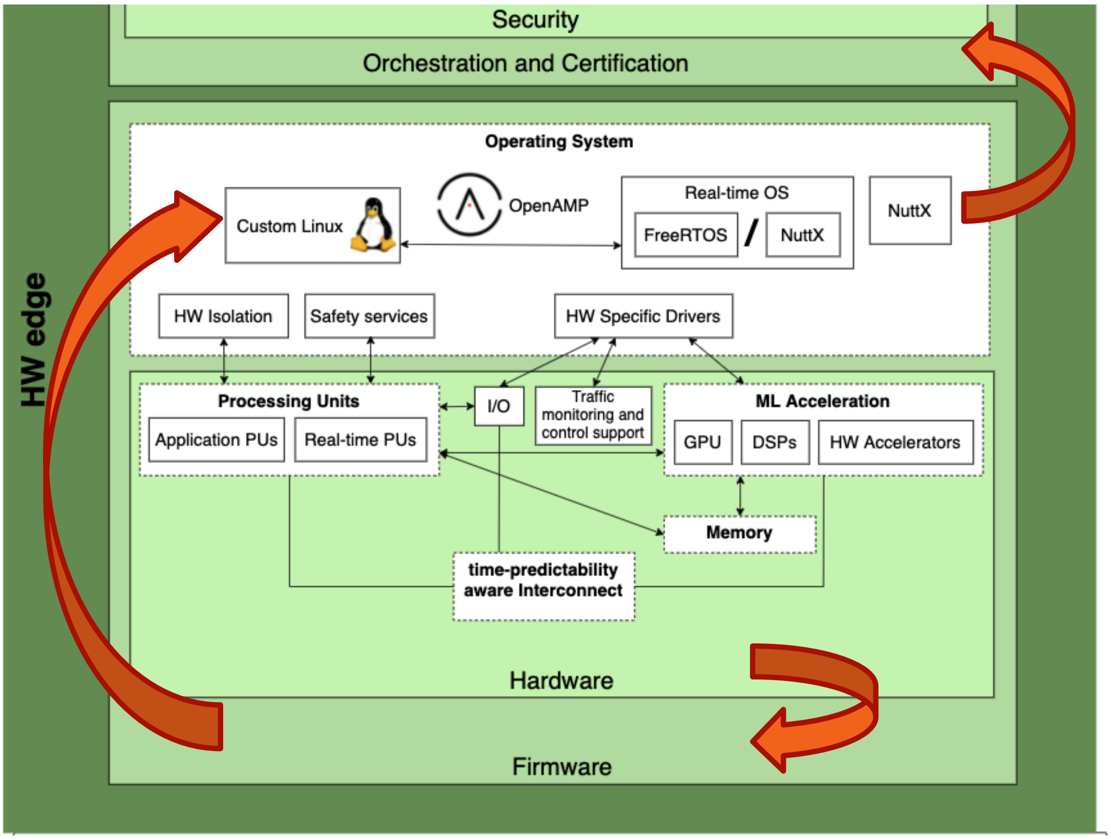


Figure 3 The hardware node part of the Big Picture for FRACTAL. To adapt to the overall FRACTAL description, this part of the graph has been layered differently and arrows have been added to guide the viewer that are familiar with a more traditional view that orders hardware, firmware, software

FRACTAL relies on a wide variety of services in the cloud, their counterparts on edge nodes and the big picture has been designed to be able to capture the significant work in this part. The lower part of the graph shown in Figure 3 maintains this software centric view and coils around the hardware capabilities in the center (arrows in the figure show the flow from low level hardware to software through the firmware).

We expect aspects of the Big Picture to consolidate over the next months, and adaptations will be made in D3.5 to reflect these. In the current version of Big Picture, starting from the innermost part, the following components have been identified:

- **Processing Units (PU)**: All FRACTAL nodes rely on at least one processor core that allows the FRACTAL software components to run.
  - **Application PUs**: are general purpose processors that do not have specialized features (for example for real-time guarantees).
  - **Real-time PUs**: processing units that have specialized features for predictable execution that allow them to be used in control applications.
- **ML Acceleration**: A significant portion of the FRACTAL improvements relies on efficient processing of machine learning related workloads. The *ML acceleration* units are optional add-ons to the hardware node that allow these

loads to be executed more efficiently (in terms of execution speed, energy efficiency or both) than when executed on *Application PUs*.

  - **GPU**: Traditionally, Graphical Processing Units have been specialized for parallel execution which have found great use for regular computing tasks such as inference.
  - **DSPs**: Digital Signal Processing units are enhancements to Application PUs that add instructions that make them more suited for typical tasks used in digital signal processing, such as using smaller bit-width numbers (8bit, 16bit integers), operations that parallelize multiple small width operations (dot product, single instruction multiple data extensions) and operations that fuse operations (multiply accumulate).
  - **HW Accelerators**: For domain specific cases, performance can be improved by specialized hardware that has been optimized to handle these operations. In a typical application, *Application PUs* will assign parts of the computation to a dedicated *HW accelerator*.

- **I/O**: The goal of any information processing system is to interact with its environment. The *I/O* block in the *big picture* encompasses all such interactions, including communication infrastructure (ethernet, Wi-Fi, Bluetooth), sensors (cameras, microphones), actuators (motors), user input (keyboard), display systems. While there is a great variety in the *I/O*s, there are no specific developments in FRACTAL related to these components, hence it is abstracted as a rather simple box.
- **Memory**: All processing in hardware requires memory. Similar to *I/O*, FRACTAL does not directly introduce improvements to memory subsystems, however there are FRACTAL services that ensure safety which could involve changes to the memory systems
- **Traffic Monitoring and control support**: part of ensuring safety is the ability to monitor and control the data traffic in the hardware node.
- **Time-predictability aware interconnect**: A significant part of the work in WP4 is concentrated on providing an on-chip network that can ensure certain timing limits are kept

The big picture of FRACTAL also contains SW components that closely interact with the hardware namely:

Operating system: A large portion of the use cases assume the presence of a **Linux** operating system **customized** to run on the hardware node (i.e. *Petalinux* on VERSAL, a standard *Debian* distribution on CVA6). The *OpenAMP* framework is utilized for hardware nodes that can support multiple cores and accelerators. Some applications can make use of a lighter weight system like ***FreeRTOS*** or ***NuttX*** especially for real-time applications. In addition, *NuttX* can also be used as a regular operating system and not only for real-time applications.

HW Isolation: Part of the security applications requires that tasks running on the system could be isolated from each other without leaving traces of their activity during context/task switches.

Safety Services: Additional drivers and hardware capabilities can be added to hardware nodes to make sure that the node operates as intended. These could include, watchdog timers, error counters, tagging systems, and return address protection.

HW specific Drivers: All additional capabilities within a FRACTAL hardware node will need drivers added to the software environment to be able to access these features.

2.2 Role of HW platforms in FRACTAL Use Cases

The comprehensive discussions with all FRACTAL partners during the preparation of D2.1 "Platform specification (a)" showed a number of issues with the initial approach regarding how FRACTAL hardware nodes will be demonstrated as part of the use cases.

As the use case is a concrete demonstration for the use case provider, it should not be surprising that the main goal of the use case provider is to make sure that the use case can run without issues within the FRACTAL project. As a result, some project partners expressed rather extensive requirements for their own use cases in order not to be limited by the hardware capabilities in the future and some others expressed interest in using systems that they are more familiar with. In practice this has led to several use case providers stating the need for a symmetric multi-core system running a standard Linux distribution.

In all cases, these requirements are perfectly understandable and most of them could be implemented using the commercial node of FRACTAL, the Xilinx VERSAL platform. As outlined in Section 3.2 the basic customizable node has been targeted towards simpler IoT applications and lacks the 'horsepower' to fulfil several of these requirements. At first sight this creates an apparent imbalance of utilization between the commercial and the customizable node.

FRACTAL partners have discussed various approaches to provide a solution and have decided on a few measures to make sure that ideas developed as part of FRACTAL are validated on common platforms that are available to all project partners and have made the following recommendations.

- There are several use cases (a detailed breakdown per use case is given in Section 0) that are content to use the FRACTAL hardware nodes as provided.
- FRACTAL has identified three tiers of FRACTAL hardware nodes: low (mist), medium (edge), high (cloud) versions that all share similar interfaces and interact with each other. Figure 4 shows such an organization where simpler nodes are acquiring data and delegating more complex tasks to nodes with higher complexity. The figure is meant as an example, and different allocation of tasks are currently under discussion within WP5/6. In this model, the commercial node covers the higher-end version, while the customizable node is seen as the lower-end version. As described in Section 3, partners have suggested several alternatives for the medium-end nodes.

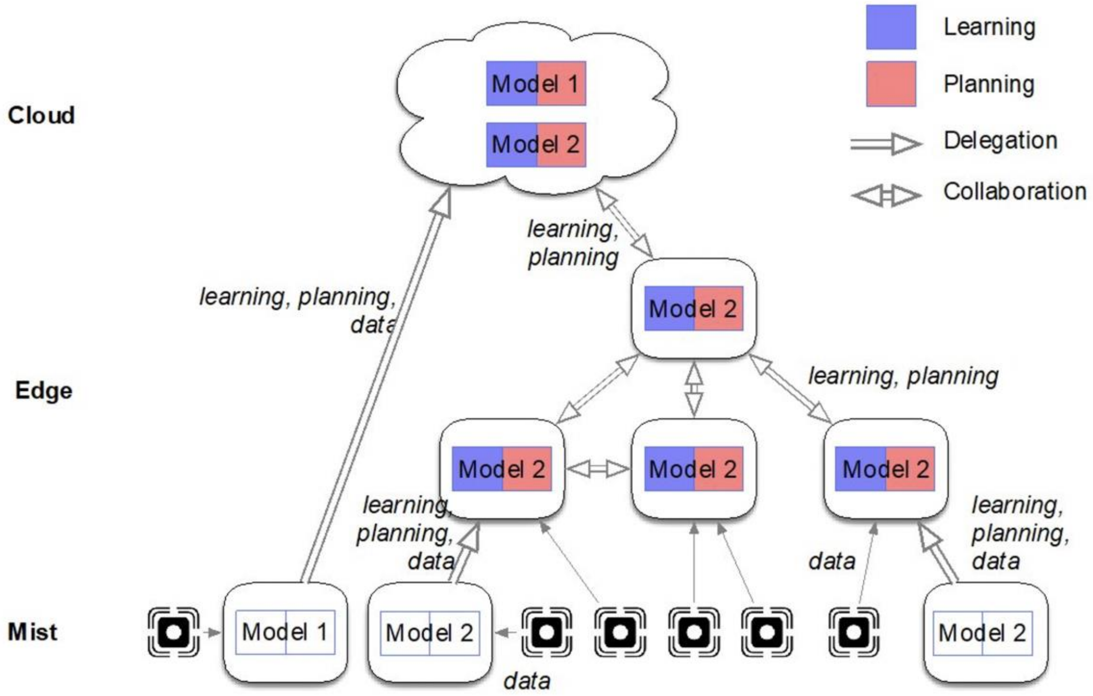


Figure 4. A schematic drawing of a possible FRACTAL system deployment using three different tiers of FRACTAL hardware nodes with different capabilities (drawing from WP5 technical meetings).

- Some partners are relying on their prior work and experience to implement some of their contributions. Most of these are based on hardware systems that are similar and/or compatible with FRACTAL nodes but have some differences. These include implementations in earlier models of Xilinx MPSoC platforms rather than the VERSAL, as well as other openly available RISC-V systems. Out of practical considerations, FRACTAL partners have added these as additional platforms to the initially identified hardware nodes.

It was also recognized that official FRACTAL nodes form a basis for research aspects involving developments in WP4/5/6 and the experience from these explorative works could then be used to evaluate the potential of these developments in use cases that use more traditional solutions. As a concrete example, novel safety solutions with hardware support could be explored on a small scale in the customizable node as part of WP4. The results of this exploration could then be used to directly estimate the gains achievable by this approach in a use case that used an alternative hardware node. The work done throughout the first part of the project allowed partners to experiment with different possibilities and showed that the key point was that all developments from FRACTAL technical work packages should be accessible for all FRACTAL partners. While the initially identified Hardware Nodes cover a large range of the specification spectrum, partners could also make use of additional hardware nodes as long as this work could be used/verified/evaluated by all partners.

Another observation that WP3 partners were able to make was that, for some FRACTAL features to be demonstrated and developed the choice of the platform made little difference. As shown in the FRACTAL Big Picture (Figure 2) a significant portion of the FRACTAL features rely on software components that are able to run on several different platforms. An example would be a use case that wants to implement a deployment of updated weights for inference for a large number of nodes. While WP5 work is essential for this use case, the actual transmission of the data will be handled using additional boards that will be connected to the hardware node using a ubiquitous interface (i.e., SPI, I^2C) which could be realized with dozens of different platforms. For such use cases, some partners have elected to concentrate their efforts on refining the FRACTAL related aspects of the work (i.e., the deployment mechanics) rather than spending the engineering effort required to adapt FRACTAL hardware nodes to fit into their current setups. While WP3 efforts concentrated as the primary alternative for deploying FRACTAL features, we also recognize the practical aspects for certain cases.

3 HW nodes provided for FRACTAL

The FRACTAL project relies on a set of hardware nodes to demonstrate the FRACTAL approach, especially the technical developments in WP4/5/6, and allows all partners to be able to experiment and use these developments in their own environment.

The two main hardware nodes from FRACTAL were already identified prior to the start of the project as:

- **Commercial node** based around the Xilinx VERSAL platform, a high-end FPGA platform with state of the art acceleration engines for machine learning applications.
- **Customizable node** based around the RISC-V based PULP platform, geared towards more IoT domain that allows the partners to experiment not only with the surrounding system, but all aspects of the platform, including instruction set externsion.

To support these, PLC2 and ETHZ have joined the FRACTAL consortium and have been assisting partners to find solutions with the respective platforms. As the project progressed, to cope with the large variety of requirements and to leverage existing prior work, additional nodes that remain compatible with the FRACTAL project goals have been identified.

This deliverable (D3.5) completes the updates to earlier versions of the deliverable (D3.1 and D3.3) and describes the state of the hardware nodes used within FRACTAL. With D3.5 the development of the hardware nodes as part of WP3 has also reached its conclusion. However, the nodes that have been developed continue to be supported, improved and adapted in WP7/8 as part of the UC developments.

3.1 The commercial node (Xilinx VERSAL)

The Xilinx VERSAL ACAP is expected to be deployed as part of the VCK190 Evaluation Kit board, which provides support for several I/O interfaces and memory devices. The VERSAL architecture combines different engine types with a wealth of connectivity and communication capability and a network on chip (NoC) on a configurable platform to enable seamless access to the full height and width of the device.

As the latest generation Xilinx products, the VERSAL series provides impressive performance while retaining its programmability. The VC1902 ACAP Xilinx FPGA device that the VCK190 evaluation board[1] is centred around, boasts 400x AI engines, almost 2,000 DSP engines, close to 2 million system logic cells and 900,000 LUTs that are traditionally used to map custom logic as well as 191 Mb of internal memory. The FPGA has 4x 256-bit DDR memory controllers, as well as 4x PCIe4.0, 4x 100G

[1] https://www.xilinx.com/products/boards-and-kits/vck190.html

Copyright © FRACTAL Project Consortium

Ethernet and one CCIX serial interface giving tremendous amount of bandwidth that should satisfy even the most demanding applications.

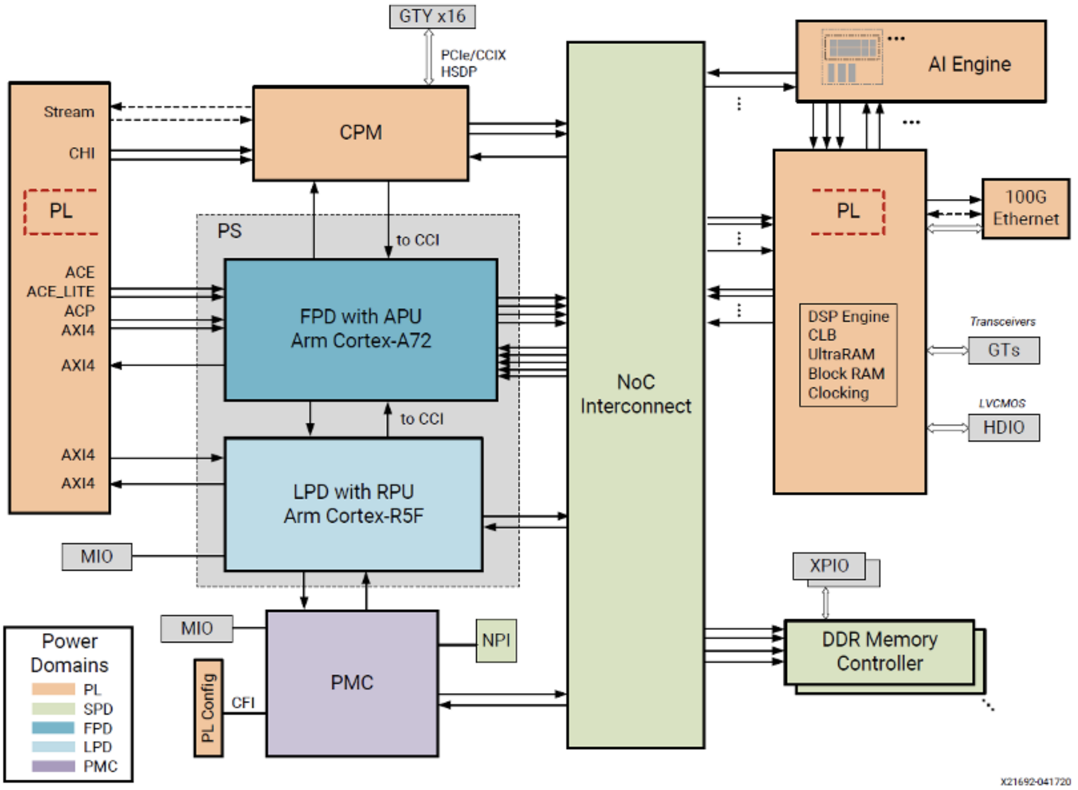


Figure 5. Top level schematic of Xilinx VERSAL Platform

Like the earlier Xilinx Zynq MPSoC products VERSAL ACAP devices still offer the two main components.

- **Processing System** (PS) consists of a dual high-performance ARM Cortex A72 cores that can run Linux or other operating systems. This system is augmented by a dual-core ASIL-C certified real-time processing subsystem based on ARM Cortex R5F cores. Together these systems address the needs of most modern computing needs using a traditional programming interface.
- **Programmable Logic** (PL) allows this system to be augmented by hardware accelerators customized to a particular compute function, which makes them best at latency-critical real-time applications (e.g., automotive driver assist) and irregular data structures (e.g., genomic sequencing).

What sets VERSAL ACAP devices apart from conventional FPGA approaches is the hardened IP platform to provide highly configurable connectivity and infrastructure to drive all computation features. On this platform each of these computation engines are deployed to best serve specific computation models to support the full application.

The compute cluster of ARM cores in the PS part constitutes the **Scalar Engines** to help with complex sequential algorithms. A further type of these engines is the **Adaptable Engines** (AE), which are made up of traditional FPGA programmable logic (see PL above) providing local memory in connection with the next generation of the industry's fastest programmable logic.

The **Intelligent Engines** are available for high computation workloads and are available with the PL side digital signal processing blocks (DSP engines) or the AI Engines (AIE) as a specific IP block within specific device family members. The AI Engines are set up as an array of innovative very long instruction word (VLIW) and single instruction, multiple data (SIMD) processing engines and memories. These permit 5X–10X performance improvement for machine learning and DSP applications. The AI Engine processors deliver more compute capacity per silicon area versus PL implementation of compute-intensive applications. AI Engines also reduce compute-intensive power consumption by 50% versus the same functions implemented in programmable logic and also provide deterministic, high-performance, real-time DSP capabilities. Because the AI Engine kernels can be written in C/C++, this approach also delivers greater designer productivity. Signal processing and compute-intensive algorithms are well suited to run on the AI Engines.

The VERSAL devices provide the hardened NoC that connects these engines together providing an aggregate bandwidth of 1Tb/s+. In addition to the NoC, the massive memory bandwidth enabled by programmable logic enables programmable memory hierarchies optimized for individual computing tasks.

The memory is devised in form of a hierarchy to improve timing performance and energy consumption by exploiting temporal reusability of Convolutional Neural Network’s (CNN’s) parameters. This is achieved through small and fast buffer memories located near the Processing Elements (PEs). While one buffer supplies the PEs with data the other one prefetches the anticipated data from DRAM and vice versa. PEs also have a local memory in the form of registers to keep the current input and output data. Each operation on PE requires at least two memory reads and one memory write. If all these accesses are performed directly on the off-chip DRAM memory the generated latency and the amount of consumed energy would make the accelerator not adequate to deal with high computation workload of CNN.

In general, the choice of acceleration hardware, whether PL or AI Engines, depends on the type of algorithm and data ingress and egress paths. Scalar Engines provide complex software support. Adaptable Engines provide flexible custom compute and data movement.

As VERSAL provides many different implementation possibilities, in the beginning of the project the main approach was not only to analyse, understand and determine the requirements coming from every UC, but also the proposed roadmap to achieve use case objectives. These inputs will potentially determine the required hardware development needed for providing cognitive awareness to FRACTAL node based on VERSAL platform. Reference architecture of a cognitive edge computing node with FRACTAL properties will be defined and a common repository of generic qualified components will be set up. Particular attention will be paid on providing flexible computing nodes, that are reusable by others and support efficiently the software by providing acceleration for the learning part.

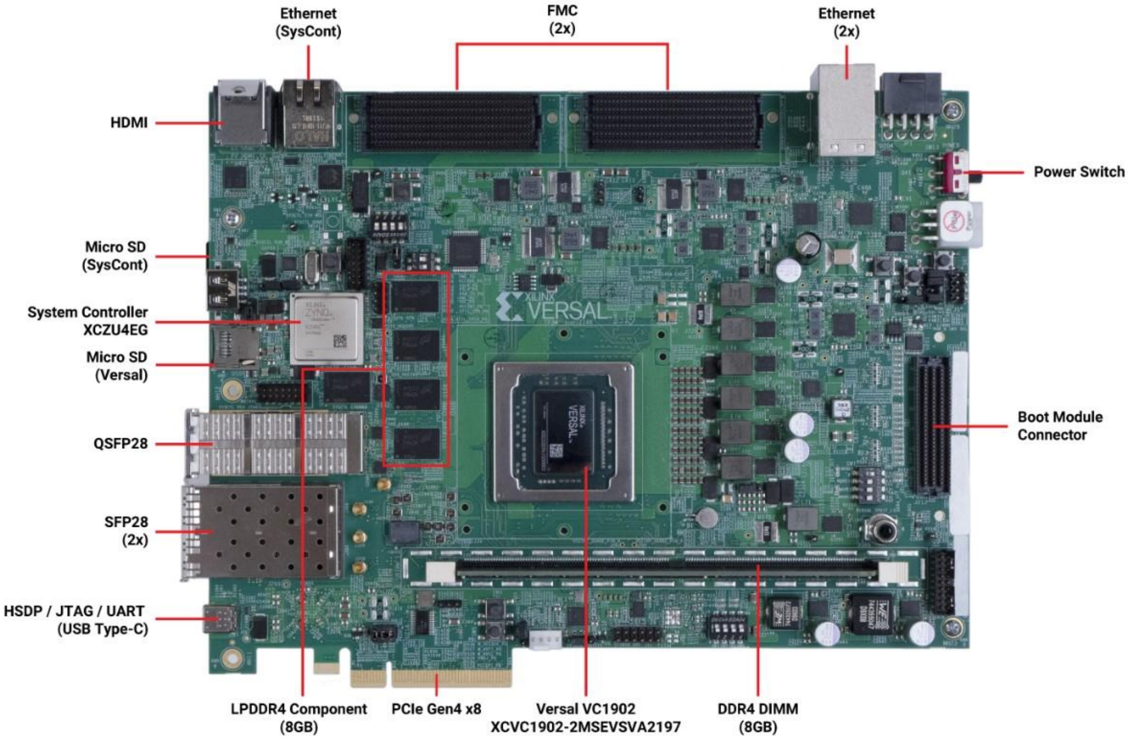


Figure 6. The Versal AI Core Series VCK190 Evaluation Kit

Several acceleration approaches (e.g., approximate computing on general-purpose CPUs, GPUs, custom AI/Machine Learning (ML)-oriented accelerators on Field-Programmable Gate Array (FPGA), Component Off The Shelf (COTS) AI/ML-oriented accelerators, etc.) will be considered, evaluated, and compared to identify the best ones for the different FRACTAL nodes also with respect to extra-functional properties (e.g., timing performance, power consumption, etc.).

The VERSAL platform provides support for integration of heterogenous compute elements with emulation and co-simulation in the Xilinx Vitis development environment. To use these features, the proper and shareable addition of FRACTAL elements like acceleration kernels in PL or the AI Engines requires packaging for this tooling. Adding such elements in the context of the tools further sets requirements for OS layer in an edge device. In the VERSAL ecosystem, services of the Xilinx runtime (XRT) are commonly used to set up and operate these accelerator components. These services restrict and define the form of the accelerators and need to be followed in the design. Also, this must be supported with insight into the application partitioning between the heterogenous compute elements in the VERSAL devices, i.e., the type of kernel and topology, e.g., for memory resources.

A customized, Linux-based OS layer is provided to FRACTAL use cases to deploy their applications. Embedded Linux solutions in Xilinx's environments are commonly build with PetaLinux Tools, which comes with a VERSAL platform-specific flavour of the Yocto Project. It also offers better integration with Xilinx's HLS tools that are required to set up and operate the accelerator components through XRT (i.e., Vitis).

On top of customizing the Linux OS layer for the FRACTAL project and providing a customized distribution of VERSAL native hardware resources, **WP3T32-10** will provide the capability to deploy different acceleration building blocks on top of available resources (AI Engines, PL kernels and DSPs), using the Xilinx framework deployed on FRACTAL OS Layer. This way, IKER will provide support to the use cases for the deployment of their accelerators on the VERSAL node using commercial Xilinx tools (i.e., Vitis AI).

The complete Xilinx Versal ACAP solution extends the platform notion beyond the device capabilities. To efficiently drive applications on these devices, the platform approach also extends into the design tools that support the common project integration across these heterogenous cores. The Versal ACAP hardware and software are targeted for programming and optimization by data scientists, software, and hardware developers by providing a host of tools, frameworks to start designs at any granularity. A full stack design spanning hardware attachment and exposing data paths and AIE accelerators to various of the heterogenous cores is also demonstrated in **WP3T34-03** as supported by PLC2.

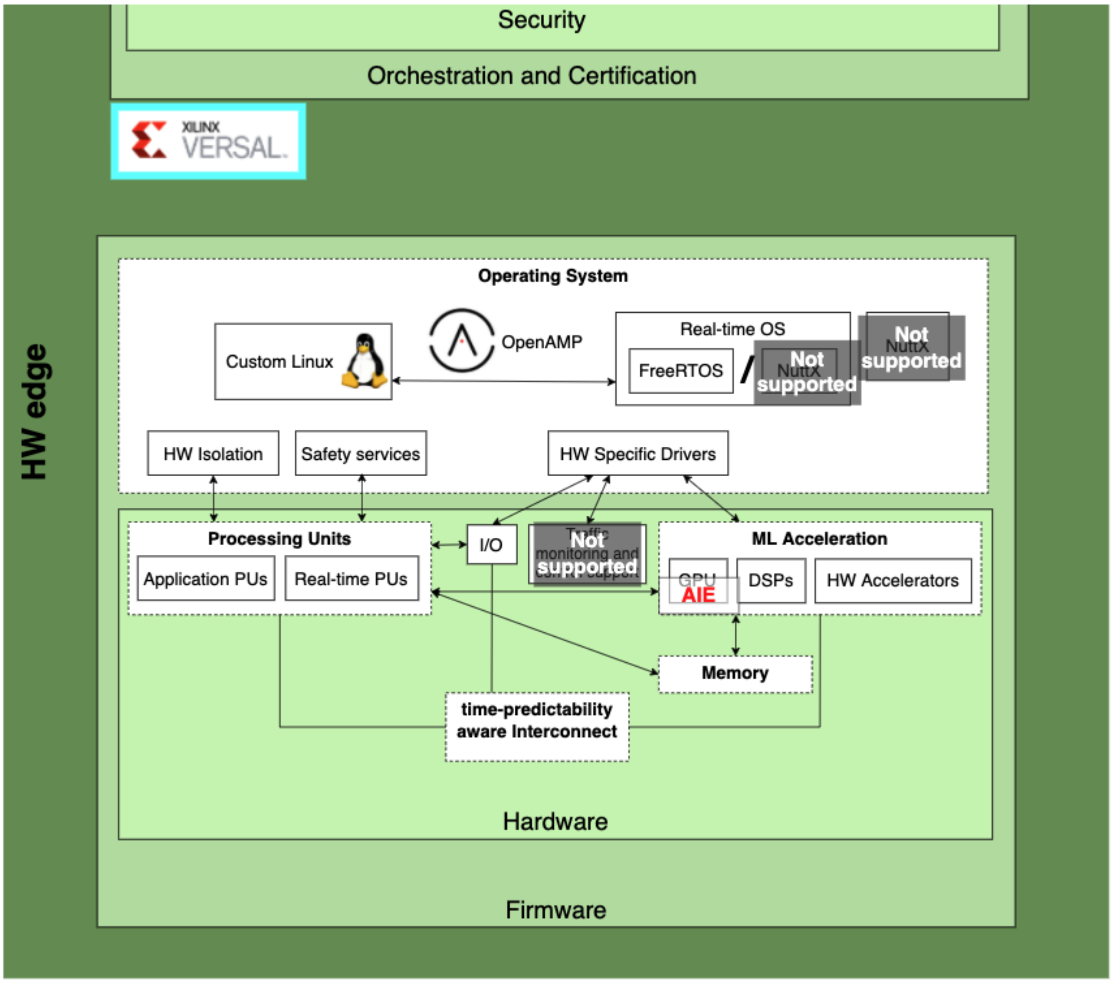


Figure 7 The VERSAL platform in the FRACTAL big picture

Figure 7 shows the VERSAL platform as described in the FRACTAL big picture. As the most capable hardware node of FRACTAL, the VERSAL node supports almost all the features identified in the big picture. The few relevant changes are the following

- **No plans for *NuttX***: *NuttX* is a low overhead operating system, and while there is nothing that would prevent from a system implemented on the VERSAL platform to support *NuttX*, there is also not practical need for such an addition.
- **Traffic monitoring and control support**: As a fully integrated system with high performance network on chips, VERSAL already features solutions for a variety of safety related aspects. Changes and adaptations by FRACTAL partners to provide additional improvements would be difficult to implement on the VERSAL platform as these features are not defined in the programmable logic but are inherent parts of the system.
- **AIE units instead of GPU**: VERSAL includes a large array of dedicated AI engines, that can be used to map applications that have traditionally

been mapped to GPUs. While the structure is not effectively a GPU, they share certain similarities.

3.2 The customizable node (PULP Platform)

FRACTAL applications that need a more mature technology and SW support and need higher performance would target the Xilinx VERSAL platform. For use cases that have lower performance requirements (closer to IoT applications), the RISC-V based open-source PULP (Parallel Ultra Low Power)[2] platform provides a second and flexible architecture that can be tailored to applications. As part of the PULP platform, there are several different single, multi-core and multi-cluster systems.

A suitable instantiation of PULP that can support the functionality of the FRACTAL framework will be used as a base platform for the customizable FRACTAL node that can be enhanced by additional cores and accelerators according to the requirements of specific use cases. The implementation of the customizable node will be on a suitable FPGA prototyping board, allowing prototypes to be rapidly deployed (Figure 8).

As a basic platform, FRACTAL will use the single core PULPissimo system (**component WP3T32-02**), but UC owners will be free to use any other implementation that fits their requirements. PULPissimo and other PULP based systems have already been implemented on a variety of FPGA based platforms, any of which can be used by the UCs.

[2] https://pulp-platform.org/

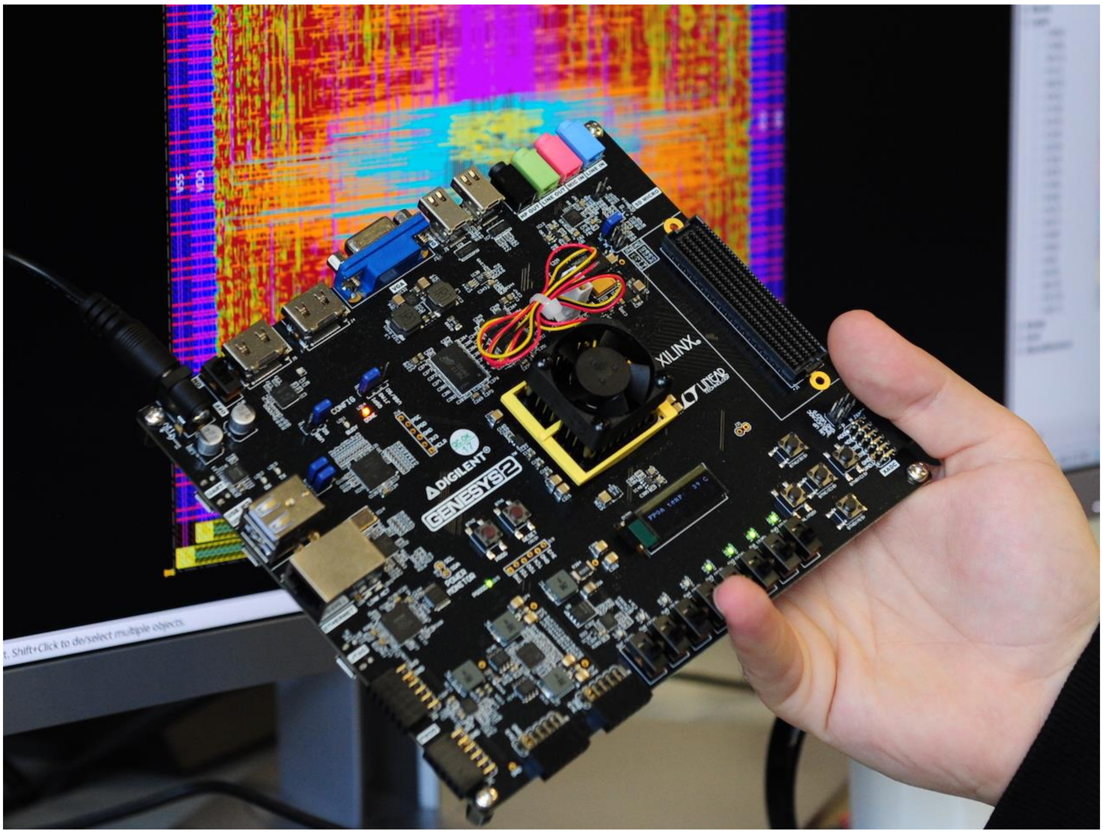


Figure 8. The Digilent Genesys 2 Xilinx FPGA board that has PULPissimo images ready to be used. The same board is also targeted by the CVA6/Ariane platform described under Section 3.3.2

The block diagram of PULPissimo is given in Figure 9. The heart of the system is a RISC-V core developed by ETH Zurich. Initially named RI5CY, this core has been adopted by the OpenHW Group[3] and has been rebranded as CV32E40P (CORE-V, 32bit, Embedded class, 4 pipeline stages with PULP extensions). The core supports the RV32IMCF extensions as well as DSP centric extensions that were developed by ETH Zurich[4]. It can use a regular RISC-V development environment for standard RISC-V instructions, and a modified compiler toolchain is needed to take advantage of the customized instructions.

[3] The OpenHW Group is a non-profit organization hosting open-source hardware projects. It is steered by its members. Among FRACTAL partners, ETH Zürich, BSC, Thales are OpenHW members.

[4] M. Gautschi et al., "Near-Threshold RISC-V core with DSP extensions for scalable IoT endpoint devices", In Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25 (10): 2700-2713, New York, NY: IEEE, 2017. DOI: 10.1109/TVLSI.2017.2654506

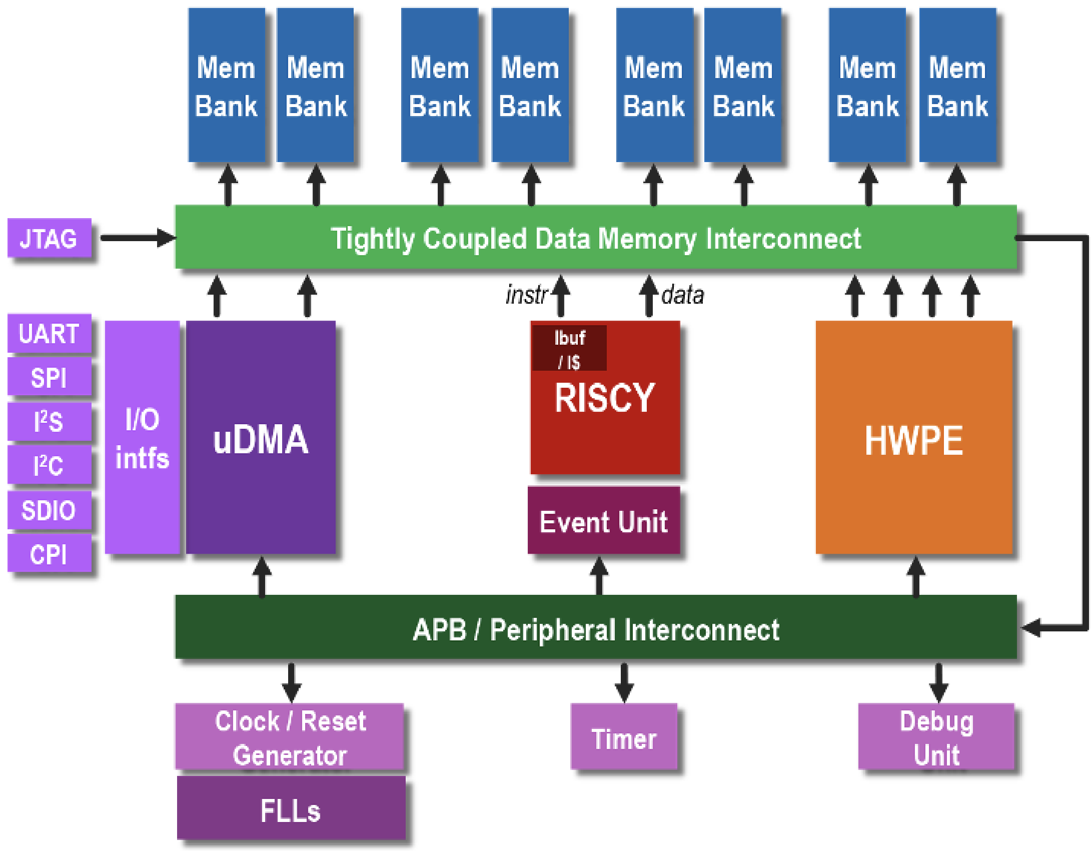


Figure 9. Block diagram of the PULPissimo system using a single 32bit RISC-V core (RI5CY/CV32E40P) and can easily be extended with accelerators, APB/AXI peripherals as well as instruction set extensions.

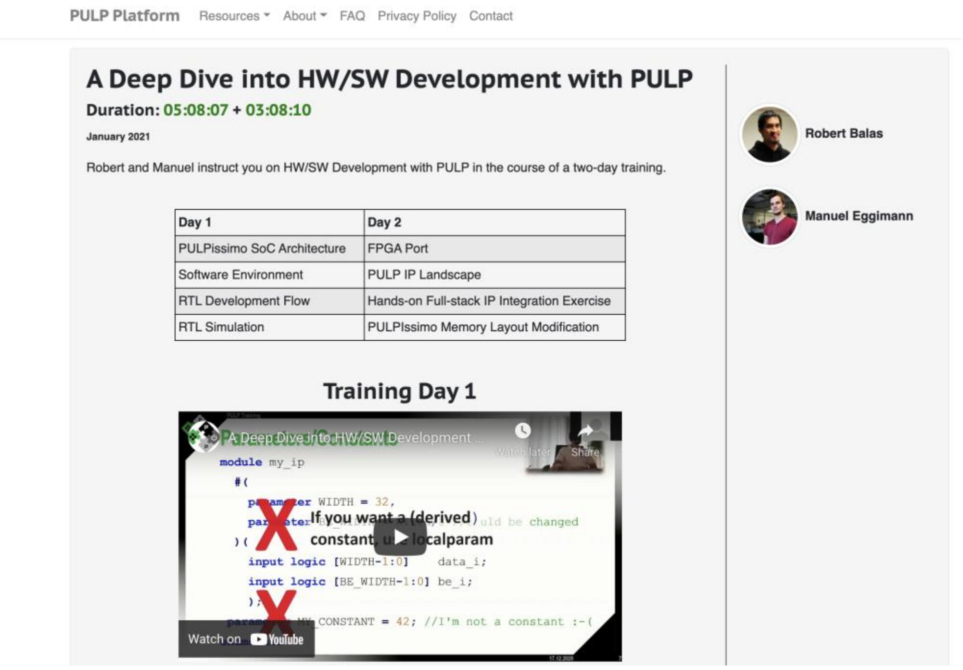
However, the processor core is only one part of the hardware node. There is a rich set of permissively licensed open-source peripheral components, connected over an Advanced eXtensible Interface (AXI) and a Direct Memory Access (DMA) unit that can transfer data independently between memory and peripherals.

Another important aspect of the architecture is the tightly coupled data memory interface between the processor core and the memory subsystem. The interconnect allows additional accelerators to be added (drawn in orange in Figure 9) that can access the memory the same way the processor does. This reduces the overhead of passing data between processor core and an accelerator and has proven to be extremely successful in multiple applications[5].

While this system was mainly designed to be used in an Application-Specific Integrated Circuit (ASIC) setting (and more than 15 ASICs have been manufactured and tested with a PULPissimo system), the important advantage for FRACTAL is that it has been mapped to a smaller scale Xilinx FPGA board Genesys 2 allowing rapid development (Figure 8). The system is also supported by a virtual platform that

[5] F. Conti et al. "An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics", In proc. IEEE Transactions on Circuits and Systems I, Regular Papers, 64 (9): 2481-2494, New York, NY: IEEE, 2017. DOI: 10.1109/TCSI.2017.269801

allows both HW and SW development to be performed using either the virtual platform, behavioral RTL simulation and FPGA emulation.



A Deep Dive into HW/SW Development with PULP

Duration: 05:08:07 + 03:08:10

January 2021

Robert and Manuel instruct you on HW/SW Development with PULP in the course of a two-day training.

Day 1	Day 2
PULPissimo SoC Architecture	FPGA Port
Software Environment	PULP IP Landscape
RTL Development Flow	Hands-on Full-stack IP Integration Exercise
RTL Simulation	PULPIssimo Memory Layout Modification

Robert Balas

Manuel Eggimann

Training Day 1

A Deep Dive into HW/SW Development ...
Watch later
Share
module my_ip
#(
parameter WIDTH = 32,
If you want a (derived) constant, use localparam
input logic [WIDTH-1:0] data_i;
input logic [BE_WIDTH-1:0] be_i;
MY_CONSTANT = 42; //I'm not a constant :-(
Watch on YouTube

Figure 10. An excerpt from the PULP training page (accessible under https://pulp-platform.org/pulp_training.html) that shows the free and accessible tutorials on the PULPissimo platform that will be used as the customizable node within FRACTAL. This particular tutorial covers more than 8 hours of training

Since the customizable node will be prototyped on an FPGA, the node will benefit from great flexibility to choose what FPGA or set of FPGAs to use to explore an arbitrarily large node with an arbitrary set of physical resources. The customizable node will be based on a highly scalable tile-based architecture in which cores and accelerators can be interconnected in a flexible manner. Furthermore, in FRACTAL this platform will include RISC-V processor cores that are made available using a permissible open-source license removing roadblocks towards commercial exploitation as well as avoiding premature lock-in and enabling unconstrained use by the different partners in the consortium, as well as by third parties willing to use FRACTAL technology. The customizable platform will allow many different approaches to implement accelerators in the project including re-using Xilinx and other existing IPs or the use of high-level synthesis.

When compared to the commercial node, the customizable node will offer complimentary opportunities. By selecting across the FPGAs in the market, one can prototype from tiny to very large nodes. In fact, multiple FPGAs can be connected to form very powerful nodes if needed. Hence, the range of trade-offs that can be explored is, by construction, much larger than that of the commercial node, which builds upon specific hardware resources. Additionally, high-performance features included in the commercial platform are not suitable for critical tasks which may limit the utilization of such platforms in the context of fail-operations autonomous systems such as the autonomous car. With the customizable platform, the aim is to extend safety properties beyond relatively simple single-core nodes by incorporating the appropriate hardware support to this node.

At the same time, while the PULP platform offers a good base to start, a FRACTAL node will still have to be developed building on technology already owned by the partners. This bears certain risks in the development of such an open and flexible platform due to the uncertainties on how the final node will look like. Moreover, time-to-market is also higher since the maturity of this platform is behind that of the commercial node.

FRACTAL involves several UCs, each with different computing requirements. In the first part of the project, the goal was to understand and determine the requirements from every UC that potentially could use the experimental FRACTAL node based on RISC-V cores, as was captured in D2.1. A research system like PULPissimo differs from conventional microcontroller systems, and FRACTAL partners have requested and were given introduction talks to the PULP architecture and open-source hardware. In addition, ETHZ made more than 19 hours of video tutorials available on the PULPissimo system (**component WP3T32-03**) under:

https://pulp-platform.org/pulp_training.html

With topics covering:

- PULPissimo SoC Architecture
- PULP IP Landscape
- Hands-on Full-stack IP Integration Exercise
- RTL Development Flow
- RTL Simulation
- FPGA Port
- PULPIssimo Memory Layout Modification
- PULP SDK / GCC Compilation Toolchain

Further resources include:

- The FreeRTOS port (https://github.com/pulp-platform/pulp-freertos)
- A tutorial for the PULP DSP library to help programmers write optimized code for DSP applications (similar to CMSIS for ARM) under https://pulp-platform.github.io/pulp-dsp/tutorial-index/
- And a Getting started guide from a software point of view under: https://pulp-platform.org/pulp_sw.html

The following Figure 11 shows the PULPissimo system as part of the FRACTAL big picture as discussed in the second Technical Workshop of FRACTAL. As the smallest hardware node with more modest capabilities, the most important aspect of the PULPissimo node is that the majority of the SW services that run on the node will be running over a FRACTAL server under *NuttX* (as opposed to systems that run a regular Linux distribution). As detailed in D3.4 the FRACTAL server and agent system will allow the PULPissimo which will be working as an edge node.

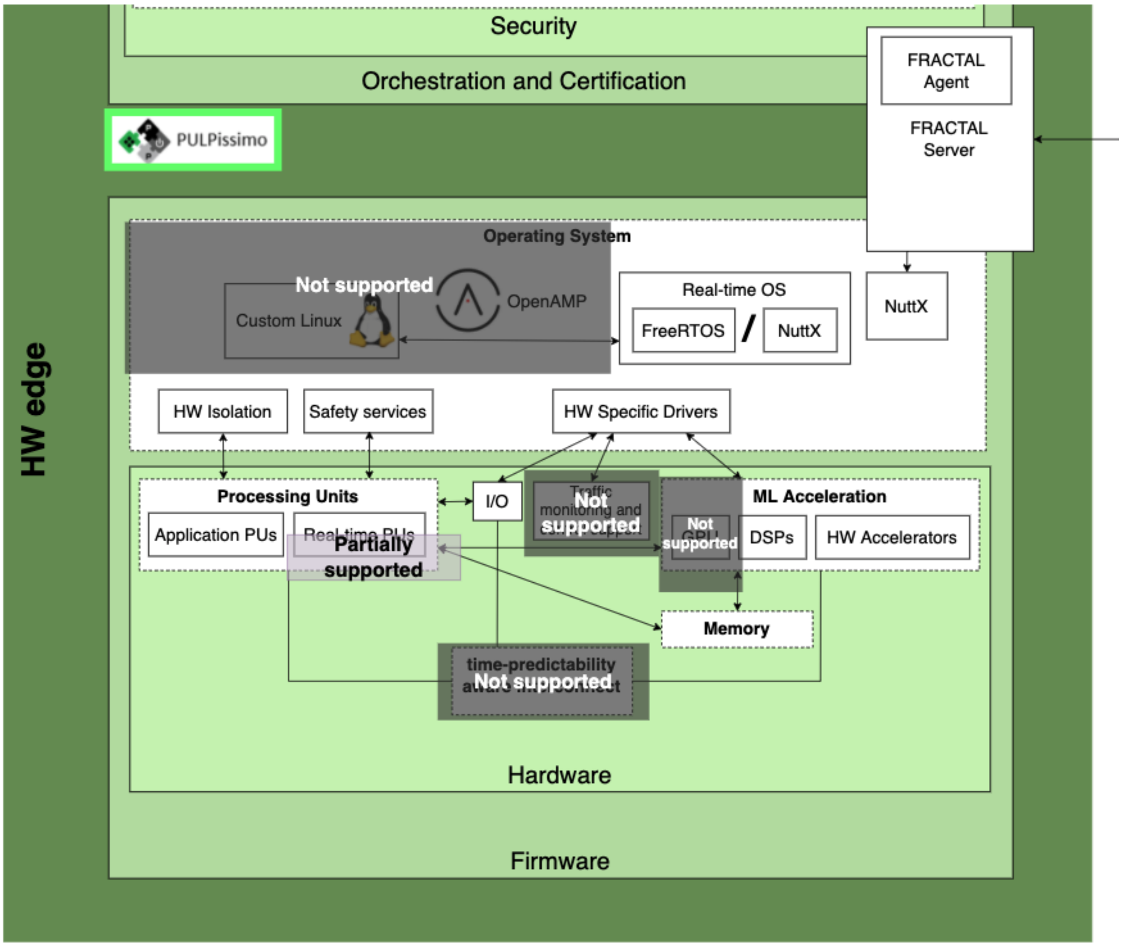


Figure 11 PULPissimo as part of the FRACTAL big picture

The following features of the big picture are not directly supported by the PULPissimo node

- **Linux and OpenAMP**: In theory it would be possible to add Linux support for the node, however, the application domain and the capability of the node is not at a level to support the applications that FRACTAL partners expect from a hardware node with Linux support Therefore the main operating system used for the PULPissimo system will remain the *FreeRTOS* (already available) and *NuttX* (in development).
- **Traffic Monitoring and control support**: As a simpler node PULPissimo does not have resources that face contention, and traffic monitoring and control support tasks are not that relevant for its operation.
- **Time predictability aware interconnect**: Similarly, within the node, owing to its simpler structure there are not many opportunities to make use of a time-predictability aware interconnect, although it is still expected that the

node make use of the features that will be introduced in WP4 on time predictable networking.

- **GPUs**: as a simple node, PULPissimo does not have support for GPUs, however, the processing core has DSP extensions, and the architecture has been specifically designed to support Hardware Accelerators
- **Real-time PUs**: PULPissimo based systems have been successfully used for real-time control applications. While the cores have not (yet) been certified, their use has been demonstrated.

While ASICs implementations were not planned directly as part of the FRACTAL efforts due to its longer development times, the work supported by FRACTAL ended up in multiple ASIC implementations by ETH Zürich as seen in Figure 12. This proves that the codebase for the PULPissimo system supported by FRACTAL has the quality to be used in modern ICs

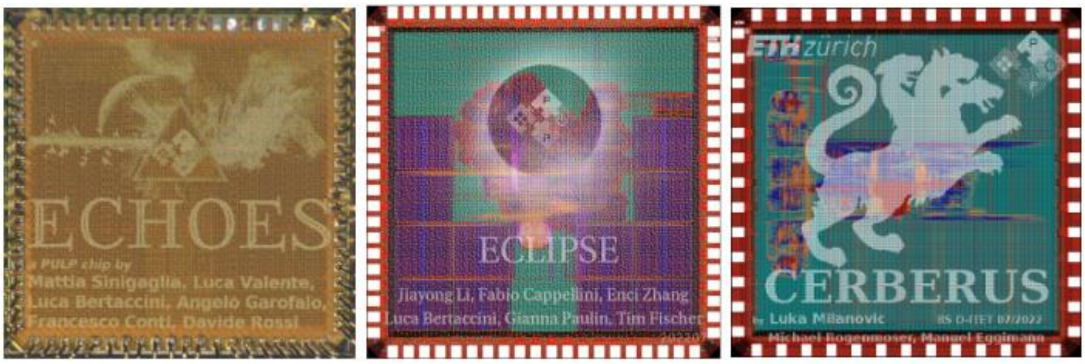

ECHOES
a PULP chip by
Mattia Sinigaglia, Luca Valente,
Luca Bertaccini, Angelo Garofalo,
Francesco Conti, Davide Rossi
ECLIPSE
Jiayong Li, Fabio Cappellini, Enci Zhang
Luca Bertaccini, Gianna Paulin, Tim Fischer
ETH zürich
CERBERUS
Luka Milanovic
IIS D-ITET 07/2022
Michael Rogenmoser, Manuel Eggimann

Figure 12 A selection of ASICs designed by ETH Zurich in TSMC 65nm using the PULPissimo code base supported by WP3 activities. From left to right: Echoes (2021) complete PULPissimo system with audio processing capabilities, Eclipse (2022) that includes new FP formats added to the PULPissimo system for better supporting ML applications and Cerberus (2022) which includes a PULPissimo system with triple cores operating in lock-step mode for safety.

3.3 Other customizable nodes

During this study, several use cases stated the need for more traditional RISC-V based systems (capable of running single-core or SMP Linux) which resulted in some additional hardware nodes being added. These will be covered in this section.

3.3.1 NOEL-V

The NOEL-V based RISC-V platform (https://gitlab.com/selene-riscv-platform) is a multicore SoC based on the open-source platform from the H2020 SELENE project[6]. The schematic of the computing part of the SoC is shown in Figure 13. As shown, the SoC includes 4 NOEL-V 64-bit cores from Cobham Gaisler implementing the RISC-V ISA, an AMBA AHB bus connecting the cores with the I/O, and an AXI crossbar based on the AXI PULP interconnect (https://github.com/pulp-platform/axi) to allow cores

[6] C. Hernàndez et al., "SELENE: Self-Monitored Dependable Platform for High-Performance Safety-Critical Systems," 2020 23rd Euromicro Conference on Digital System Design (DSD), 2020, pp. 370-377, doi: 10.1109/DSD51259.2020.00066.

and accelerators share a DDR3 controller. An AMBA APB peripheral bus is also included for I/O device controllers like DUART, I2C, etc.

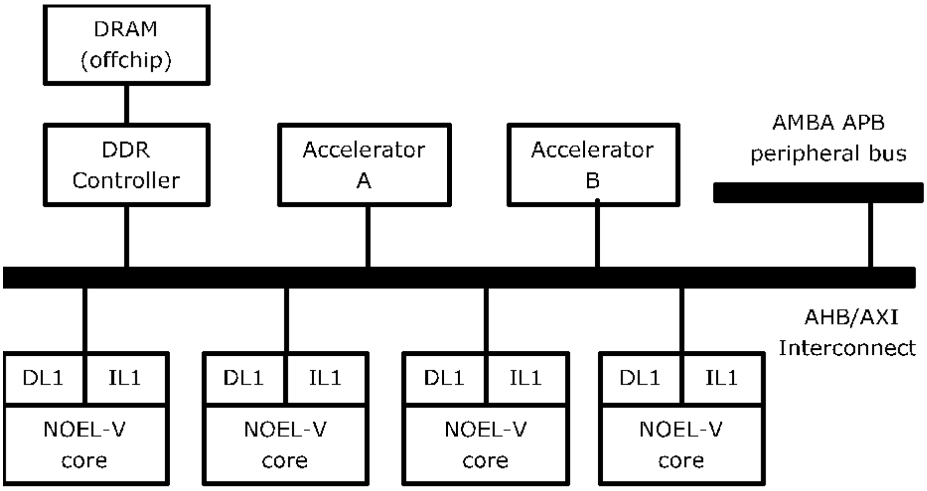


Figure 13: Schematic of the computing part of the NOEL-V based SoC

Gaisler's NOEL-V cores provide moderate performance figures despite being in-order cores building on their dual-issue 7-stages pipelined architecture with branch prediction, return address stack, write buffers, and data (DL1) and instruction (IL1) 16KB set-associative cache memories. The cores also implement integer and floating-point pipelines.

The NOEL-V based SoC can be easily extended with different accelerators and other components. Therefore, it eases the integration of features such as the AI-based accelerator from UPV, as well as BSC's statistics unit, both intended to be connected to the interconnect. For AI acceleration we have integrated the HLSinf (https://github.com/PEAK-UPV/HLSinf) open-source platform developed in High-Level Synthesis (HLS) by UPV. HLSinf is an FPGA-based accelerator specifically aimed to accelerate convolutional neural networks. The support for other type of networks as the one required by UC7 is also planned in the context of WP8.

HLSinf is designed around the channel slicing concept. Indeed, the main operation performed by the accelerator is the 2D convolution operation. This operation takes as an input a set of input channels (feature maps from previous layers) and produces a set of output channels (output feature maps). The accelerator handles in parallel a set of input channels (CPI, channels per input) and produces in parallel a set of output channels (CPO, channels per output). Both CPI and CPO parameters can be instantiated at design time, enabling the implementation of accelerators of different sizes and performances. The accelerator is designed in order to process CPI input pixels and produce CPO output pixels per clock cycle.

HLSinf is efficiently supported by the LEDEL library (**Component WP3T35-02**). This means that heavy computations can be offloaded from the NOEL-V CPU cores to this accelerator. To that end, the LEDEL library exploits the existing support for FPGA acceleration. Using the FPGA computation target provided by the LEDEL we can offload computations to an FPGA device which in the case of the NOEL-V system used in FRACTAL is the same FPGA device that is used for the SoC prototype. The LEDEL library and the HLSinf accelerator use different data formats to store tensors in memory. Thus, a new layer that adapts the data format of the LEDEL library to the HLSinf has been created. Additionally, to avoid unnecessary data movements and communications between cores and CPU all operations performed in the accelerator are grouped into a single layer that has the ability to perform in an efficient manner the concatenation of several operations (that are performed in different layers in the LEDEL when deployed in CPU targets).

The most compute-demanding operation in image-based NNs is the 2D convolution. HLSinf convolution module can be customized in the type of convolution to perform. Currently, direct convolution, Winograd's algorithm, and DepthWise Separable convolutions are supported. These different convolution implementations can be exploited to achieve diverse redundant execution for safety purposes. For that a redundant acceleration scheme based on HLSinf has been developed (**Component WP3T32-06**). The developed scheme allows implementing a parameterized N-modular redundant scheme where accelerators are configured and interconnected to memory with the AXI-pulp interconnect. To that end an AXI-lite crossbar and width converters were included the NOEL-V SoC interconnect (**Component WP3T31-02**) using the appropriate VHDL to SystemVerilog wrappers. On completion, accelerators do not trigger interrupts but require the CPU to periodically check the status of the computations. Additionally, this CPU monitor checks whether computations were correct or some errors were found. Both the hardware and the software monitors are available to be integrated with use-cases.

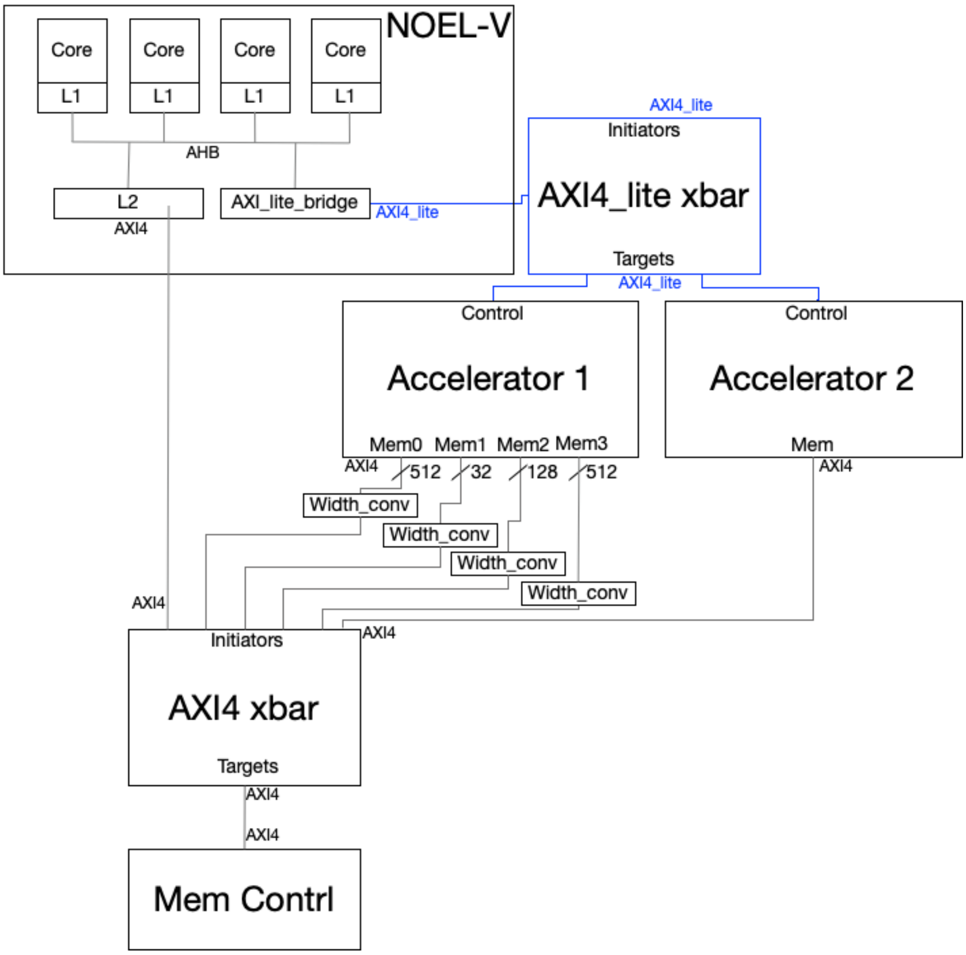


Figure 14 Dual redundancy acceleration scheme instance interconnected in the NOEL-V Platform

The NOEL-V SoC supports memory management units, and implements Translation Lookaside Buffers (TLBs), both for data and instructions, locally in each core. The SoC also provides support for cache coherence. Those features allow booting SMP Linux and RTEMS operating systems among others and allow sharing data across cores. For booting Linux, both buildroot and ISAR-based workflows have been adapted to the NOEL-V.

The SELENE SoC has been synthesized in a Xilinx Virtex UltraScale VCU118 FPGA and the original NOEL-V SoC is also available for the KCU115. While its primary target is the space domain, it has been retargeted to enable its use for avionics, railway and automotive applications.

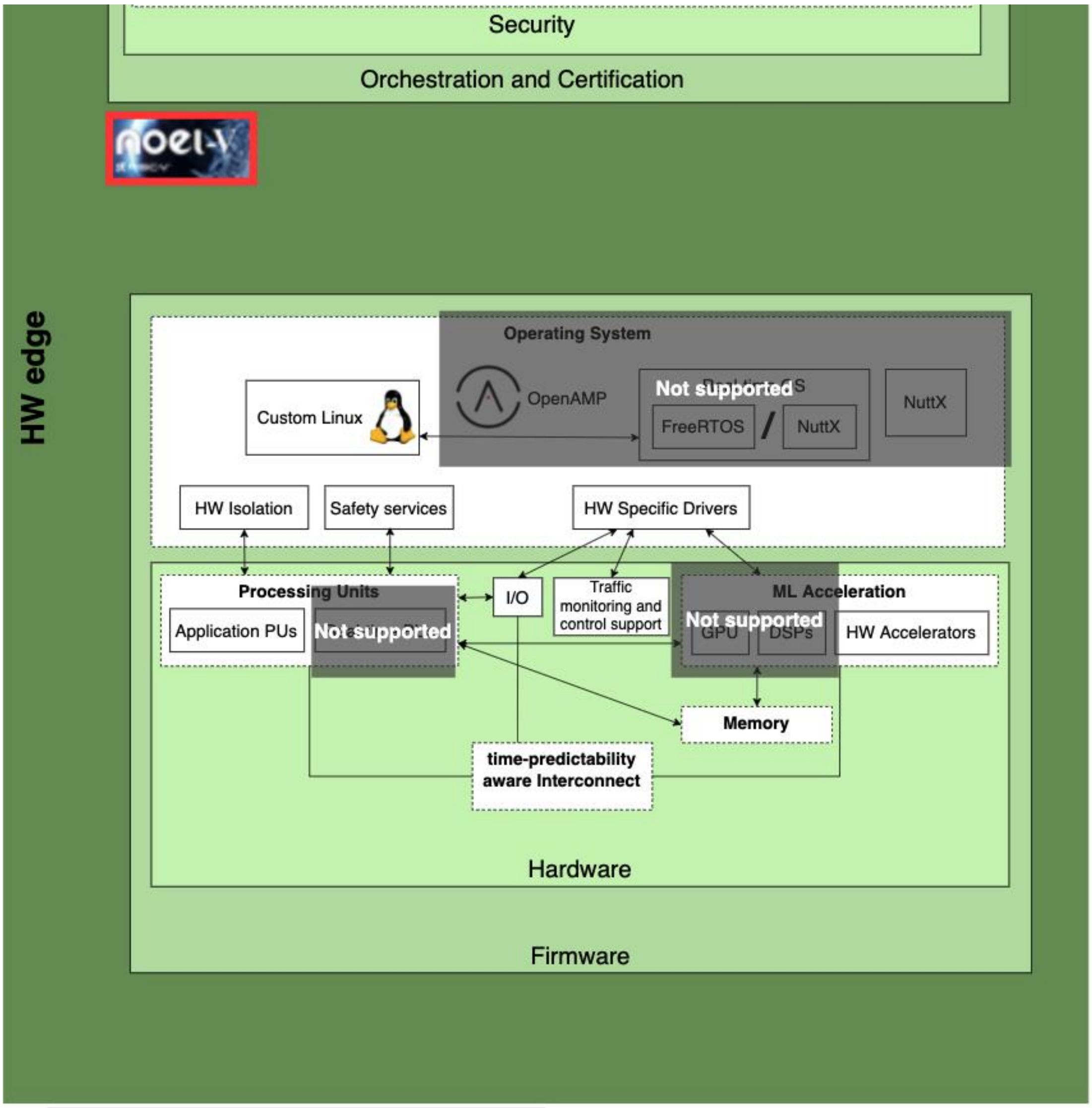


Figure 15 Noel-V as part of the FRACTAL big picture

In Figure 15, Noel-V can be seen as part of the FRACTAL big picture. As a testbed for safety running multi-core Linux, the Noel-V hardware concentrates mostly on features that allow multiple cores function in a predictable and safe manner. The main features that are not supported are:

- **OpenAMP, FreeRTOS, NuttX**: As the features that will be developed are all based around the Linux kernel, there are no plans to support other operating systems.
- **GPUs, DSPs**: While the system supports hardware acceleration, computing performance is not the reason why this system has been developed.
- **Real-time PUs**: The Noel-V system uses standard RISC-V cores and does not feature additional processing units with hardware support for real-time features.

3.3.2 Ariane/CVA6

As noted in Section 3.2, the open-source PULP Platform provides many different RISC-V based solutions, but the 32bit PULPissimo system was chosen as the default configuration for the customizable node due to its simplicity and wide range of options to adapt it, including adding peripherals, adding hardware accelerators as well as adding instruction set extensions. At the same time, the PULPissimo system is too simple to run a modern Linux operating system.

To support partners who wish to use a RISC-V system with Linux support, the PULP-based Ariane system has also been considered to be used as part of the FRACTAL project. Similar to the RI5CY core used in PULPissimo, Ariane was the code name of the 64bit Linux capable core (RV64GC) developed at ETH Zurich[7]. This core has been taken over by the OpenHW Group and has been renamed as CV64A6 (CORE-V, 64-bit, Application class, 6 stage pipeline) (**component WP3T32-02b**).

In co-operation with OpenHW Group, Thales has developed a 32-bit version of Ariane codenamed CV32A6 (**component WP3T33-03**), also supporting Linux, and offering a reduced footprint (compared to CV64A6) and upcoming PPA optimizations for FPGA targets. CV64A6 and CV32A6 share a common RTL base and are therefore jointly referred to as CVA6.

CVA6 compares favorably to existing processors:

- It is an application core (able to run rich OSes like Linux)
- It exists in 32 and 64 bits.
- It is technology-independent and targets integrations both in ASICs and as a soft core in FPGAs.
- It is written in SystemVerilog, a widespread hardware description language.

But as a unique selling point, it is available under a permissive open-source license (Solderpad), which allows its integration also in proprietary designs. The closest comparable open-source cores are NOEL-V, which is distributed under the GPL (not permissive) license and CHIPS Alliance's Rocket, which is written in CHISEL, a powerful yet not standardized language, a hurdle for the adoption in several industrial domains. If we look at proprietary cores, ARM's closest core is the Cortex A5, while on the FPGA side, you can find soft technology-dependent soft cores like Xilinx's MicroBlaze.

[7] Florian Zaruba, Luca Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-ready 1.7GHz 64bit RISC-V Core in 22nm FDSOI Technology", In Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol: 27, Issue: 11, Page(s): 2629 – 2640, Nov. 2019, DOI: 10.1109/TVLSI.2019.2926114

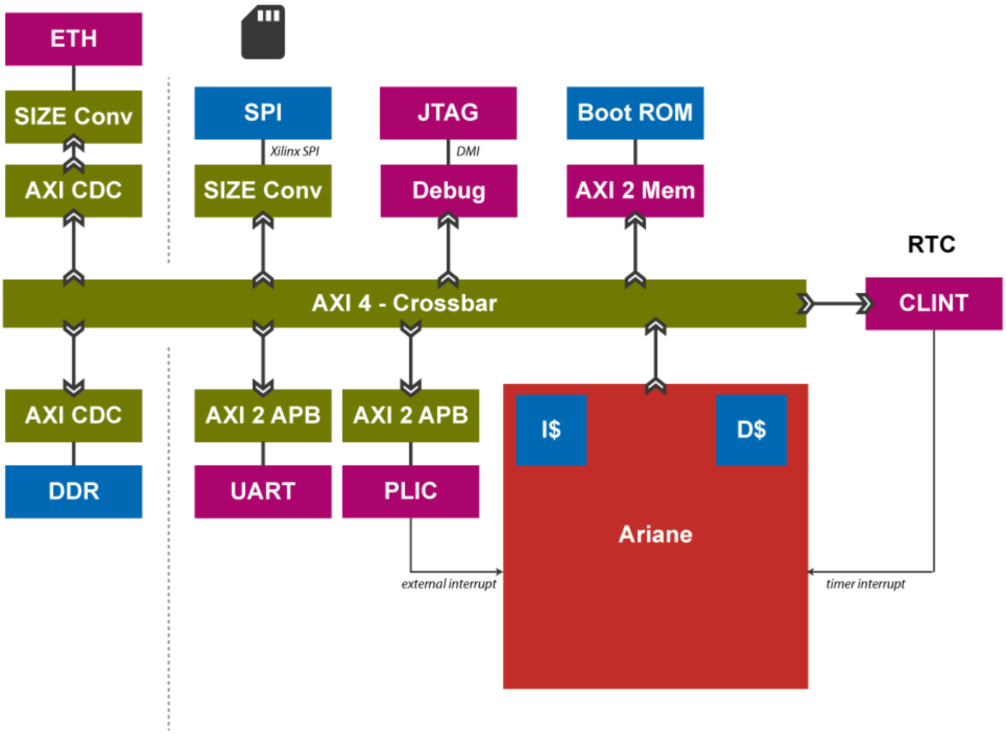


Figure 16: Schematic of the Ariane/CVA core mapped to the Genesys 2 board.

Ariane/CVA6 comes with a mapping to the same Xilinx-based Genesys 2 board that also supports PULPissimo (Figure 16). Within FRACTAL project, the Ariane/CAV6 is also ported to Xilinx UltraScale+ ZCU102 which offers a larger PL component that allows larger HW accelerators to be added. The FPGA SoC currently contains the following peripherals:

- DDR3 memory controller
- SPI controller to connect to an SD Card
- Ethernet controller
- JTAG port with support for OpenOCD
- Bootrom containing zero stage bootloader and device tree.

And more additions are expected to come through independent contributions by both the OpenHW Group and ETH Zurich developments which will be made available to FRACTAL partners as well.

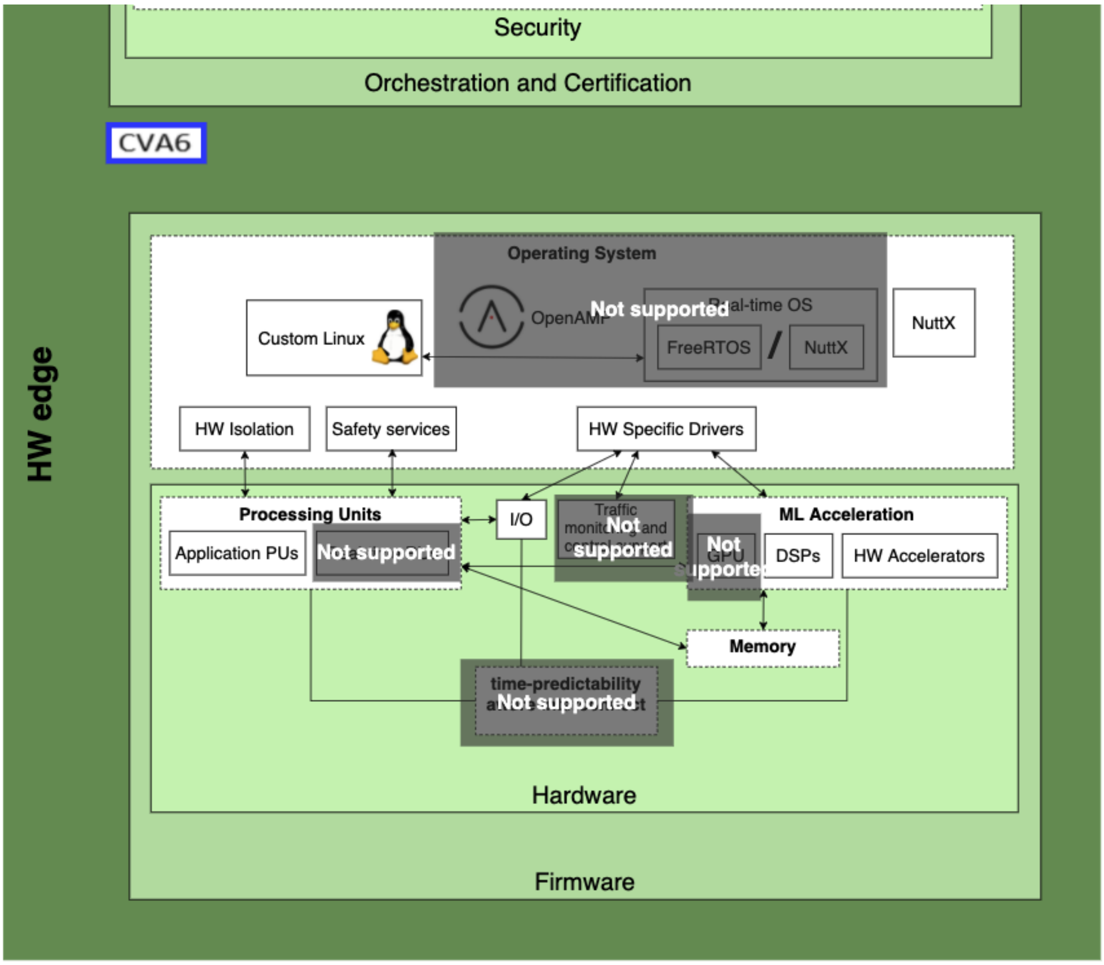


Figure 17 CVA6 as part of the FRACTAL big picture

In Figure 17, the CVA6 can be seen as part of the FRACTAL big picture. The following FRACTAL features form this big picture are not implemented on CVA6:

- **OpenAMP and Real-time OS**: There is no reason why FreeRTOS could not be ported to CVA6, however, as the core has been designed to run Linux, and for smaller applications there is already PULPissimo based solutions, there was no pressing need to address this support.
- **Traffic Monitoring and control support**: Work done on FRACTAL on traffic monitoring has been concentrated on the NOEL-V platform. There are no plans to replicate this on a CVA6 based system for the targeted use cases.
- **Time predictability aware interconnect**: While as a single core system there is no express need to make use of a time-predictability aware interconnect, it is still expected that the node makes use of the features that will be introduced in WP4 for communications between the platform and the rest of the system.
- **GPUs**: There is no GPU support, any such additions are normally addressed by HW accelerators that take the form of a cluster based many-core architecture.

- **Real-time PUs**: There are no plans to update the CVA6 core for a real-time PU.

Similar to PULPissimo (Section 3.1), the goal of the developments in FRACTAL did not see any ASIC implementations, but the codebase was used in the implementation of several ASICs by ETHZ. Neo (seen in Figure 18) is one such implementation which has been sent to fabrication in a mature TSMC65nm technology and contains the support system for an Ariane/CVA6 based Linux capable system within a 6.4 mm^2 and a power envelope of around 100mW it is expected to be an interesting solution for mist nodes. Of course such a small design has more limited processing capability than traditional desktop computers running Linux.

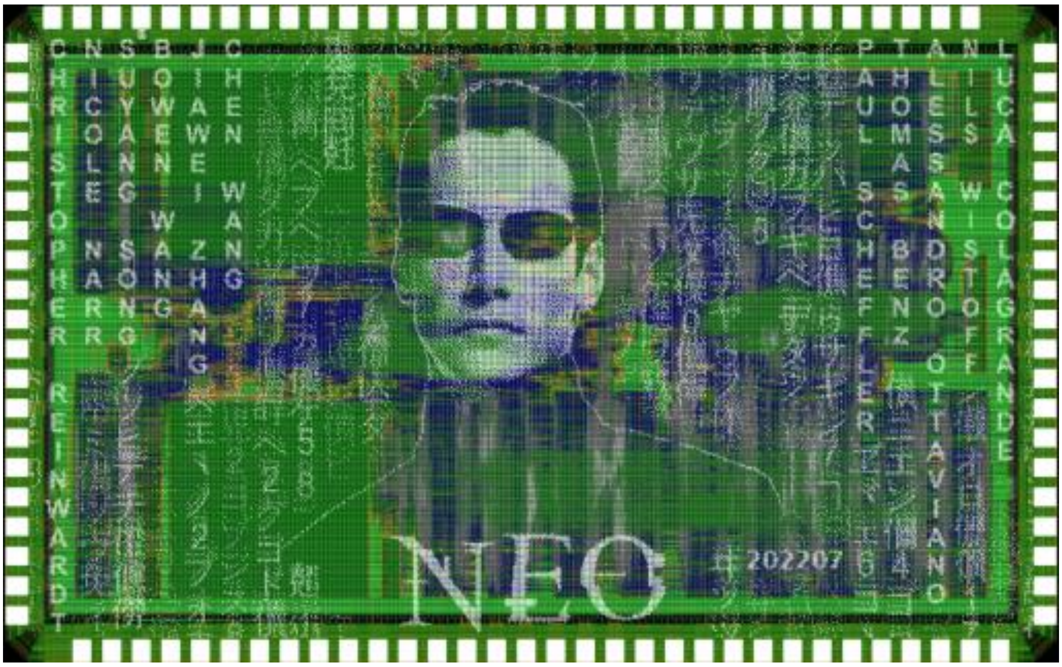


Figure 18 Neo, an Ariane/CVA6 implementation in TSMC65nm using the codebase supported by FRACTAL developments.

4 Supporting FRACTAL developments on safety, security, low-power and cognitive awareness

4.1 Supporting FRACTAL developments on low-power

The PULP Platform is the result of the research work of FRACTAL partner ETHZ on low power architectures and as such it has been designed around principles to increase energy efficiency and reduce unnecessary power consumption. This is an ideal starting point for architectural modifications envisioned as part of WP4 to reduce power consumption of FRACTAL nodes.

The VERSAL platform relies on a centralized Platform Management Controller (PMC) for device management control functions. Power efficient designs require usage of complex system architectures with several hardware options to reduce power consumption and usage of a specialized CPU to handle all power management requests coming from multiple masters to power on, power off resources, and handle power state transitions. In addition, there are other resources like clock, reset, and pins that need to be similarly managed.

The platform management in VERSAL is available to support a flexible management control through the PMC. This platform management handles several scenarios and allows the user to execute power management decisions through its framework (equivalent to what it is done in Linux, which provides basic power management capabilities like CPU frequency scaling).

4.2 Supporting FRACTAL developments on safety

Component WP3T31-01

State of the art: PMUs are often regarded as non-functional FUBs since, in general, end user applications do not build among them. Thus, their design, verification, and validation has received little attention from industry, which explains why products are shipped with erroneous event counters, even if intended for some critical real-time embedded markets[8], or with incomplete sets of event counters unable to monitor some events that can only be approximated to some extent with other events[9,10]. In fact, PMUs are often set up with a limited set of event counters, and extended with potentially exotic event counters as a means to ease design debug during product validation whenever erroneous behaviour is observed. In other words, extended event counters become the hardware counterpart of software `printf` statements. Thus, while most of those events in the SoC become visible to end users,

[8] J. Barrera et al., "On the reliability of hardware event monitors in MPSoCs for critical domains," in 35th ACM/SIGAPP Symposium On Applied Computing (SAC), 2020.

[9] J. Jalle et al., "Bounding resource-contention interference in the next-generation multipurpose processor (NGMP)," in 8th European Congress on Embedded Real Time Software and Systems (ERTS2), 2016

[10] X. Palomo et al., "Tracing Hardware Monitors in the GR712RC Multicore Platform: Challenges and Lessons Learnt from a Space Case Study," in 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020), ser. Leibniz International Proceedings in Informatics (LIPIcs), 2020

they have not been conceived for any other purpose different to hardware debugging of errors already fixed, and are poorly – if at all – validated[11].

However, the use of multicores in critical real-time domains brings a need for mastering inter-core interference as a way to determine execution time bounds for real-time applications. Successful solutions on Commercial-Off-The-Shelf (COTS) SoCs resort to PMU-related information to estimate how much interference applications can cause and experience and the number of experiences of that type building upon PMUs is abundant[12].

Unfortunately, lowly validated PMUs and critical real-time systems building upon the information provided by PMUs for the most critical applications, are hard to reconcile and impose extensive software-based PMU validation for their reliable use[13].

In this work, we propose extending PMUs with appropriate features that allow validating them during SoC validation and verification phases and, additionally validating the design and integration of other FUBs monitored by PMUs.

Description: A multicore interference-aware PMU will provide safety support for verification, validation and deploying safety measures during operation. The PMU is an advanced statistical unit including controllability and observability channels that will be used to deal with timing interference concerns in safety-critical real-time applications on top of the PULP-related SoCs. Currently, this effort has been integrated into the NOEL-V platform, however the PMU is Advanced Microcontroller Bus Architecture compliant (AMBA-compliant), all registers can be accessed to be read or written through Advanced eXtensible Interface (AXI) or Advanced High-performance Bus (AHB) interfaces. The PMU is fully customizable and can be tailored to a wide variety of multicore architectures, including those based on RISC-V architecture. The PMU's control and parametrization are done by software with an appropriate library. Some work performed in the PMU is as follows:

[11] E. Mezzetti et al., "High-integrity performance monitoring units in automotive chips for reliable timing v&v," IEEE Micro, vol. 38, no. 1, pp. 56–65, January 2018

[12] E. Mezzetti et al., "Aurix tc277 multicore contention model integration for automotive applications," in 2019 Design, Automation Test in Europe Conference Exhibition (DATE), March 2019, pp. 1202–1203.
- E. Diaz et al., "Modelling multicore contention on the aurixtm tc27x," in Proceedings of the 55th Annual Design Automation Conference, ser. DAC '18, 2018.
- G. Fernandez et al., "Assessing time predictability features of arm big. little multicores," in 2018, 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Sep. 2018, pp. 258–261.
- J. Jalle et al., "Bounding resource-contention interference in the next-generation multipurpose processor (NGMP)," in 8th European Congress on Embedded Real Time Software and Systems (ERTS2), 2016.
- J. Nowotsch and M. Paulitsch, "Leveraging multi-core computing architectures in avionics," in EDCC, 2012.
- J. Nowotsch et al., "Multi-core interference-sensitive WCET analysis leveraging runtime resource capacity enforcement," in 2014 26th Euromicro Conference on Real-Time Systems, July 2014, pp. 109–118.
- S. Girbal et al., "METrICS: a measurement environment for multi-core time critical systems," in 9th European Congress on Embedded Real Time Software and Systems (ERTS2), 2018.

[13] F. Cazorla et al., "Understanding interference in critical multicore systems," in 24th Data Systems In Aerospace Conference (DASIA), 2019

- We have found out how multicore interference manifests in the interfaces between the cores and the shared L2 cache module, and between the L2 cache and the DRAM controller. Based on that, the PMU has been shown to capture properly interference in the core-to-L2 interface, but it was not capturing it in the L2-to-DRAM interface. Hence, the PMU has been adapted to capture properly such interference as part of its integration. Most of such support has already been inherited from the original platform, but needed to be tailored to capture properly interference for the specific masters and slaves present in the specific SoC used in FRACTAL.
- The implementation of additional statistics has finalized. Those statistics have been adapted to the particular characteristics of the final node as part of the integration process. Those statistics include number of transactions and data transferred per master and slave, broken down across data sent and received. The detailed statistics per master are shown below in
- Table 2.
- Validation of the integration is complete. However, some additional testing is being conducted during the preparation of the integration with UC7.

Table 2 Additional events measured per master in the PMU

Event	**Description**
`DataSent`	Number of bytes sent by the master to any slave.
`DataReceived`	Number of bytes received by the master from any slave.
`ReqSent`	Number of requests sent by the master to any slave.
`ReqReceived`	Number of requests received by the master from any slave.
`ReqSent4B`	Number of 4-byte requests sent by the master to any slave. Typical size of word transactions.
`ReqReceived4B`	Number of 4-byte requests received by the master from any slave. Typical size of word transactions.
`ReqSent16B`	Number of 16-byte requests sent by the master to any slave. Typical size of cache line transactions.
`ReqReceived16B`	Number of 16-byte requests received by the master from any slave. Typical size of cache line transactions.

Component WP3T31-03

Safety and security unified support in the NOEL-V platforms is provided using an interface similar to SiFive’s *WorldGuard* specification. *WorldGuard* restricts the access to shared resources to the masters that belong to a specific world preventing non-CPU masters, to access specific memory regions or resources extending the properties of the memory management unit to the I/O regions. To that end, every SoC request is labeled with a specific ID that defines the world that request belongs to. Only requests with a specific ID can be granted access to a specific resource.

Although, *WorldGuard* was conceived for security purposes our goal in FRACTAL is to exploit its properties for safety as well. Every SoC requests is labelled with an ID to allow performing arbitration decisions and quota enforcement mechanisms in shared resources. The scheme is already implemented in the NOEL-V platform and the PMU is able to use this information to assign the contention to specific CPU cores.

Additionally, to improve the robustness of Staggered Redundant Execution (SRE) in the presence of Common-Cause Faults (CCFs) affecting the CPU's register file a novel randomization mechanism has been include in the NOEL-V cores. In particular, the proposed mechanism performs a Register File Randomization (RFR) that increases the robustness of homogeneous multicores by improving the protection against CCFs affecting the register-file. The goal of this technique is to dynamically modify the physical register file access pattern by using a randomization mechanism. We distinguish the architectural (or logical) from the physical register file entry. This randomization mechanism allows us to have different register file mappings (i.e modifying the binding from the logical register to the particular physical register entry) so that systematic faults affecting the register file structures can be detected with SRE. In particular, RFR combines the index bits required to access the register file with the core ID and a random value to generate a randomized index to access register file contents.

RFR modifies the access to the register file by keeping the remaining parts of the core unaltered. These modifications are needed in each of the SoC cores. The architectural view of the RFR mechanism is depicted in Figure 19. As shown in the diagram, RFR only requires the introduction of a hashing circuit at the register file input to randomize the indexes to access the register file.

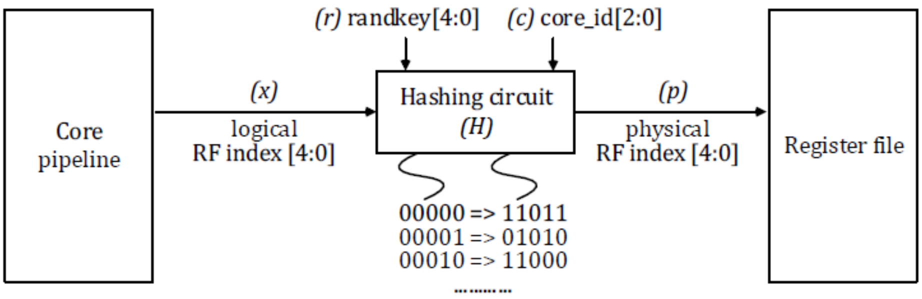


Figure 19 RFR scheme from the FRACTAL component WP3T31-03

RFR improves the robustness in two main aspects. First, it allows to correct or detect common-cause faults originated in the register file by modifying the effective physical location of registers in the different core replicas. Notice that common cause faults are more likely to affect the same physical register since core replicas are implemented using the same SRAM macrocells and usually have the same layout structure. Additionally, in the absence of RFR, the same register entries are stressed with very similar activity patterns. Specially, when they are devoted to implement SRE. Second, RFR also mitigates the effect of wearout in the layout due to stress by uniformizing the utilization of register file cells. Generally, Application Binary

Interface (ABI) and programs impose specific usages to registers that make them to be unevenly utilized. Additionally, in the context of critical systems, workloads are usually repetitive, which exacerbates more this uneven distribution effect.

Additional safety features are actively being investigated on the Ariane/CVA6 core including:

- Ability to invalidate/flush the cache and Translation Lookaside Buffer (TLB) to return to a known state (predictability)
- Ability to define non-cacheable address chunks for peripherals (predictability)
- Ability to disable caches (higher predictability at the expense of performance)
- Ability to disable predictions, such as branch predictions (higher predictability at the expense of performance)
- Availability of local memory, such as scratchpad or locked lines/set in the L1 cache (strong real-time and higher predictability for critical processes)
- Addition of performance counters

Component WP3T32-09

This component is developed by UNIMORE and UNIVAQ, within the framework of Task 3.1 and Task 3.2, and it is mainly focused on real-time safety-related aspects. These activities are related to the mobility of the UC6 (intelligent totem), that represents a firm real-time UC.

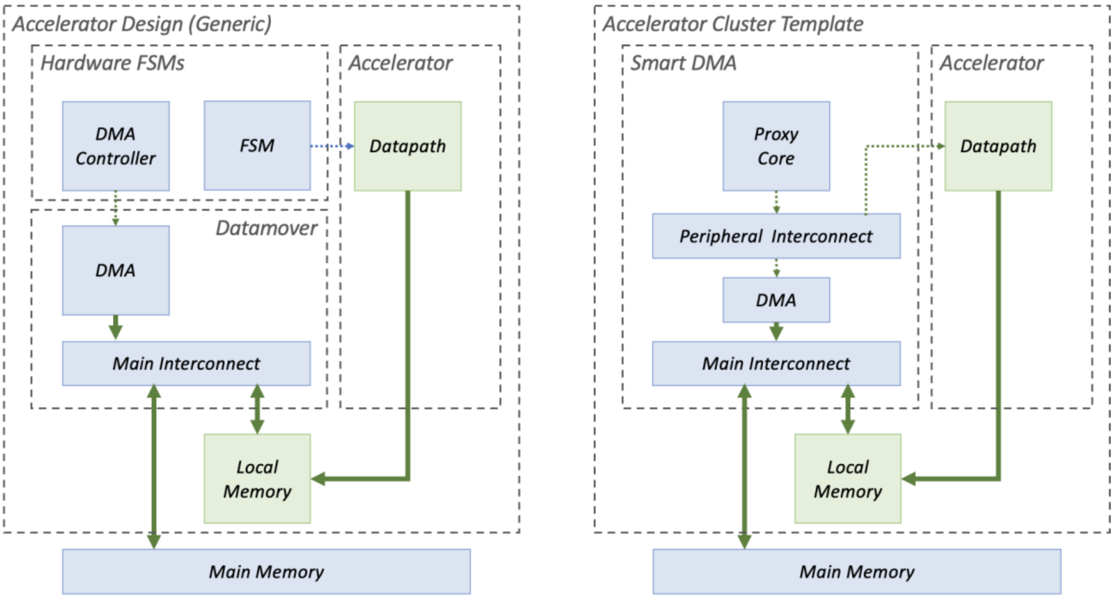


Figure 20 Accelerator Design Options

Problem statement and State-of-the-Art: Our preliminary study regards the impact on memory interference, to which the host cores (APU and RPU) of the two platforms considered (Zynq UltraScale+ and Versal ACAP) are subject, in presence of concurrent execution of memory-bound tasks on FPGA accelerators. When focusing on a FPGA-based HeSoC, a typical way of generating configurable synthetic memory traffic is that of deploying some form of traffic generators.

More in general, full-custom acceleration logic is typically designed as shown in left part of Figure 16, where the core acceleration logic (*datapath*) is coupled to some sort of data mover or DMA engine and a local memory. The data-mover leverages a Finite State Machine (FSM) or DMA controller to supervise the flow of data in and out the local memory as the datapath executes.

"Accelerator Cluster Template" is shown on the right of the Figure. We consider the DMA and local memory as immutable parts of an IP, the Smart DMA, that can be interfaced to different datapaths. To flexibly support different types of control logic for the DMA (and the datapath itself) we rely on a programmable core – the proxy core– rather than on custom FSM logic. This moves the control on the software side, allowing for improved flexibility and reconfigurability for different traffic patterns. The Smart DMA component allows DMA-based memory accesses in a controlled manner, which can be used to ensure that the bandwidth request generated from the FPGA accelerators does not impact the performance of other active cores/tasks in the system beyond what can be tolerated.

In this case we made use of the Smart DMA component to generate a variable amount of interference on the main memory, in order to study its behavior on the two reference platforms, Zynq UltraScale+ and Versal ACAP.

Component **WP3T32-08** and **WP3T32-11** aim to increase the stability of PULP type nodes when real-time critical applications and low priority user applications run on the same processing system. WP3T32-08 adds real-time awareness to the instruction cache hierarchy. The main goal is to prevent a low priority user application to flush the instruction cache and thus increasing the runtime of real-time critical code. An unpredictable runtime can lead to failures and must be prevented as much as possible. This is achieved by splitting the banks of a m-way associative cache in a set of real-time (RT) and non-real-time (non-RT) banks. If a real-time critical thread is handling an instruction cache miss, it will set the RT-flag and indicate to the cache that the cache miss should be handled with priority and that the corresponding instruction can be stored in any of the cache banks. If the RT-flag is not set (low priority thread), the refill FSM will still handle the miss and return the corresponding instruction, but it will select one of the non-RT banks, and hence, never flush real-time critical code. In case of a hit, both type of threads will be able to fetch the instruction regardless in which bank it is located.

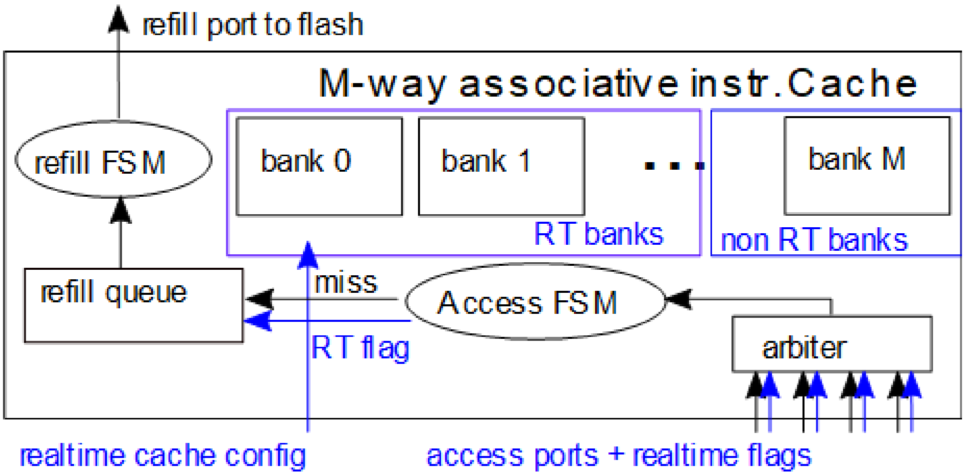


Figure 21 Real-time aware instruction cache design, part of Component WP3T32-08

WP3T32-11 adds a real-time aware hardware scheduler to distribute interrupts to the best possible processing element. The goal is to prevent real-time critical threads from being interrupted by low/high priority interrupts because this can break real-time requirements and lead to failures. To achieve this the interrupt controller has been extended with an additional configuration that allows to declare interrupts as real-time (RT). In addition, the scheduler can look at the current state of each processor and decide to which processing element an interrupt is sent. The priorities for the interrupt controller are:

- Idle core
- Non-realtime core
- Real-time core (only if interrupt is declared RT)

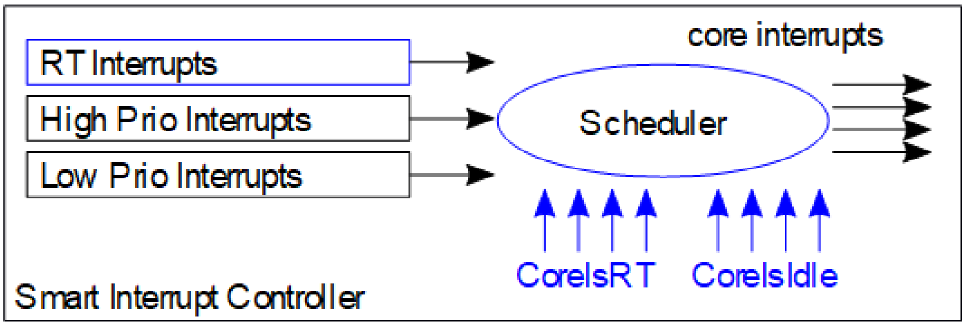


Figure 22 Smart input distribution system, part of component WP3T32-11

4.3 Supporting FRACTAL developments on security

As an open platform PULP based systems have a long track record in supporting research in security for evaluating changes and adaptations needed. PULPissimo based systems have already been designed to support accelerators for commonly used cryptographic primitives like AES and SHA3 and such solutions could also be adapted to the needs of FRACTAL partners. In addition, the Ariane/CVA6 core has been used in recent work to suppress timing channels[14].

As explained in previous section, the AXI PULP interconnect will be extended to mimic the behaviour of the WorldGuard specification.

The VERSAL platform supports several functionalities that could be used for device-level security such as boot image encryption and authentication. Additional functionality can be added through the Programmable Logic (PL) resources of the VERSAL platform as well.

The specifics of how the security related capabilities of these platforms are being utilized in use cases will be described in Deliverables of WP7 and WP8.

4.4 Supporting FRACTAL developments on cognitive awareness

As an open platform, PULPissimo offers a wide range of possibilities to accelerate machine learning applications that can be used for cognitive awareness and support the software infrastructure. These include:

- Hardware accelerators connected over standard interfaces (APB/AXI).
- Hardware accelerators with shared memory access.
- Clusters of cores working as an accelerator to a main core.
- Instruction set extensions to RISC-V ISA.

These options give a lot of flexibility to FRACTAL partners that are working on solutions in this field.

One of the key features of the VERSAL platform are the AI engines that can be used to efficiently map machine learning applications. We expect that these will play a major role in implementing hardware assisted algorithms that support cognitive awareness.

As part of the FRACTAL platform a customized HW accelerator for cognitive awareness integrated with Ariane/CAV6 is under development. For its implementation we use Siemens Catapult HLS. The architecture of the HW accelerator consists of the computation part and the local memory. The RTL for computation component is derived automatically by exploiting the HLS capability of Catapult tool and its features for pipelining and loop unrolling for performance improvement. The local memory is split in two independent partitions. Thus, one partition of the local memory supplies the computation with data, the other one fetches the data from the main memory and vice versa. The HW accelerator performs only convolution part, while all pre- and post-processing for cognitive awareness is done on Ariane/CAV6 core.

[14] N. Wistoff, M. Schneider, F. K. Gürkaynak, L. Benini and G. Heiser, "Microarchitectural Timing Channels and their Prevention on an Open-Source 64-bit RISC-V Core," 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021, pp. 627-632, doi: 10.23919/DATE51398.2021.9474214.

Component WP3T32-07

State of the art: Recently, automatic image-based demographic classification has been exploited into industrial applications such as surveillance monitoring, security control, and targeted marketing systems. Implementing a demographic classifier on embedded platforms can extend its applicability to a wider variety of fields in mobile services, such as human-robot interaction[15]. Electronic Customer Relationship Management (ECRM)[16] is another technology that facilitates the marketing of customized products and services on the base of customer age and/or gender in a non-intrusive and automatic way. Many of such systems demand a real-time demographic classifier able to process 15 to 25 frames per second (fps). To this end, the arising challenge is to provide such performances despite the constrained memory and computation power of the embedded systems.

In recent years, due to their good features extraction ability, Convolutional Neural Networks (CNNs) have been exploited in machine learning and pattern recognition fields, having high accuracy in the results[17]. The development of CNNs requires one or more benchmark databases to perform training, validation, and testing.

State of art has proposed different solutions for age estimation and gender classification implemented on embedded devices (see, for example [18] [19] [20] [21]). However, none of them consider the execution of workloads on highly heterogeneous platforms, such as heterogeneous SoCs with FPGA (e.g., Xilinx Versal), where multiple tasks contend for shared resources, causing interference that can lead to the violation of real-time requirements.

Description:

The age estimator and gender classifier accelerator represent a single component that implements the tasks of age estimation and gender classification by using either the Deep Learning Unit (DPU) on Xilinx Zynq Ultrascale+ SoC or the AI engine on Versal SoC.

[15] C. B. Ng, Y. H. Tay, and B. Goi, "Vision-based human gender recognition: A survey", CoRR, vol. abs/1204.1611, 2012

[16] T. Reponen, ed.,Information Technology Enabled GlobalCustomer Service. Hershey, PA, USA: IGI Global, 2002

[17] Di Mascio, T., Fantozzi, P., Laura, L., Rughetti, V. (2022). Age and Gender (Face) Recognition: A Brief Survey". In Methodologies and Intelligent Systems for Technology Enhanced Learning, 11th International Conference. MIS4TEL 2021. Lecture Notes in Networks and Systems, vol 326. Springer, Cham. https://doi.org/10.1007/978-3-030-86618-1_11

[18] Pyramics Pysense https://pyramics.com/en/products/

[19] AXIS, Demographic Identifier, https://www.axis.com/it-it/products/axis-demographicidentifier

[20] Nagnath Y, Kao C C, Sun W C, Lin C H, Hsieh C W (2020) Realtime Customer Merchandise Engagement Detection and Customer Attribute Estimation with Edge Device

[21] Lee J-H, Chan Y-M, Chen T-Y, Chen C-S (2018) Joint Estimation of Age and Gender from Unconstrained Face Images using Lightweight Multi-task CNN for Mobile Applications

The component **WP3T32-07** is composed of:

- two CNNs, one for age estimation and one for gender classification, accelerated using both Deep Learning Unit (on Xilinx Zynq Ultrascale+ SoC) and AI Engine (on Versal SoC).
- Mechanisms to keep the response time under control in case of interference on main shared memory.

The component is integrated with the component WP3T32-09 (Runtime Bandwidth Regulation) from UNIMORE, that allows to take control of the response time of the whole task when multiple actors share the DRAM external memory.

Due to delivery difficulties with the VERSAL board (as highlighted in the Risks and Mitigation plan chapter), after developing the CNNs, they have been gradually integrated considering heterogeneous platforms similar to Versal. In this section, we report on the evaluation of the Age estimation and Gender classification workloads execution on several edge-computing platforms: a GPGPU (NVIDIA Jetson Nano), and two SoCs with FPGA and dedicated accelerators (Xilinx Zynq Ultrascale+ on ZCU102 board and Xilinx Versal on VCK190 board).

The Age and Gender CNNs are based on a VGG16 CNN. They have been trained using images of MORPH data set, running on a host composed of an Intel i7-9700K CPU processor with 8 cores working at 3.60 GHz, a NVIDIA Titan XP, and 32 GB of RAM. The data set has been divided in training set (70% of data set) and test set (30% of data set).

Table 3 reports data related to the acceleration of the Age estimation and Gender classification CNNs on the considered edge-computing platforms. For each platform, the inference is performed using the test set images. The quality of the inference is obtained by evaluating the Mean Squared Error (**MSE** in the Table 2) for the Age network (as it is a regression problem), and the Accuracy (**Acc** in the Table 2) for the Gender network (as is a binary classification problem). In addition, the response time and the occupied RAM are evaluated for the processing of a single image.

Focusing on Age estimation CNN, Table 3 shows that the inference on the host PC is executed in 68 ms with a MSE of 44.11 (first row). The MSE value is acceptable, as it considers the cumulative error all over the whole range of Age (from 16 to 77 years old). Results on GPGPU give an inference in 428 ms, while occupying 1.75 GB of RAM. The inference on Xilinx Ultrascale+ (using the DPU accelerator) requires an 8-bit quantization of the network. The results show that the MSE diverges, while the inference time is 52 ms (third row). To understand the effect of quantization on the predictions, the quantized neural network has been run also on the host (first row), showing that an 8-bit quantization produces an MSE equal to 51; that means that the divergence of the MSE for TPU and DPU is not caused by the network quantization.

Results of Gender classification CNN, reported in Table 3, follow the same trend of the Age ones.

While the work on WP3 has officially ended, UNIVAQ will continue supporting further the debugging of execution on DPU accelerator, and the evaluation of accuracy and response time by accelerating both CNNs on the AI engine of Xilinx Versal (with support by PLC) as part of their involvement in WP8.

Table 3 Acceleration of Age (Blue) and Gender (Grey) neural networks on edge computing platforms. OF stands for overflow, while Time column reports inference time for one image

			V1		
Device	**Type**	**Q**	**MSE/ Acc (%)**	**RAM (GB)**	**Time (ms)**
Host	Age	No	44.11	-	68
	Gen	No	95.3	-	36
	Age	8 bit	51	-	-
	Gen	8 bit	**54**	-	-
Jetson Nano	Age	No	49.53	1.75	428
	Gen	No	92.7	1.8	470
Xilinx ZCU102	Age	8 bit	**5192**	0.214	52
	Gen	8 bit	**51**	0.224	51

5 Interaction of UCs with FRACTAL nodes

As part of WP2, one of the first actions of FRACTAL was to determine the use case requirements, which were detailed in D2.1. The results were then discussed internally at regular FRACTAL meetings and the following observations were made:

- **UC1** *Edge computing technologies applied for engineering and maintenance works* (PROI):

  Two separate use cases were identified, one for an autonomous drone application (UAV supervision of critical structures), and the second one involving workspace safety (Wireless Sensor Network for safety at construction sites).

  PROI has chosen using both VERSAL and PULPissimo as part of its evaluation.

  UC1 (Demonstrator 1): UAV supervision of critical structures

  This demonstrator is focused on the supervision of critical facilities, where images of the structural status will be collected by the use of UAVs. The obtained images are analyzed with a visual segmentation system, so the families of cracks can be detected and categorized.

  The computational requirements of the crack detection system severely limit the hardware options to be used. Two options have been considered. First, it has been considered the use of a combination of PULPissimo as a low-end node and VERSAL as a high-end node (cloud). The PULPissimo node will perform fewer demanding tasks, such as image collection or detection of crack families. The more complex tasks, such as the execution of the image segmentation system, will be delegated to the VERSAL node.

  In addition, it has also been considered the usage of a medium-end more traditional node that is capable of handling the computational demand of the crack detection system, without the need of a high-performance hardware such as the one offered by VERSAL. However, this second option is still under study.

  UC1 (Demonstrator 2): Wireless Sensor Network (WSN) for safety at construction sites

  This demonstrator is focused on monitoring of both workforce and machinery within a construction area through the deployment of a WSN that will provide information about the status and location of the workers, the workforce in real time. This information will be managed through an IoT platform, registering possible dangers and alarms, apart from establishing a protocol in case of emergency.

  The demonstrator requires wearable sensors to be sending data from the workers and machinery location, real-time processing of this data and then

Machine Learning treatment for alert notifications and potential risk predictions. Although these tasks may not require really high computational capabilities, there exists a necessity of a stable OS (Linux distro preferably) which is a constraint that must be supplied by the platform. For this reason, the PULP platform may be not sufficiently adequate to fulfil the OS requirement. On the other hand, VERSAL is clearly over provisioned for the demonstrator, as only a fraction of its computational power is required, so an intermediate platform with a medium-end node is presumably going to be used.

- **UC2** *Automotive air path control* (AVL):
  This use case will be using VERSAL as a development platform as AVL is making use of the Vitis-AI toolflow.

- **UC3** *Smart meters for everyone* (ACP):
  As a pure IoT application, this use case will develop a demonstrator of a smart meter system using the PULPissimo node.

  While the final system will be battery operated, the demonstrator is based on the FPGA implementation of the PULPissimo node. Low power features such as clock gating and sleep modes, as well as the possibility of a full integration into a microchip are important to be able to run the system from a battery. These are the main reason why PULPissimo is selected for UC3. Another important aspect of UC3 is security. We want to make sure that the firmware is not altered and is verified during each boot process. Further the system needs a service for authentication and encrypting/decrypting user data. To achieve this an open-source root of trust (OpenTitan) has been studied. OpenTitan is open-source and features many of the requirements of UC3. Since both, OpenTitan and PULPissimo are open-source, it is possible to connect the two systems. However, OpenTitan uses a different bus technology "tile link uncached lightweight" (TLUL) than PULPissimo. To be able to connect OpenTitan with the PULPissimo node, two protocol adapters (AXI2TLUL, TLUL2AXI) have been implemented. (**Component WP3T32-12**)

  To allow a user application to run on the same processing system as a real-time critical application such as the protocol stack of the modem in UC3, a real time aware cache and interrupt unit is being used. (**Components WP3T32-08, WP3T32-11**). This allows the fractal node and the modem to share many resources such as processors, memory, and peripherals and thus result in a smaller SoC that consumes less power.

- **UC4** *Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications* (SIEM):

  The use case is a vision-based object detection and recognition system. Its goal is to perform object recognition in real-time with low latency. The detection, localization, and recognition of the objects is done using the tiny-

YOLOv3 AI algorithm, where the most demanding layers are processed by a dedicated AI-Accelerator. The use case uses a camera for video input generation and a display for the output of the results. All image processing is performed on FRACTAL edge node based on CVA6 (**Component WP3T33-03**) under Linux OS (**Component WP3T36-01**).

The part of the node responsible for the execution of the convolution as part of the AI algorithm is a very specialized hardware accelerator (**Component WP3T32-01**) with high computation capacity for inference in order to fulfill the soft real-time requirement for image processing.

The FRACTAL edge node runs only the inference of tiny-YOLOv3. When a new object needs to be added for recognition, the AI algorithm is pushed to the FRACTAL cloud for retraining due to its higher computation capacity.

After retraining the new model is transferred back to the edge node to perform further inference. (**Components WP5T52-04-03, WP5T52-04-05, WP5T52-04-07, WP5T52-05-01, WP5T54-01-02, WP5T52-04-10**)

- **UC5** *Increasing the safety of an autonomous train through AI techniques* (CAF):
  The processing requirements (high-definition video, real-time computation) for this system calls for the VERSAL platform.

- **UC6** *Elaborate data collected using heterogeneous technologies - Intelligent Totem* (Aitek, Univaq, Modis, Rulex, Unimore, Unige, RoTechnology):

  The UC6 is focused on creating an infrastructure for a sentient space in a commercial mall, where fractal nodes are distributed and have the goal of supporting different types of users (e.g., buyers, vendors, commercial mall administrators) in different scenarios.

  Different sensors are used to collect images and sounds. The obtained data could be analyzed, for example, with people detection, idiom recognition, and age and gender recognition tasks, so that the data can be used to decide how to interact with the user supporting in solving the issue.

  These tasks are executed at the edge (on the node), both in standalone mode (all the tasks executed in the node) and fractal mode (the computational load is shared with nodes nearby).

  The computational loads conduced on considering as hardware option the usage of Versal in the nodes, associated with accelerators for the above mentioned more demanding tasks; these accelerators are going to be implemented either on the FPGA side or dedicated ASIC (off-chip connected). The nodes are able to interact to share the computational load.

- **UC7** *Autonomous robot for implementing safe movements* (VIF):
  With UC7 two driving functions of the mobile hardware-in-the-loop platform SPIDER are implemented on a Noel-V.

  The collision avoidance function and the path tracking function are both processing sensor data on the edge while exchanging data with cloud nodes. Both functions are safety relevant, thus usage of FRACTAL safety services, especially a software light-lockstep function will be integrated.

  The collision avoidance function further uses an AI function for obstacle avoidance based on a neuronal network. An AI accelerator from FRACTAL will be used to ensure the necessary latency for the computationally intensive functions.

- **UC8** *Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse* (BEE):

  The designated use case platform is VERSAL as the UC currently uses cameras with higher resolution and computational demands, but there are considerations in using a RISC-V based system such as Ariane and/or PULPissimo if the capabilities of the system are sufficient for the UC.

Table 4 Summary of use cases, their inputs and HW nodes that they will use

Use Case	Short Description	Inputs	HW nodes
UC1	Drones for construction	Images	VERSAL / PULP
UC2	Automotive Airpath Control	Time-series	VERSAL
UC3	Smart meters	Images	PULP
UC4	Object detection	Images / Video	ARIANE
UC5	Autonomous trains	Images / Video	VERSAL
UC6	Smart totem	Video / Audio	VERSAL
UC7	Autonomous robots	Sensor data	NOEL-V
UC8	Autonomous warehouses	Images	VERSAL / PULP

6 Conclusions

The two main hardware nodes (commercial and customizable) have been made available to all FRACTAL partners. Following discussions regarding the design requirements, it was seen that partners would benefit from additional hardware nodes that fall in between the two default options. WP3 partners have discussed providing such solutions in agreement with other partners from technical WPs 4/5/6 as well as the UCs.

The development and adaptation of HW nodes continues as part of WP7/8 activities that are involved with actual implementations of UCs.

6.1 Risks and Mitigation plans

Before the start of the project, the following hardware platform related risks had already been identified:

- FPGA based PULP platform implementations will not have the necessary performance (speed/cost/power) profile needed for use cases (Medium likelihood/Low Impact).
  For some use cases, this has been indeed the case. However, the issue was not the FPGA implementation of the platform, but the desire to have systems that are more like traditional computing systems complete with running large software ecosystems running on full fledged operating systems (i.e. Pytorch) which is beyond the practical capabilities of the system. In part this has been remedied by providing additional platforms (NOEL-V and Ariane). As the projects mature and partners become more experienced with different FRACTAL platforms, it is highly probable that more partners will make use of the experimental and customizable aspects of the provided systems.
- Diversity/maturity of tools/development environments for RISC-V systems is lower than expected (Low likelihood/Medium impact).
  As mentioned in the project proposal, the development rate of the RISC-V ecosystem is quite high, and so far this has not been a major issue.
- Unable to properly and timely integrate multiple services developed. Due to incompatibilites or services not provided by selected hardware platforms (Low likelihood/impact).
  The integration of services are still at an early stage, so it is too early to tell about the possible impact. However, the effort in WP2 has allowed partners to anticipate some of the issues, and there is good momentum in the project that leads us to believe these issues could be mastered.
- Difficulties to integrate hardware developments from different partners (Medium likelihood/impact).
  This is one of the main challenges that faces the developments around the HW platform at the moment. However, all partners are aware and are looking for solutions. In some cases, one solution will be to demonstrate technical solutions running in parts, and not in concert for a given platform, use case combination. I.e. a security service maybe demostrated on a smaller scale,

allowing the use case partner to be able to judge and evaluate its impact, but the overall use case could still use a more traditional approach.

Part of the mitigation efforts also led partners to add two additional hardware nodes, as not to spend initial efforts on porting previous work from architectures they were familiar with to the two nodes presently available.

In addition the following issues have been detected, and efforts have been put in place to mitigate the effects:

- Delivery difficulties with the VERSAL board. There is a global shortage on electronic components, and unfortunately the delivery of the development boards have been hit with longer delays than anticipated. For some of the projects that do not rely on exclusive VERSAL properties, suggestions were made to use more previous generation Xilinx MPSoC boards until the VERSAL shipments can be organized.
- Node computation demands are too high. Some partners on technical workpackages are working on solutions that require significant resources from the hardware nodes. While the VERSAL board can satisfy these requirements, both its price and its power envelope is higher than what could be expected for IoT applications.

  The addition of a lower-end (mist) node will help address this issue. The FRACTAL system will have nodes with less capabilities that defer to more capable nodes for higher complexity operations. This will allow hardware nodes within mW power envelope to be part of the FRACTAL system.

7 Deviations from workplan

Following the feedback from the first project review, it was decided to align the deliverables and reports to a consolidated big picture of the FRACTAL project. This big picture was discussed and agreed on as part of the 2nd Technical Workshop that took place in early February. As a result, D3.3 as well as this follow-up D3.5 were delayed slightly allowing the deliverables to be updated.

It must be noted that lack of face-to-face meetings due to COVID restrictions have impacted such discussions, and while the FRACTAL consortium has put in a very good effort to maintain virtual meetings and discussions, the level of interaction of such meetings cannot replace hands on workshops and technical discussions to bring agreement between a large group of participants. We are happy to report that the last technical meeting was held face to face and has brought back much needed personal interaction. Unfortunately, the efforts of WP3 have been completed, but partners are actively working on supporting UCs in WP7/8 until the end of the project.

8 List of Abbreviations

ABI	Application Binary Interface
ACAP	Adaptable Compute Acceleration Platform
AE	Adaptable Engines
AHB	Advanced High Performance
AIE	Artificial Intelligence Engines
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application Specific Integrated Circuit
AXI	Advanced Extensible Interface
CCF	Common Cause Faults
CCIX	Cache Coherent Interface for Accelerators
CNN	Convolutional Neural Networks
COTS	Components of the Shelf
DDR	Double Data Rate
DMA	Direct Memory Access
DRAM	Dynamic Random-Access Memory
DSP	Digital Signal Processing
ECRM	Electronic Customer Relationship Management
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FUB	Functional Unit Block
GPU	Graphical Processing Unit
HeSoC	Heterogenous System on Chip
HW	Hardware
I/O	Input / Output
IoT	Internet of Things
IP	Intellectual Property
ISA	Instruction Set Architecture
ML	Machine Learning
MPSoC	Multi-Processor, System on Chip
NoC	Network on Chip
OpenAMP	Open Asymmetric Multi Processing
OS	Operating System
PCIe	Peripheral Component Interconnect Express
PE	Processing Element
PL	Programmable Logic
PMC	Platform Management Controller
PMU	Performance Monitoring Unit
PPA	Power Performance Area
PS	Processing System
PU	Processing Unit
PULP	Parallel Ultra Low Power
RFR	Register File Randomization
RTL	Register Transfer Level
SDK	Software Development Kit
SIMD	Single Instruction Multiple Data
SRE	Staggered Random Execution
SW	Software
TLB	Translation Lookaside Buffer
TLUL	Tile Link Uncached Lightweight
UC	Use Case

VLIW	Very Large Instruction Word
WP	Work Package
XRT	Xilinx Runtime

9 List of figures

10 List of tables