D2.2 Methodological Framework (a)

Deliverable Id:	**D2.2**
Deliverable name:	**Methodological Framework (a)**
Status:	**Final**
Dissemination level:	**Public**
Due date of deliverable:	**2021-06-30 (M10)**
Actual submission date:	**2020-06-23**
Work package:	**WP2 "Specifications & Methodology"**
Organization name of lead contractor for this deliverable:	Thales Research & Technology
Authors:	Jérôme Quévremont, Thales David Faura, Thales Pierluigi Scarpa, Modis Andrea Lisi, Modis Daniela Angela Parletta, Modis Lauri Lovén, University of Oulu Teemu Leppänen, University of Oulu Daniel Onwuchekwa, University of Siegen Ruben Lorenzo, BSC Sergi Alcaide, BSC Jaume Abella, BSC Matti Vakkuri, HALTIAN Antti Takaluoma, OFFC Janne Rosberg, OFFC Leticia Pascual, Solver Machine Learning Bekim Chilku, Siemens Carles Hernández, UPV Christina Schwarz, AVL Milan Zivadinovic, AVL Igor Bisio, University of Genoa Andrea Sciarrone, University of Genoa Frank K. Gürkaynak, ETH Zürich Enrico Ferrari, RULEX Gianluca Brilli, UNIMORE

	Iñaki Paz, LKS Luigi Pomante, Università degli Studi dell'Aquila Gabriella D'Andrea, Università degli Studi dell'Aquila Ernst Wehlage, PLC2 Alexander Flick, PLC2 Juan Manuel Garcia, QUALIGON
Reviewers:	Leire Rubio, Ikerlan – technical coordinator Cristina Ganado Ateaga, PROINTEC Bernhard Peischl, AVL

Abstract:
D2.2 "Methodologic Framework (a)" introduces an initial methodological framework specification. It is presented as a compositional workflow, which introduces the interactions of FRACTAL building blocks towards the integration of the FRACTAL computing node and the use cases. Following this global picture, the deliverable focuses on methodologies for several important topics for the project: AI and safe autonomous decisions, certification of safety-related products and integration of FRACTAL platforms.

Contents

Acknowledgement

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Austria, Germany, Finland and Switzerland.

1 History

Version	Date	Modification reason	Modified by
0.0	2021-02-25	Agreed template	Thales
0.1	2021-05-27	Complete version	Authors
0.2	2021-06-10	Reviewed version	Reviewers
0.3	2021-06-23	Reviewers' remarks solved	Authors
1.0	2021-06-23	Final cleanup, delivered version	Thales

Table 1 – Document history

To cope with the high number of contributors, this document has been edited online. The Microsoft Sharepoint solution has been selected to keep information under EU legislation. This solution offers a reduced feature set compared to a "regular" Word editor. For instance, we have not been able to build a table of references and have instead used footnotes.

2 Summary

The task T2.2 is described in FRACTAL DoA as "For the integration of the FRACTAL platform in an industrial environment, an important aspect is to describe (1) how it should be used and (2) how this usage helps to qualifications and certification of products developed using it, including safety-critical products."

Accordingly, D2.2 "Methodologic Framework (a)" introduces an initial methodological framework specification. It is presented as a compositional workflow, which introduces the interactions of FRACTAL building blocks towards the integration of the FRACTAL computing node and the use cases.

Following this global picture, the deliverable focuses on methodologies for several important topics for the project:

- AI and safe autonomous decisions
- Certification of safety-related products
- Integration of FRACTAL platforms

A list of abbreviations is available at the end of the document.

An update, D2.4 "Methodologic Framework (b)", is planned at M30 (2023-02-28).

3 Introduction

This deliverable provides an initial methodological framework specification. The aim of this framework is to identify the key enabling technologies with their supporting methodology and tools.

Development of the FRACTAL node product and its sub-products (or sub-components) will be supported by this workflow.

This document is structured into 4 main sections. First the overall workflow of the project will be presented in section 4. Then sub-workflows/methodologies for "AI and Safe Autonomous Decisions" are presented in section 5, for the "Certification of Safety Related products" in section 6 and for the "integration of FRACTAL platforms" in section 7.

Different WPs provide or use components for composition. This requires prior alignment of what is provided and what is expected: functional boundaries, interfaces, and other necessary information. Collaboration within the workflow includes handover of these artifacts between stakeholders and workflow steps while ensuring, managing, and maintaining composability during the workflow.

4 Workflow of the project

The **FRACTAL** platform specifications (see deliverable D2.1 for more details), as shown in Figure 1, define what the framework should provide (i.e. components, tools, methodology and workflow) to be key enabling technologies (KETs) for the FRACTAL node. To limit the scope and better target the domains considered in the project, the use cases specifications are used as inputs. The final specifications should also consider the extensibility and usage of the framework in related domains of application.

The composition workflow (described in Figure 1) is the activity of putting together building blocks. The workflow defines the steps and order to bring together all participants. It must address their individual needs for system composition (see Figure 3).

Figure 3 shows the overall workflow to capture requirements during the **FRACTAL** project. First, the different demonstrators are specified (i.e. scenarios, features, and functional and non-functional requirements) in "Integration and verification" (WP7) and "Case Studies" (WP8). The demos' requirements are then analyzed to get a unified list of requirements and key enabling technologies that are going to be developed during the project. These are identified into "Specifications & Methodology" (WP2). Third, the identified key technologies are characterized and decomposed into the technical work packages: the "node architecture & building blocks" (WP3), "Safety, security and low power techniques" (WP4), "AI & safe autonomous decisions" (WP5), and "CPS communications framework" (WP6). At the same time WP4, WP5 and WP6 can decompose requirements received from WP7 and WP8 and realize they need to add extra requirements into WP3.

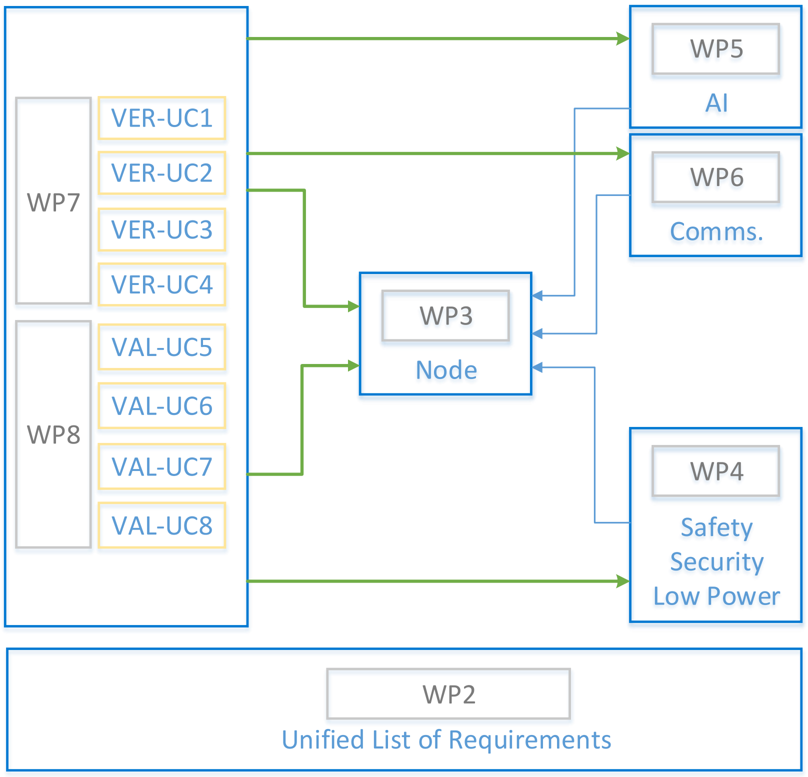

Figure 1 – Requirements workflow in FRACTAL project

Upon reception of external requirements, each Work Package works its internal requirement list and develop its “products”. Work Package’s tasks and their dependencies become visible in workflows like the one in Figure 2. These workflows also make clear dependencies among Work Packages.

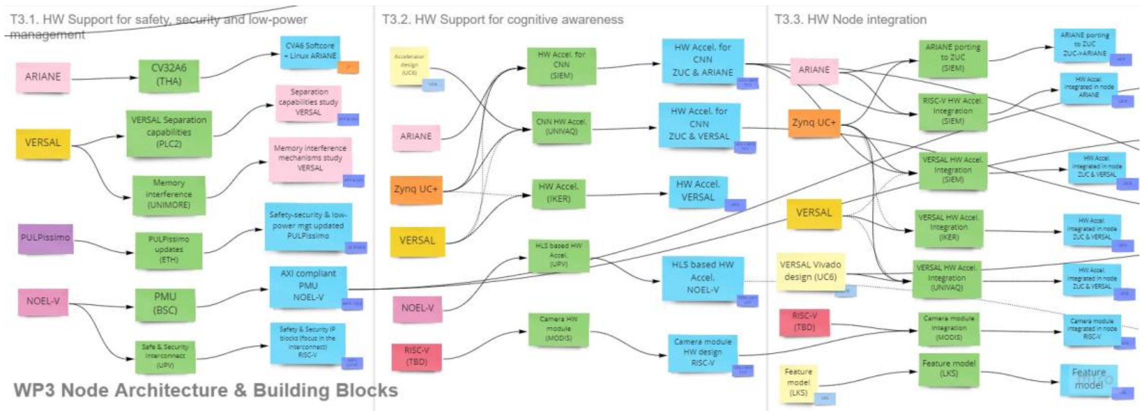
T3.1. HW Support for safety, security and low-power management
T3.2. HW Support for cognitive awareness
T3.3. HW Node integration
ARIANE
VERSAL
PULPissimo
NOEL-V
CV32A6 (THA)
VERSAL Separation capabilities (PLC2)
Memory interference (UNIMORE)
PULPissimo updates (ETH)
PMU (BSC)
Zynq UC+
RISC-V (TBD)
Feature model (LKS)
WP3 Node Architecture & Building Blocks

Figure 2 – Example of workflow

Finally, as detailed in Figure 3, WP3 will provide the node definition and platform. WP4, WP5 and WP6 will add their products on top of the results of WP3. In particular, WP3 provides the hardware and software components (aka *primitives*) and platform nodes that allow building complex services and properties atop in WP4, WP5 and WP6 by smartly combining them in accordance with specific goals. If any problem arises in the integration of WP4/WP5/WP6 with WP3 products it will be solved at this stage. The first global integration attempt will be done in the verification phase WP7 that will help fine tune all products and then everything should be ready to the final validation on WP8.

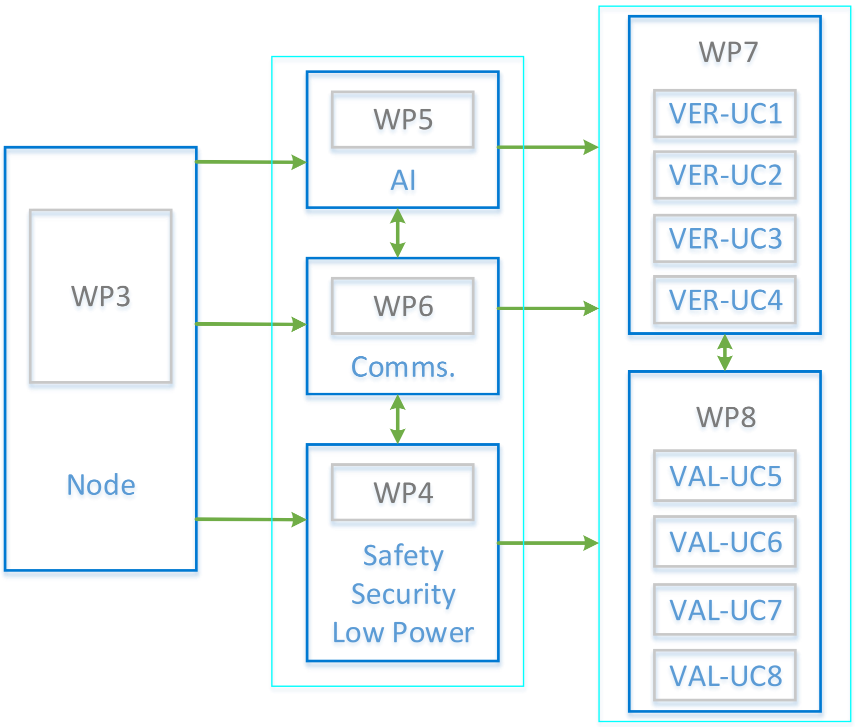

Figure 3 – FRACTAL products composition workflow

5 Methodology and workflow for AI and safe autonomous decision

This chapter looks at the AI building blocks in WP5, focusing especially on the FRACTAL autonomous decision framework (section 5.1), the FRACTAL service architecture and middleware (section 5.2), and a common framework and format for FRACTAL inference models (section 5.3). Further, the chapter will look at AI functionality required by the use cases, concentrating in particular on UC2 (Automotive air path control) (section 5.4) and UC6 (Intelligent Totem) (section 5.5).

5.1 AI and autonomous decision framework

The FRACTAL AI framework aims at integrating the AI functionalities to allow advanced prediction capabilities in the Fractal node. Even if use cases will be a guidance in the development of the AI tools, the whole framework will be planned and developed to be adapted in any context. In general, when talking about AI, a distinction is needed between the training phase and the inferencing phase. In the first one, historical data are used to train the AI models while in the second one the already built models are used to make predictions and decisions about the new incoming data.

If the system's behavior is not changing with high frequency, it is not necessary to perform the training phase very often and, above all, to have a response quickly. Usually, the training phase is done once or is updated on a regular basis if the system is supposed to undergo some changes in the behavior. So, two different scenarios will be considered:

- The training is done offline. Data are transferred manually on some database and the model are trained starting from them. Then the model is manually transferred in the Fractal node where it is supposed to be executed for the inferencing phase. This is probably the approach that will be used for the first tests of the FRACTAL node since it allows a quick implementation of the models in the device.
- The training is done periodically. In this case (see Figure 4) data are automatically transferred to a cloud service (via 5G or Wi-Fi connection) where they are stored in a database and used for building a model. Since it could be unfeasible (due to huge amount of produced data) or inappropriate (for example for security or privacy reasons) to send all the data the FRACTAL will probably be equipped with an aggregation layer able to do some basic processing on the data, to aggregate data and reduce the amount of data to be sent over the network. The model is then sent back to the device where it is used for subsequent elaborations. Notice that in this case the network is not a bottleneck since FRACTAL node can continue its processing even without

the generation and the transferal of new model. So, if for some reason the connection is not guaranteed the system can go on working.

It is possible to imagine also that the training could be performed on the FRACTAL node, but usually the computational resources needed for training a model fit better with a cloud environment.

As regards the inferencing phase, this is performed on the FRACTAL node since the decisions should be taken very quickly, which is not compatible with sending data over the internet. Moreover, the system should work also without connection, so the node must be autonomous in taking decisions.

So, the FRACTAL node will be equipped with a layer able to perform some basic preprocessing operations on the data and an AI module able to use already built models to make decisions about new incoming data. The elaboration must be very quick because in some use cases, decisions are expected to take place within 100 ms. Moreover, the module will be able to do some aggregation on the data; in this way, data could be sent in an easier way to a cloud service where the AI models could be updated.

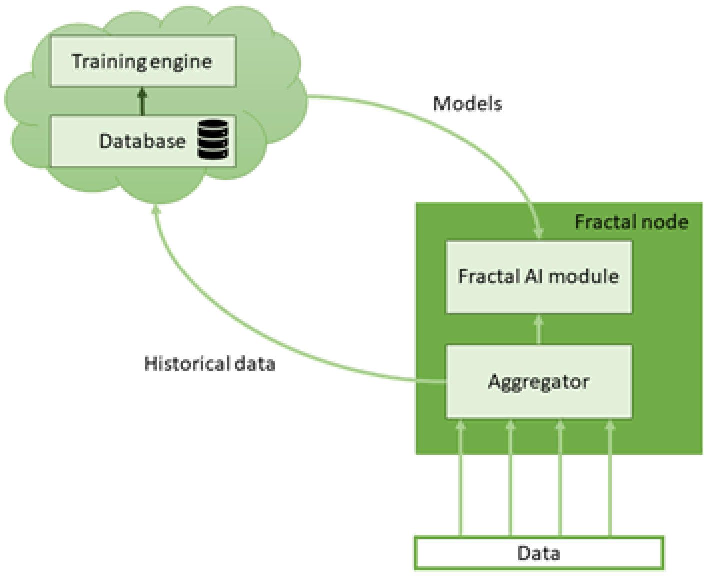

Figure 4 – Functioning of the online training for the FRACTAL node

AI module will include these components that could be activated according to the specific applications:

- Video analysis
- Supervised and unsupervised learning

5.1.1 Video analysis

Video Content Analysis deals with the extraction of information from images and video. Such information can be used for further processing done in "high-level" applications that collect and correlate data from heterogeneous sensors. Modern Video Content Analysis (VCA) systems are based on AI approaches: after a proper training phase, they can understand how to analyse and detect relevant information inside images and video streams.

The first step needed to build an AI-based VCA system is to define exactly the kind of information to be recognized and detected. There are different possible tasks:

- Classification: assignment of images to different classes or groups, according to their content.
- Tagging (or labelling): it is a classification task more complex with respect to the previous one; multiple labels can be associated to an image as the VCA is able to recognize multiple "scenarios" or "concepts" in it. A practical example can be useful to explain differences: classification can distinguish between images collected indoor or outdoor while tagging can add multiple labels to the same images like outdoor, city, road intersection or indoor, house, bedroom for example.
- Detection/Segmentation: both previous tasks are focused on detecting the presence of one or more reference target (object, person, a scenario etc.) in an image or a video. Detection and segmentation can also infer the location of such target(s) inside the image. In particular:
    - Detection process generates as output a bounding box that surround each target detected,
    - Segmentation process detects the shape of each target as it performs a pixel-based decision (i.e., each pixel can be assigned to background or to a specific target).

During FRACTAL, and in particular in Use Case 6, detection and/or segmentation tasks will play a crucial role, as they are extremely useful to the FRACTAL node to understand its surrounding environment. As a matter of fact, targets detection and localization enable further processing like for example counting the number of persons and or reference objects within a specific area. Also target tracking is a quite relevant function in the scope of FRACTAL, in particular for those applications related to safety and security control.

VCA, as any other AI system, is based on a training phase in which the system learns to detect and recognize a target. In general, the training can be supervised or unsupervised. For the VCA, training will be supervised, meaning that the system learns how to perform the detection through a set of labelled (annotated) images. Basically, such annotated dataset (e.g., a large number of images with a person inside) is the input for the training phase; after the training phase, the output is a model that, at runtime, can be used to perform the task (in this case people detection).

Such training phase has a relevant impact on VCA performance, that is dramatically affected by the training set characteristics. In particular, this dataset must include an adequate number of images, it must be accurate, meaning that images used for training should be similar to those analysed at runtime, but at the same time, it must be representative of all the different alternatives that will be possible at runtime.

Overfitting and underfitting are two typical problems of AI-based VCA system: the model achieves poor classification/detection performance after training. In particular, overfitting means that there are too many parameters in the model and a high variability of the classification. Therefore, the model is too complex and sensitive to training dataset (high variance). On contrary, underfitting means that there are few parameters in the model and a high classification discrepancy (high bias). In other words, underfitting can be explained as the model is too simple and therefore unable to provide good results during prediction; overfitting is when the model is too good to be true, as it performs very well analysing training data but it is completely unable to be generalized and therefore achieve very poor results on runtime prediction.

As previously said, the training phase and therefore the training set play a crucial role in the VCA system performance. A limited amount of data for training, in this case annotated images, is one of the worst scenarios concerning an AI-based VCA system. Data augmentation is a useful technique to overcomes this problem. It consists in a manipulation of available images in order to increase artificially the dimension of the dataset. For example, some pictures in the database can be transformed by rotating, flipping them or by modify colour, contrast or brightness. Recently has been emerging a new approach for data augmentation based on the use of Generative Adversarial Network (GAN). Such particular type of Neural Network is here mentioned as it is quite relevant in the AI framework for the VCA. Moreover, it will be taken in deep consideration during the next phases of FRACTAL, in particular in WP5. Several details will be included in the deliverable D5.1 and D5.3.

5.1.2 Supervised and unsupervised learning

Besides the analysis of video and audio streams, other AI tasks could be implemented in the FRACTAL AI framework. In this subsection a short overview of more traditional machine learning applications that could be used to analyse data deriving from use cases. The methods introduced here deal with structured data, i.e. that can be organized in tables. Usually, a distinction is done between supervised and unsupervised methods. Supervised methods assume that data are somehow labelled either in a natural way or because some human has labelled them manually. This label is the target of supervised methods since they aim at building a model able to predict the value of the target starting from a set of inputs. According to the type of target, classification or regression problems could be defined. Multilayer Perceptron (MLP), Support Vector Machine (SVM), Logistic Regression (LR), Decision Trees (DT) are methods for supervised learning.

On the other hand, in unsupervised problems, no target variable is available and the goal is to find information within the data. For example, some unsupervised approaches are:

- Clustering, aimed at organizing data in homogeneous groups.
- Outlier detection, that are devoted at finding configurations that deviate from standard behaviour.
- One-class classification, whose goal is finding a classification model when only data of one class are available. For example, data about failures could not be yet available in historical data in a predictive maintenance application.
- Sequence Analysis and Anomaly Detection, aimed at analysing time sequences to detect frequent or uncommon patterns that could be related to regular or anomalous behaviours.

In general, these approaches could be used in association with the analysis of audio and video streams. As a matter of fact, the analysis of video and audio could generate features that can be used as an input of supervised or unsupervised tasks.

Moreover, it is worth noting that some techniques belonging to this class allow also the generation of intelligible models, according to the Explainable AI (XAI) paradigm, enabling applications where the understandability is a key feature.

5.2 Distributed AI and the AI services in the middleware

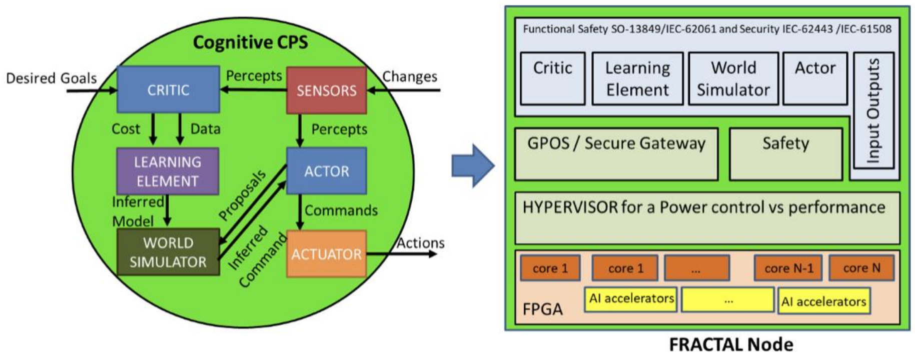

Figure 5 – FRACTAL cognitive agent and the corresponding node architecture.

A FRACTAL cognitive agent (Figure 5) interacts with other system components, devices and data sources through the services provided by the platform middleware. In its operation, it uses and exports both internal and external interfaces for connecting to its sensors and actuators and to external services and data sources, for operational control (e.g., setting the agent goals), and for sharing its results, knowledge and data. The agent architecture is internally composed of software components with varying roles, such as modules for interactions, decision making, implementing and evaluating the selected actions and interactions.

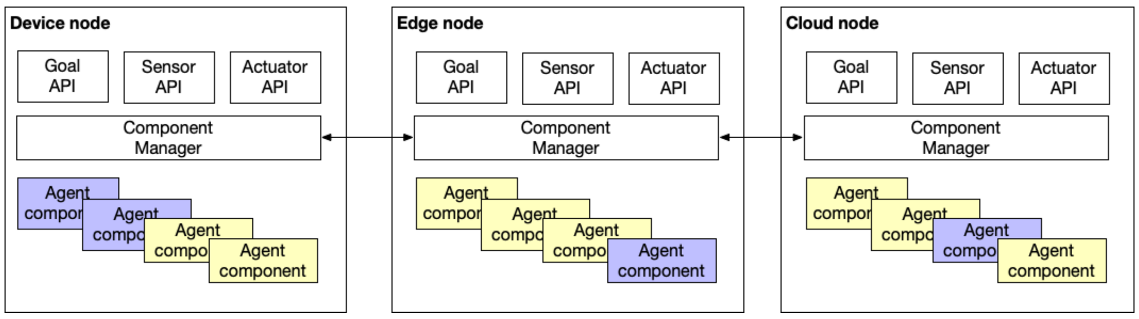

Figure 6 – Agent components and APIs. Active component shown in blue.

To facilitate orchestration and choreography of the operations towards optimal Quality of Service and performance, the run-time deployment of each software component must be decided. To this end, the agent architecture with regard to the application requirements must be considered. For example, some processing-heavy components (e.g., the *learning element*) could be run on a nearby edge node or on the cloud, while others may run on-device (see Figure 6). As a result, the edge-cloud framework provides component online deployment (including offloading and migration), management and monitoring functionalities.

On one hand, the framework enhances the adaptivity of the individual nodes in response to the dynamics of the environment, the state of the platform, and application requirements, by allowing control of its communication-computation tradeoff (e.g. latency vs. data transmission vs. computational load). On the other hand, such a framework increases operational complexity significantly and introduces a need for (partially) autonomous decision-making by the components. Figure 7 further illustrates the related horizontal offloads and vertical migrations.

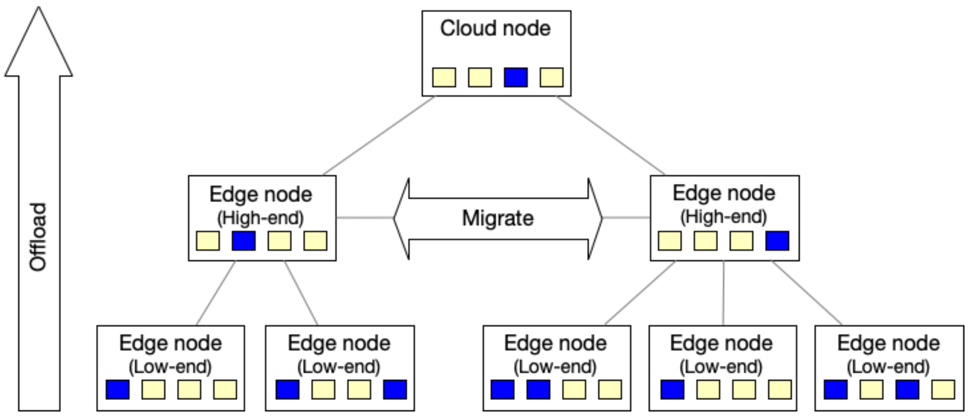

Figure 7 – Hierarchical component transfer framework.

For those of the agent components, which encompass learning and decision-making elements, the framework requires data and knowledge sharing and collaboration across the platform. For example, some components may employ a *federated learning* schema (see Figure 8), where a number of edge nodes, coordinated by a cloud node, collaboratively build a shared understanding (e.g., a model). The architecture, and largely the framework, must thus allow the online sharing of data, results and knowledge, all the while monitoring and evaluating the operation and environments of each component taking part in the learning.

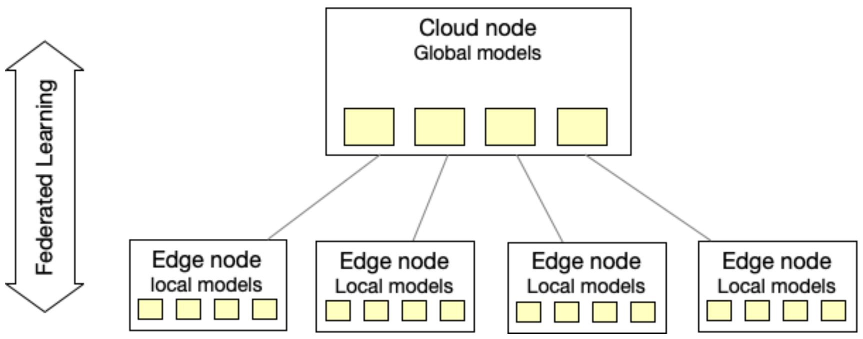

Figure 8 – Federated leaning in a FRACTAL system.

Further, while an agent makes partially autonomous decisions, a number of use cases also call for collaboration and co-operation (e.g. through swarm intelligence) as distributed decision-making, targeting operational efficiency and effectiveness of the system. To facilitate such a multi-agent system, the agent must support sharing of both data and knowledge, leading to complex interactions between the agents and other system components (Figure 9). Such an integrated architecture is also required to support the top-down/bottom-up control in the operational framework, further specified in T5.4.

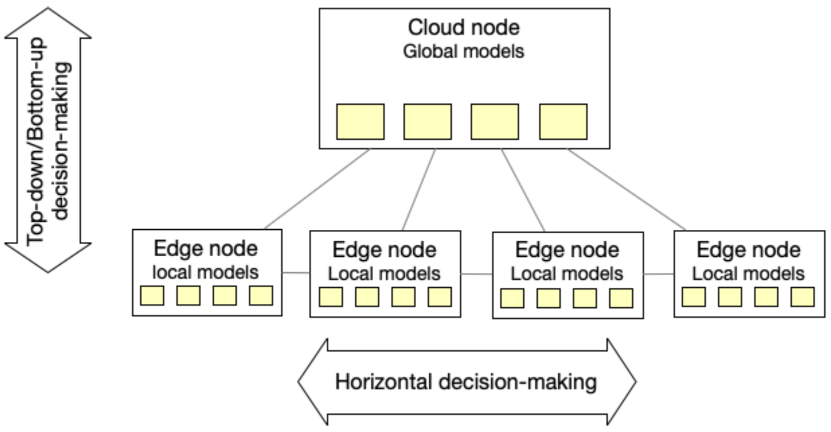

Figure 9 – Distributed decision-making in a FRACTAL system.

5.3 Use of LEDEL

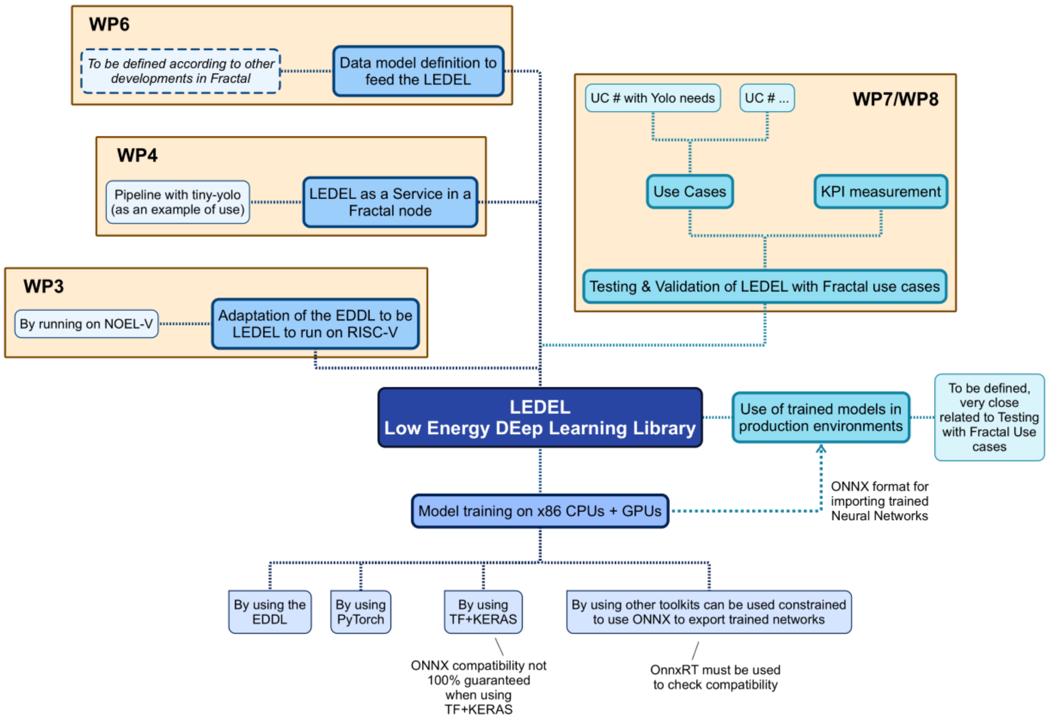
WP6
To be defined according to other developments in Fractal
Data model definition to feed the LEDEL
WP4
Pipeline with tiny-yolo (as an example of use)
LEDEL as a Service in a Fractal node
WP3
By running on NOEL-V
Adaptation of the EDDL to be LEDEL to run on RISC-V
WP7/WP8
UC # with Yolo needs
UC # ...
Use Cases
KPI measurement
Testing & Validation of LEDEL with Fractal use cases
LEDEL
Low Energy DEep Learning Library
Use of trained models in production environments
To be defined, very close related to Testing with Fractal Use cases
ONNX format for importing trained Neural Networks
Model training on x86 CPUs + GPUs
By using the EDDL
By using PyTorch
By using TF+KERAS
By using other toolkits can be used constrained to use ONNX to export trained networks
ONNX compatibility not 100% guaranteed when using TF+KERAS
OnnxRT must be used to check compatibility

Figure 10 – Development stages of LEDEL in Fractal

EDDL (https://github.com/deephealthproject/eddl, European Distributed Deep Learning Library) is a Deep Learning (DL) toolkit designed and developed to provide support to design and train Deep Neural Networks (DNNs) on single computer nodes and on hybrid HPC + Big Data computing architectures. EDDL is ready to leverage hardware accelerators, such as GPUs and many-core CPUs. It also uses the ONNX standard format (https://onnx.ai) to import/export DNNs. Thus, trained DNNs can be used on production environments to infer/predict. LEDEL (Low Energy EDDL) will be the adaptation of the EDDL to run on Low Energy hardware, so that trained DNNs using the EDDL will be easily used to infer/predict on production environments using the LEDEL. In the picture above we can observe different tools provided by EDDL.

LEDEL will be ready to run on Edge computing hardware being developed in FRACTAL. Hardware with limited computing capabilities, but more powerful than existing low energy hardware so far. Thus, thanks to LEDEL, not only will it be possible to make decisions based on simple conditions, but more complex decisions based on indicators provided by more sophisticated algorithms running on the edge will be feasible to be made. As an example, we will be able to evolve from simple presence detectors to decide whether to switch on lights, to much more complex scenarios, where the number of people in a room and the distance between individuals will be computed every few seconds in order to adapt room conditions as light, cooling/heating system, switch on only the required lights, etc.

The tasks to accomplish that are shown in Figure 10 are defined as follows.

- EDDL will be adapted to run on RISC-V based hardware in the NOEL-V processor model proposed (WP3, T3.5).
- In order to check the correct execution of the LEDEL as a software service in a Fractal node, an example of a common object detection algorithm, like Tiny-YOLO, will be implemented (WP4, T4.1).
- Correspondingly, LEDEL will need a data model definition to be used (WP6), that will take into consideration dependencies and requirements of other components developed in WP5 of the Fractal project.
- Finally, SML will provide support to check the integration of the LEDEL in the Fractal platform and guarantee a good performance and a proper behaviour in those use cases that require it.

In conclusion, LEDEL as a service will be offered in a FRACTAL node for it to be able to perform more complex calculations (i.e., to run more sophisticated algorithms). The goal is to develop an API that provides deep learning functionalities that are devoted to face the deployment on low energy computing infrastructures. LEDEL will be accessible in order to make it easy to Fractal partners to use it in their use cases.

5.4 Advanced control strategies in the automotive domain

Existing automotive air-path control strategies are fully reliant on model-based control strategies. These techniques imply a high calibration effort and the ability to perform self-learning through observations is very limited. This use case will, therefore, contribute to integrate the environmental influences and changes as a fundamental part of the system, among other benefits, like potentially increased product quality and increased efficiency for the development of customized air-path controllers. The FRACTAL nodes are crucial to the implementation of this use case.

The FRACTAL framework shall demonstrate the following objectives:

- Inference of data-driven models aimed at improved energy efficiency and reduction of environmental pollutants on the FRACTAL node
- Online self-adaptation algorithms of the initial state model, to react to variations in the combustion engines and different driver behaviors
- Freeze frame data collection and connection to the cloud for re-training purposes
- Identification of potential cyber-security breaches through anomaly detection

In Figure 11 an overview of the planned (implementation) interactive environment schema can be seen. Three different model operations can be differentiated. Firstly, the model inference of the initial state AI-based model, as a result from the model development process, which will be implemented to replace the conventional air path control. Secondly, the model adaption during the vehicle in-use phase, to cover the model blind spots coming from the limitations of the input data used for model training and to have the possibility to adjust to vehicle specific parameters. The adaptation algorithm would compare the output of the model inference with the measured data and in case of a deviation, learn and preserve the additional information. Thirdly, a cloud connection will be established to utilize the potentials of crowdsourcing and the access to extensive information from other drivers/vehicles, with the overall target to optimize the control strategy from many different aspects (e.g., changes in environmental conditions, variability, coverage of different operation modes, etc.). Since this task comprises the use of big data, a computationally heavy training infrastructure is needed and would therefore require cloud computing for the execution.

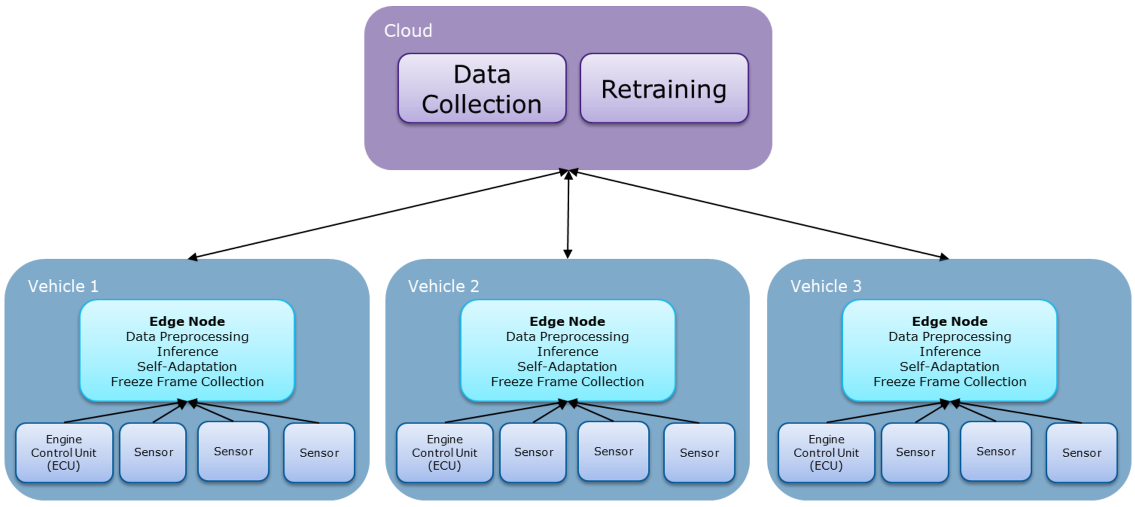

Figure 11 – Overview of FRACTAL framework needed for Automotive implementation

5.5 Image recognition (iris diagnosis using AI algorithms)

At this time, existing models for the fundus image analysis to detect diseases and malformations are still being tested. These techniques imply a human intervention in the pre-processing phase, and a high calibration effort for the fundus segmentation.

This experimentation, therefore, wants to provide a tool to study how machine learning (from now ML) techniques can support the ophthalmology sector in the analysis of the ocular fundus, and at the same time seeks to evaluate the performance and accuracy by comparing the results obtained with those of other accredited research studies.

The following objectives should be demonstrated:

- Implementation of an image acquisition system provided by a camera module.
- Implementation of a Deep Learning process able to obtain an automatic image pre-processing feature.
- Implementation of a classification system.
- Implementation of a ML environment able to perform:
    - An extensive performance evaluation.

  - o An hyperparameter optimization session.
- Implementation of a functionality for tracking the pupil and iris.

In Figure 12, an overview of the planned architecture schema can be seen.

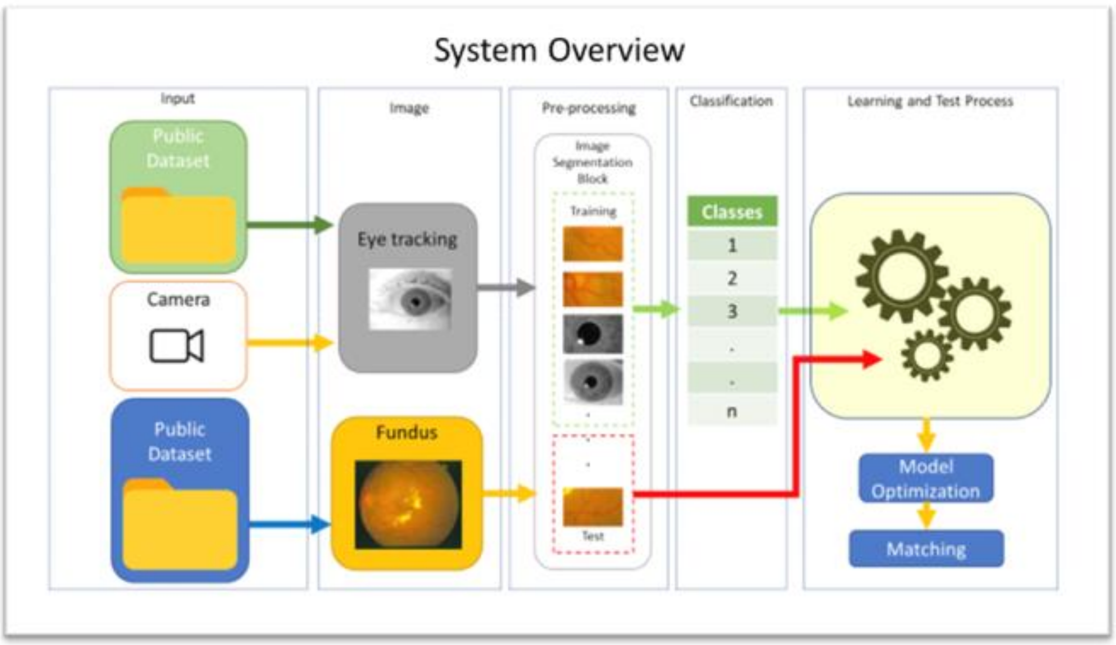

Figure 12 – Iris diagnosis: Overview of the functionality

As shown in the Figure 12, we have divided the system architecture into seven blocks:

Block Name	**Description**
Input	Public dataset for Eye tracking: The system should be able to analyse the images of a biomedical dataset that contain information on malformations and / or pathologies already diagnosed.
	Camera module: System should integrate with an infrared camera to perform eye tracking.
	Public Dataset for fundus: The system should be able to analyse the images of a biomedical dataset that contain information on malformations and / or pathologies already diagnosed.

Block Name	Description
Image Acquisition	This block should be able to do: • Image size normalization. • Discard any images that cannot be used. • Distinguish image type (for fundus or eye tracking). • Transfer the data to the next block.
Pre-Processing	This block should be able to do: • Identify the iris. • Extract the iris image. • Identify the pupil. • Extract the iris pupil. • Trace the path of the veins. • Identify the points of intersection of the veins. • Extract the image of the surroundings of the intersection points of the veins. • Transfer the data to the next block.
Classification	This block should be able to do: • Group images into two subgroups (test and training). • Classification of images to detect malformations and / or pathologies. • Transfer the data to the next block.
Neural Network	This block should be able to analyse the data collected to obtain a cognitive model that allows the optimization of the results collected in subsequent iterations.
Optimization	This block should be able to analyse the output data in order to optimize the accuracy and the computational process in the testing phase.
Matching	This block should be able to analyse the output data in order to compare the results and performances obtained.

Table 2 – Image Recognition for Iris Diagnosis - Definition Table

The results of this experimentation will be compared with those published in research articles consistent with the purpose of FRACTAL nodes performing eye analysis to evaluate the performance, accuracy and integrability of the system with embedded solutions.

6 Methodology for the certification of safety-related products

This chapter proposes methods tailored to the FRACTAL project to anticipate the certification of safety-critical use cases.

This chapter starts with an introduction to get more acquainted with safety standards (section 6.1.1) and derives guidance to anticipate the certification of safety-critical use cases (section 6.1.2).

The section 6.2 addresses functional safety for certain critical building blocks of the FRACTAL project, with topics traditionally addressed in functional safety (redundancy, mitigation of timing interferences) and topics that are quite novel in the safety domain: fractal communications and artificial intelligence.

Finally, the section 6.3 explains how to leverage VERSAL primitives for safety-critical applications[1].

With a few tens of pages, we cannot claim to present a full-fledged methodology to build a certified product when the certification evidence of cars or trains requires several thousands of pages. However, we think these recommendations can help ease the future integration of FRACTAL technologies into safety-critical end-products.

6.1 Guidance to define a methodology for a safety-related development (ISO61508) tailored to an R&D cooperative project

6.1.1 Introduction to safety standards

IEC 61508[2] is the main European standard for functional safety. It provides a generic approach to all activities related to the safety lifecycle of safety-related Electrical/Electronic/Programmable Electronic (E/E/PE) systems that will be used to perform safety functions.

6.1.1.1 Functional safety

According to the IEC 61508 standard, functional safety[3] is the subset of the overall safety relating to equipment and its control system which depends on the correct

[1] Per the preparation of D2.1 specifications, it turned out that the RISC-V based nodes will seldom be used by safety-critical applications.

[2] IEC 61508-1 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements, 2011

[3] MTL application note, Functional safety, An introduction to Functional safety and the IEC 61508 series, 2002

operation of its safety related system which implements the required safety function. Functional safety ensures that there are no unacceptable risks and addresses the ability of safety-related systems to perform their safety functions as intended.

6.1.1.2 Objectives and application domain of the IEC 61508 standard

IEC 61508 standard was first published in the period 1998-2000. The standard was updated and improved with a second version in 2011. The general objective of this standard is to permit the development of E/E/PE safety related systems that will perform safety functions in accordance with the specification. For this, the standard proposes an operational approach to harness the E/E/PE safety-related system, starting from the study of the safety requirements and taking into account all stages of the system lifecycle.

The first intention of the working group was to produce a generic standard to be used as the basis for drafting other product and application sector international standards. However, in practice, IEC 61508 is used directly by industries.

6.1.1.3 General structure of the standard

In order to cover all aspects related to E/E/PE systems, the general structure of standard 61508 is organized in 7 parts (the references to the different parts can be found in the referenced IEC document[4]). The parts 1, 2, 3 and 4 are normative, while the parts 5, 6 and 7 are only informative, offering advice and guidance to apply the normative parts. The part 1 sets the requirements for the certification documentation and the way to be compliant with the standard. It also defines the technical requirements and the associated management and assessment for achieving safety throughout the entire lifecycle of the system, see Figure 13. The parts 2 and 3 cover the requirements for the development of E/E/PE hardware and for the software development while the part 4 provides the definitions used in the standard.

[4] IEC Functional safety Essential to overall safety, 2019; https://www.iec.ch/basecamp/functional-safety-essential-overall-safety

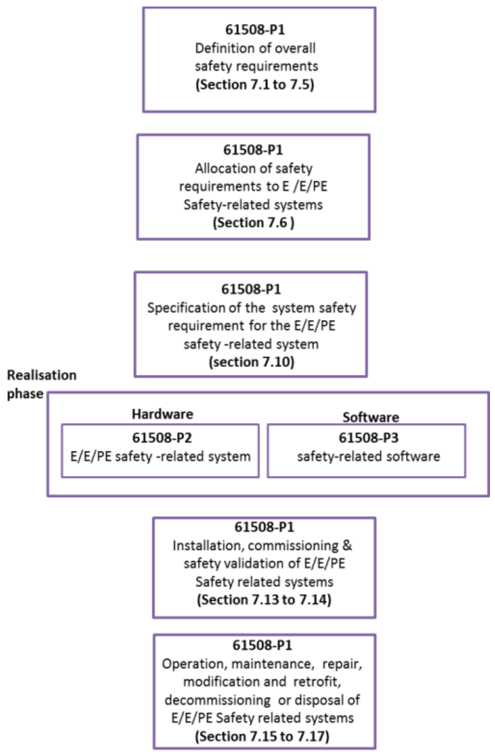
61508-P1
Definition of overall safety requirements
(Section 7.1 to 7.5)
61508-P1
Allocation of safety requirements to E /E/PE Safety-related systems
(Section 7.6)
61508-P1
Specification of the system safety requirement for the E/E/PE safety -related system
(section 7.10)
Realisation phase
Hardware
61508-P2
E/E/PE safety -related system
Software
61508-P3
safety-related software
61508-P1
Installation, commissioning & safety validation of E/E/PE Safety related systems
(Section 7.13 to 7.14)
61508-P1
Operation, maintenance, repair, modification and retrofit, decommissioning or disposal of E/E/PE Safety related systems
(Section 7.15 to 7.17)

Figure 13 – IEC 61508 safety life cycle model

6.1.1.4 Risk reduction

The safety assessment in the IEC 61508 standard is based on risk analysis and risk reduction. In the risk analysis, hazardous events are identified and the necessary risks reduction for these events are determined. After the specification of the risks reduction, the safety requirements will be provided with an associated Safety Integrity Level (SIL) for each safety function and will be implemented into one or more safety-related systems to mitigate the identified risk. The SIL[5] indicates a level of safety integrity (between 1 and 4) and its value depends on the level of risk reduction required by the analysis. The SIL may be defined as a measurement of operational safety that determines the recommendations related to the integrity of the safety features to be assigned to E/E/PE systems.

The standard considers that the risk values are always approximate and the actual reduction obtained by risk reduction measures can never be determined with precision and cannot be zero, see Figure 14.

[5] Felix Redmill, "Understanding safety integrity levels", Measurement + Control, Volume 32, September 1999

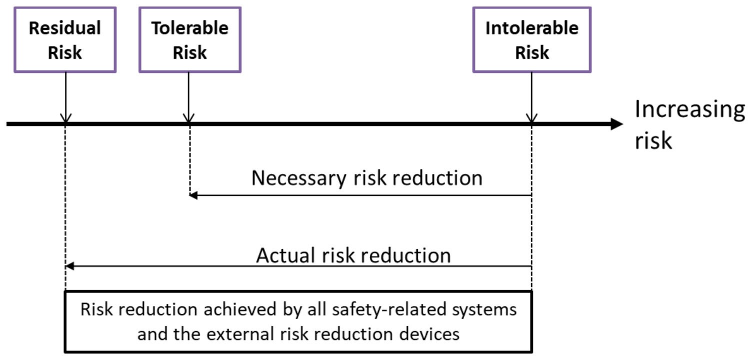

Figure 14 – Risk reduction principle

The risk reduction is linked to the development of the safety functions following the safety standard and it is well described in[6] and requires the following steps:

- Identify and analyze risks
- Determine the tolerability of each risk
- Determine the risk reduction necessary for each intolerable risk
- Specify the security requirements for each risk reduction and their SIL
- Design and implement the safety-related function to meet security requirements
- Validate the safety functions

6.1.1.5 IEC 61508 a stand-alone standard and a basis for other standards

IEC 61508 can be a stand-alone standard. It provides suppliers and users of safety equipment with a common framework for the design of products and systems for safety-related applications. All parts of IEC 61508 are suitable for direct use by the industry.

IEC 61508 parts 1, 2, 3 and 4 are the basic IEC publications in the field of functional safety. One of the responsibilities of IEC technical committees is to base, wherever possible, the drafting of their own industrial or product standards on these four parts whenever E/E/PE safety related systems are within their scope. IEC 61508 is also the basis for other industry standards such as automation[7], railway[8] and automotive domains[9] as shown in Figure 15.

[6] MTL application note, Functional safety, An introduction to Functional safety and the IEC 61508 series, 2002.

[7] IEC 61511-SER Functional safety – Safety instrumented systems for the process industry sector, 2004.

[8] CENELEC EN 50126-1 Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS), 2010.

[9] ISO 26262-2 Road vehicles — Functional safety — Part 2: Management of functional safety, 2018.

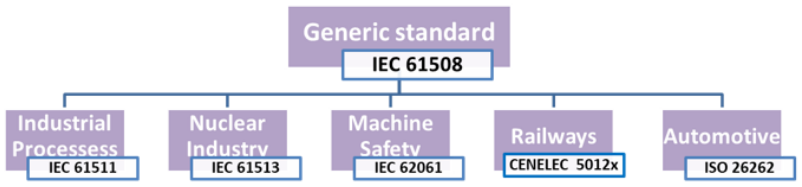

Figure 15 – Industry standards based on IEC 61508

IEC 61508 has a strong impact on the development of E/E/PE systems and multi-sector products concerned with safety. However, it should be noted that specific industry or product standards usually refer only to the specifications of the IEC 61508, therefore the users will always need to consult IEC 61508.

For the application domains addressed in FRACTAL, IEC 61508 forms the basis for:

- ISO 26262 in the automotive sector,
  CENELEC EN 50126, 50128, 50129 and 50159 in the railway sector[10].

6.1.2 Guidance to apply safety standards in FRACTAL

6.1.2.1 Introduction

In the FRACTAL project, we have established a specific methodology derived from the functional safety standards that will give the system designer an early assurance of the reduction of systematic errors and the feasibility of functional, non-functional and safety specifications on some of the building blocks for use cases that require it. This confidence in the different building blocks of the system can be acquired and confirmed as they increase in maturity during or after FRACTAL. In order to guarantee the expected final safety level, the system "industrialization" project should adopt one of the recognized functional safety standards such as IEC 61508 in early stages.

6.1.2.2 Methodology used in the FRACTAL project

In this section, we present a simplified methodology, derived from the functional safety standards, which can be used by the FRACTAL partners. This methodology is available for use cases that are interested in increasing the level of confidence in the feasibility and viability of their building blocks that will be introduced in the industrialization phase of a system with operational safety constraints.

6.1.2.3 Prerequisites of the methodology

The methodology assumes the use of a system description method from systems engineering such as the hierarchical decomposition of a system in functional blocks or in building blocks. The following figure shows an example of the hierarchical

[10] RF 0015 Reference Document For The Certification Of The Safety Integrity Level Of Products Or Systems According To EN 50126, EN 50128, EN 50129, EN 50657, ISO 26262 and IEC EN 61508 standards, Certifer 2019,

breakdown of a system into building blocks down to the basic building blocks which are considered as elementary building blocks.

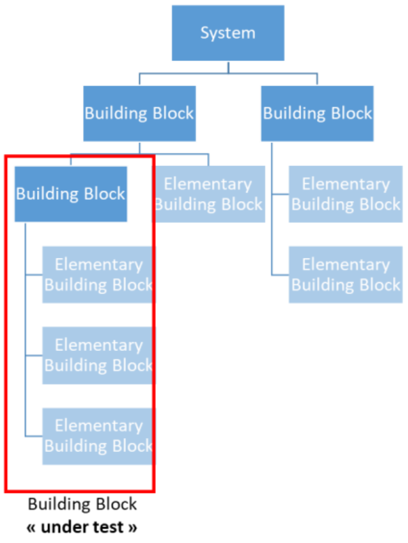
System
Building Block
Building Block
Building Block
Elementary Building Block
Elementary Building Block
Elementary Building Block
Elementary Building Block
Elementary Building Block
Elementary Building Block
Building Block
« under test »

Figure 16 – Hierarchical breakdown of a system

6.1.2.4 Best practices for the FRACTAL project

Some best practices can be anticipated for safety-related building blocks and sub-systems and then reused during the actual product certification. In the following, we provide a short set of recommendations and best practices to anticipate future safety certification that can be applied in the project:

- Sort safety-critical UC requirements into three categories:
  - Functional requirements
  - Non-functional requirements
  - Safety requirements
- Capture these requirements in a deliverable related to the use case.
- Test safety requirements in WP7/8.
  - Capture the results
- Full traceability among the UC specification, development, test, verification and validation process

6.1.2.5 Safety by construction

While the design is intended to be built safe by construction, hardware random faults cannot be avoided, and hence, appropriate safety measures are incorporated during the architectural design to guarantee that those faults are properly managed by means of either fault-tolerance features or by transitioning timely to a safe state. For instance, in the context of this document, safety features relate to (i) avoiding common cause failures (CCFs) that might lead redundant components to the same error, and hence a failure despite redundancy, (ii) limiting and monitoring time overruns due to multicore interference to preserve freedom from interference, (iii) reliable (fault-tolerant) communication, and (iv) allowing the use of AI-based components by resorting to appropriate ASIL (short of Automotive Safety Integrity Level) decompositions relieving AI-components from inheriting safety requirements.

6.1.2.6 A simplified safety related methodology for the FRACTAL project

1) Implementation of a hierarchical traceability system

2) Breakdown of the system into building blocks

3) Identification of the elementary building blocks

4) Hierarchical specification of the use case (HW/SW)

- Specification of the functional requirements
- Specification of the non-functional requirements

5) Hazard analysis / feared events analysis

6) Safety requirements specification

7) Identification of relevant buildings that will be under test during the project

- Select the buildings blocks that require an increased confidence level and a reduced number of systematic faults

8) Isolation/extraction of specifications related to the building blocks "under test"

9) Specification & realization & documentation of functional tests

- Functional tests of the elementary blocks
- Functional tests of software integration
- Functional tests of HW/SW integration

10) Specification & realization & documentation of non-functional tests

11) Specification & realization & documentation of the fault injection tests

- This goes beyond the tests usually used in non safety-critical domains.
- Faults are injected on the inputs of the functional blocks and shall not propagate to the outputs.
- Fault injection is a time-consuming practice and the choice of the hierarchical breakdown where to apply it is crucial.
- Fault injection can detect implementation errors, dysfunctional architecture and also incomplete or erroneous specifications.
- See Figure 17 below

12) Functional validation coverage of building blocks under test

- Verification of the traceability between the building blocks specification and the tests specification

- Verification of the adequacy of the hierarchical specification, the tests specification and the obtained test results
- Identify what has been validated
- Determine if the building blocks meet the requirements of the building block specification
- Analysis and identification of malfunctions and gaps

13) Technical recommendations regarding the "buildings blocks under test" for future projects and towards a future product

- What has been validated in the FRACTAL project
- What doesn't work or may cause problems
- Desirable future improvements
- What remains to be validated

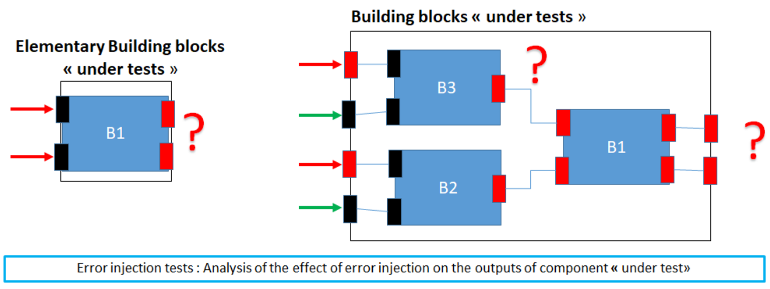
Elementary Building blocks « under tests »
B1
?
Building blocks « under tests »
B3
?
B1
?
B2
Error injection tests : Analysis of the effect of error injection on the outputs of component « under test»

Figure 17 – Illustration of fault simulation: error propagation

6.1.2.7 Reusing building blocks

In the context of FRACTAL, products targeting safety-related application may rely on pre-existing building blocks. The main reason for this is that the cost of development from scratch can be prohibitive in relatively complex systems. For instance, creating a complex safe and secure system with capabilities comparable to mainstream computing systems (Microsoft Windows or GNU/Linux) would incur in prohibitive costs – estimated at 50 B$ for the Linux kernel[11].

Safety standards allow the reutilization building blocks. For instance, IEC 61508 allows using a proven-in-use argument (named Route 2S in IEC 61508-3). However, to achieve certification based on this argument, the product developer needs to provide a vast amount of detailed information of collected historic data (e.g. in IEC 61508-7, C.2.10.1) that sometimes is not available.

However, there are other means to achieve qualification of pre-existing software elements. For instance, the SIL2LinuxMP[12] project provides the safety qualification

[11] Nicholas Mc Guire and Carles Hernandez, An open dependable platform for safety critical systems, Hipeac Magazine April 2020, https://www.hipeac.net/magazine/7154/

[12] https://sil2.osadl.org/

argument for the pre-existing software elements of a constrained Linux environment using IEC 61508, Route 3S. For that, one has to provide arguments explaining why the development process of those pre-existing software elements satisfies the high standards of IEC 61508[13].

Unfortunately, in the context of close-source libraries (e.g CudaDNN) certification cannot be achieved by the application developer unless the owners of these libraries go through an out-of-context (an element in isolation) certification process and/or adapt their libraries to fit specific standards and are able to provide the required safety documentation. While in theory, the usage of close-source libraries not developed in conformance with safety standards is not actually precluded by certification standards (e.g end-users can use black-box testing), this however, has severe implications for applicability. Thus, for these HW/SW products to be qualifiable for safety-related applications, suppliers should either adapt their products to comply with specific safety standard or allow end-user to go for alternative open-source libraries. For the latter, in order to make this approach attractive, open-source libraries must provide competitive performance with respect to the existing closed-source libraries[14].

6.2 Focus on some FRACTAL building blocks

To build the safety concept for a fractal system, a number of safety-related properties need to be built bottom up. Generally, these properties relate to the ability of the system and its components to detect, manage and/or tolerate faults and or behavior beyond the intended functionality. This section covers those aspects for the different components in the form of safety measures and strategies. In particular, approaches are presented to deal with faults for the computation of an application (diverse redundancy), timing interference across applications, communication across nodes, and AI-related processes.

6.2.1 Diverse redundancy

Fault detection in a fractal system can be efficiently provided bottom-up. This implies that appropriate safety measures need to be deployed along with computation means. Safety measures for the highest criticality functionalities (e.g. ASIL C and D) include some form of diverse redundancy so that a single fault, despite affecting all redundant elements, cannot lead to exactly the same error, which might escape detection.

[13] Andreas Platschek, Nicholas Mc Guire, Lukas Bulwahn, Certifying Linux: Lessons Learned in Three Years of SIL2LinuxMP, Embedded World 2018.

[14] H. Tabani, et. al., Assessing the Adherence of an Industrial Autonomous Driving Framework to ISO 26262 Software Guidelines, Design Automation Conference 2019.

CCFs, in automotive terminology, are those failures caused by a single fault that makes safety measures, such as redundancy, ineffective. For instance, two identical cores executing the same task redundantly fully synchronized have the same state, and upon a common fault (e.g. a voltage droop) could experience the same error. To avoid CCFs, safety-related systems implement redundancy with some form of diversity so that the risk of experiencing identical errors in redundant elements is residual. In the case of storage, this is usually achieved using Error Detection Codes, in the case of communications using Cyclic Redundancy Check codes, and in the case of computation, using some form of lockstepped execution where two or more identical cores execute identical software but with some staggering (i.e. time shift) so that cores' state is sufficiently diverse.

Therefore, platforms intended to run functionalities with high integrity requirements need some form of lockstepping support. This can be achieved with tight lockstepping, where only one redundant core is visible at software level and the others can only work in lockstep mode, as done, for instance, by the Infineon AURIX microcontrollers for the automotive domain.

Tight lockstepping at core level can be implemented with different flavors. For instance, one could compare the outcome of each instruction or even each pipeline stage every cycle. However, the most effective solution has been shown to compare only off-core activity (e.g. requests visible in the interconnect) to reduce the overheads while avoiding any visible impact due to errors.

While this solution is highly effective to attain diverse redundancy, it is inflexible since it does not allow using the cores independently to run different tasks. An alternative is using light lockstepping, where redundancy is created and managed at software level, and independent cores are used enforcing staggering with SW-only means, as illustrated in Figure 18.

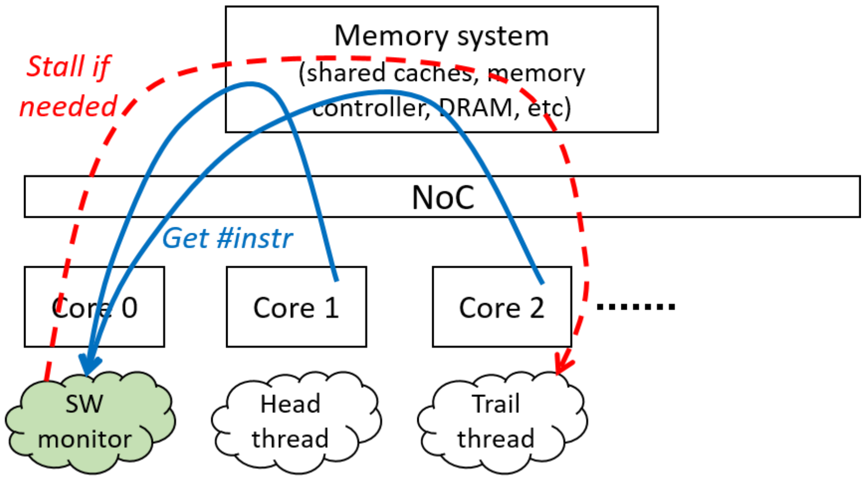

Figure 18 – Schematic of the SW-only lightweight lockstep.

SW-only diverse redundancy builds upon the creation of redundant processes at software level, so that both of them receive the same – redundant – input data and return their results for comparison in a *safe* CPU. Without loss of generality, this discussion focuses on dual-core lockstep, which is the common approach in many domains, including automotive.

The methodology builds on the use of three threads (see Figure 18):

- **Monitor**. The monitor is the one spawning the redundant computation threads (head and trail), and monitoring and enforcing staggering between them.
- **Head thread**. The head thread executes the functionality without any specific control for the sake of achieving diverse redundancy.
- **Trail thread**. The trail thread executes the functionality redundantly with some staggering (delay) w.r.t. the head thread. Therefore, if the staggering at any point is too short, it is stalled for a while.

Since the monitor lacks any form of redundancy, it needs to run in a native lockstepped core, which may be in a separate microcontroller, or may also be in the same microcontroller. The monitor spawns the head and trail threads into two other cores, which do not implement tight lockstepping support. Then, the monitor performs the following steps periodically every **`Tcheck`** cycles:

1. Collects the instructions executed count (**`IC`**) from both threads, so **`IChead`** and **`ICtrail`**.
2. Computes the difference between both counters and compares it against a threshold **`Istagger`**.
    a. If the head thread is sufficiently ahead from the trail one, then both threads continue the execution. Formally, execution continues normally if (**`IChead - ICtrail) > Istagger`**.
    b. Else, if the trail thread is too close to the head one, then the monitor stalls the trail thread during the next monitoring period (so **`Tcheck`** cycles). Formally, the trail thread is stalled if (**`IChead - ICtrail) <= Istagger`**.
3. Finally, the monitor sleeps until the remaining time until elapsing **`Tcheck`** cycles.

This mechanism guarantees that the trail thread cannot catch up with the head thread as long as the instruction threshold **`Istagger`** is large enough so that, even if the head thread stalls completely and the trail thread executes at the maximum possible speed, the trail thread cannot catch up with the head thread since one monitoring period until the next one. This implies that executing a **`Istagger`** instructions must require at least **`Tcheck`** plus the time to detect that the current staggering is too low and stalling the trail thread.

6.2.2 Management of timing interference

Safety-related systems, as part of their development process, must adhere to specific verification and validation (V&V) processes, and include safety measures to deal with random hardware faults that may occur in the field. This is achieved by means of observability and controllability knobs. Those knobs have already been deployed in mono-core SoCs. However, MPSoCs pose a number of challenges related to performance interference in the hardware shared resources across tasks running in different cores. Such interference must be properly accounted for and mitigated to meet performance-related safety requirements in line with safety standards specifications. In the context of FRACTAL, since such interference emanates at node level, that is the scope where safety measures and V&V means need to be deployed.

Commercial safety-related MPSoCs, such as, for instance, the Infineon AURIX processor family, include limited features to monitor and mitigate multicore timing interference. Typically, those MPSoCs include some form of Statistics Unit (SU) capable of tracking access counts, number of instructions executed of different types, as well as aggregated stall cycles in some buffers and queues. Those MPSoCs may also include powerful debug facilities such as Aurora, Lauterbach and CodeWarrior, which allows collecting detailed information about events in the MPSoC. Unfortunately, those SUs and debug facilities have some limitations:

- It is generally hard – if at all possible – discriminating how much interference each core creates on each other core since stall cycles, if available, are provided in an aggregated manner.
- The MPSoC may lack means to exercise control over multicore timing interference.
- Despite debug facilities are powerful, they are normally only usable during the development process given the fact that they require specific equipment and software for their use. Hence, safety measures to be used during operation cannot exploit those facilities.

Specific SUs can be tailored to deal with those challenges and meet the requirements of safety-related systems in terms of multicore timing interference, both during V&V and during operation. In particular, a timing-interference aware SU would allow implementing appropriate safety measures if it provides the following features:

1. Interference monitoring. Monitoring the amount of delay experienced by each task due to interference, and being able to identify the particular task causing such interference is particularly relevant to optimize the system during design and verification, and to diagnose deadline overruns in the field.
2. Interference quota enforcing. Those deadline overruns can be simply avoided if means are deployed to set quotas on how much interference each task can create on each other task.

Those features allow implementing both, reactive mechanisms and diagnostics by monitoring interference, as well as proactive mechanisms based on quotas to avoid effects due to undesired timing behavior. Overall, those mechanisms are the basis to meeting freedom from interference, in line with the requirements of functional safety standards (e.g., ISO26262).

Timing interference generally manifests in the on-chip interconnection networks since all data sent to/from the cores needs to traverse those networks. Hence, SUs for timing interference monitoring and control must likely be deployed along with those interconnection networks. Since those interconnects use standard interfaces, such as AMBA AHB/ACE/AXI, SUs may be easier to integrate and reuse if they comply with those protocols.

In the context of FRACTAL, if the MPSoC includes a centralized interconnect managing all on-chip traffic from cores (e.g., a bus), a single SU may suffice. Alternatively, if distributed interconnects are deployed (e.g. NoCs) so that no single location drives all core-related traffic, we may need to set up multiple SUs to be able to monitor and control all on-chip traffic. Those SUs may further require some post-processing of the information collected in a distributed manner to gain global knowledge about how multicore timing interference is occurring.

6.2.3 Reliable communication

Fractal systems will rely on inter-node and intra-node communications. To satisfy safety requirements, on-chip communication networks or interconnects may rely on state of the practice protection mechanisms such as Error Correction Codes (ECC), replication or monitoring.

<u>ECC.</u> Is usually required for data links. The choice for the actual ECC implementation, (e.g. SECDED or parity) depends on the implementation costs and the actual safety need. For instance, fail-safe applications or applications with long fault tolerance time intervals can use simple parity bits while more stringent applications requiring fail-operational capabilities may require also redundancy bits with correction capabilities such as SECDED (single error correction double error detection).

<u>Hardware replication.</u> Hardware replication (e.g. duplication) might be also needed in addition to ECC to achieve the highest integrity levels.

<u>Hardware monitors.</u> Shall be integrated with the interconnect to allow built-in self-test (BIST) and error reporting capabilities. This is usually a requirement that expands all integrity levels and it is aimed to ensure safe operation during the lifetime of an application. In the context of highly complex SoC platforms (e.g multicores) these monitors should also have the capability to track software timing information.

For off-chip communications, applications safety shall not rely on FRACTAL communications. That is, the node needs to remain safe (i.e transition to a safe state) if the communication with other nodes is down. However, data integrity and authenticity conveyed by the FRACTAL communication are secure features that are also important for safety, not to make any decision based on flawed information.

6.2.4 Artificial intelligence

The utilization of artificial intelligence (AI) techniques in the context of FRACTAL applications with safety requirements will be constrained. In general, the utilization of AI will not involve tasks of the system mapped to a certain criticality level. For instance, following ISO26262 nomenclature AI tasks will be restricted to QM (quality management) functionalities only. That is, functions without safety relevance and that only require standard Quality Management processes. Otherwise, safety violations in absence of HW or SW failures that are a consequence of limitations in the algorithms, sensors or actuators must be also considered as defined in the ISO/DIS 21448: Safety of Intended Functionality (SOTIF) automotive sector standard.

Even in this constrained scenario, the deployment of AI techniques in platforms that involve critical functionalities (e.g mixed criticality systems) is challenging. The reason is that sharing the same CPU between critical and non-critical tasks requires ensuring the existence of partitioning mechanisms (provided by Linux or a hypervisor) to allow AI related tasks not to interfere with critical tasks (neither at the functional nor at the temporal level). Additionally, if FRACTAL platforms make use of AI outputs to improve non-safety properties like quality of service (QoS), energy or availability, the AI system shall provide a measure of uncertainty in all the decisions in order to get an estimate of the QoS that these systems based on AI will be able to provide.

In summary, for FRACTAL systems, the safety of the platform should not depend on the output of the artificial intelligence algorithms. Involving AI in the decisions that may affect the safety of an application is however mandatory to achieve fully autonomous certified applications. Unfortunately, this technology, while being currently a hot research topic, is not expected to be available in the near future for critical industrial domains.

6.3 Certification of VERSAL platform and related applications

6.3.1 Versal ACAP functional safety origin

Functional safety of the Xilinx Versal ACAP device families are based on the concepts that are already successfully deployed in Zynq Ultrascale+ MPSoC and inherit the specific core certifications as applies. Due to short time since introduction the qualification and publishing of all the features has not been achieved up to now. The chapter 6.3 aligns the available information at with the proposed methodology for the Fractal project forwarded in 6.1.2.

For more accurate and up to date information PLC2 and related partners will track the vendor's Xilinx Functional Safety Lounge. This collection (available under NDA through https://www.xilinx.com/products/technology/functional-safety.html) provides safety related topics as:

- Certified Hardware and Software Design Tools
- Functional Safety Certificate and Reports

- Functional Safety Package
- Certified Methodologies
- Certified IPs
- Reliability Reports
- Xilinx Certification Papers

This Xilinx Functional Safety Lounge provides such information for all device families and particular information on Versal ACAP may not be available at this time.

The proceeding as described here is to apply the device specific information to a precisely defined platform setup (forwarded in 7.4) that is held against certification requirements for specifically IEC 61508 and ISO26262. To achieve this, the safety related items are listed with relation to safety impact, guiding how to apply this along said methodology.

6.3.2 Versal ACAP safety features

Versal ACAP has been developed with supporting the Functional Safety Standards IEC61508, ISO26262, SIL 3 in IEC61508, ASIL D in ISO26262 using decomposition while customers can use further artifacts from IEC61508 and ISO26262.

For the physical device there exists 3 isolated domains in Versal

- FPD (Full power domain)
- LPD (low power domain)
- PL Domain (programmable logic)

And domains in Versal which need to be used in a shared context:

- PMC (Platform management controller)
- DDR (DDRAM memory controller)
- NoC (Network on Chip)

The Processing system provides CPU based platforms with

- Cortex-A72 in the FPD (dual core)
- Cortex-R5 in the LPD (dual core)
- PMC: (hardened Xilinx MicroBlaze processors) in the LPD for boot and platform management

The Versal architecture is isolated in the groups as described but the sharing concept allows a global inter-communication capability. This also requires customer use-case dependent isolation where protection mechanisms are required.

Also, the Versal Power Domains (supply voltages) provide an isolation concept with some of major blocks as listed here:

• PS FPD	(APU, CCI)
• PS LPD	(RPU, OCM, dedicated Peripherals)

• NoC	(Network on Chip for memory and streaming interfacing)
• PL+AIE+CPM	(programmable hardware, programable engines and i/o)
• PMC	(platform management: boot, power, initialization)
• Battery	(RTC, BBRAM)

6.3.2.1 Versal Safety Components

The first safety component group is the LPD which includes the Cortex-R5 processors and the platform management control for initialization purpose:

- ARM Cortex R5 Cluster (2 R5 CPU's, TCM 128KB/Core, 32KB I/D Cache)
- Split mode: independent R5 cores, using an independent OS
- Lockstep mode: one OS is running on two cores
- Global Interrupt Controller (GIC) for the RPU
- On Chip Memory (OCM) 256KB can be used in addition with TCM
- Dedicated I/O: Gbe, CAN FD, SPI, I2C, GPIO, UART, WDT, TTC
- Processing System Manager (PSM), controls PS power Islands
- Accelerator coherence port Direct Memory Access controller (ADMA)
- Network on Chip (NoC) Port for DDRAM access
- Xilinx Memory Protection Unit (XMPU), customizable isolation

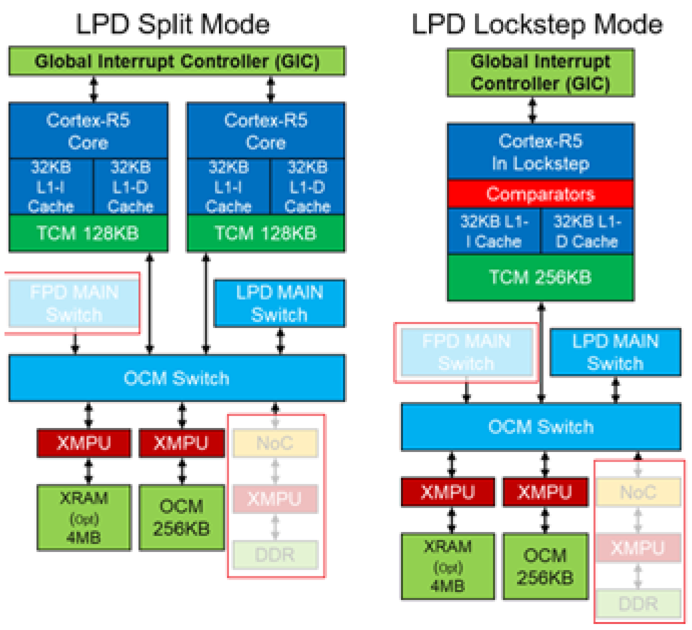

Figure 19 – Low Power Domain Components

For random safety SIL 3 can be achieved for the LPD domain and for systematic safety SC 3 / ASIL D can be achieved for this domain with appropriate requirements of initialization and run-time protection. When using shared resources like PL or AIE accelerators these needs to be developed in responsibility of the user defined criteria as they exist outside of the LPD.

The second safety component group is the FPD which includes the Cortex-A72 processors, Cache coherency management and the NoC:

- ARM Cortex A72 Cluster (2 A72 CPU's, 32KB I/D L1 Cache, 1MB L2 Cache)
- Neon + FPU
- Cache Coherent Interconnect (CCI)
- System Memory Management Unit (SMMU)
- Windowed Watchdog Timers
- Network on Chip (NoC) Port

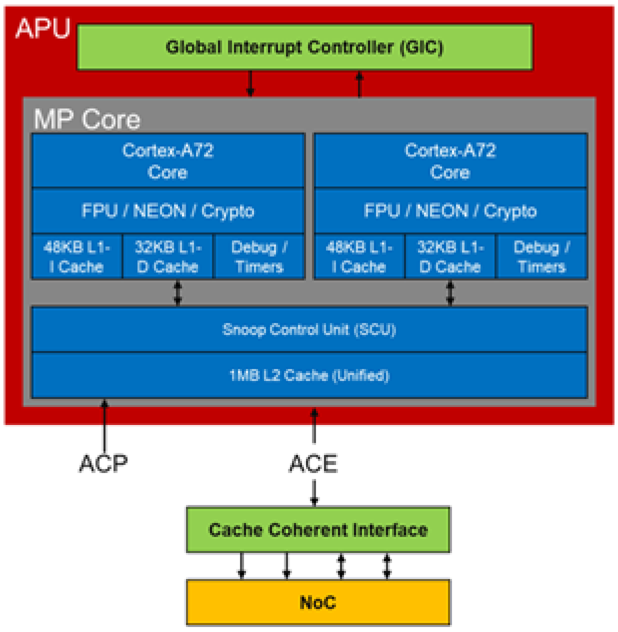

Figure 20 – Full Power Domain Components

For random safety the target is to support SIL 2* / ASIL B* for the FPD domain and for systematic safety SC 3 / ASIL D can be achieved for this domain with appropriate requirements of initialization and run-time protection.

*) actual in the certification process at Xilinx in Q2/2021

For systematic safety SC 3 / ASIL D can also be achieved for the PL domain modules programmed in the PL. For random safety the target is to support SIL 2 / ASIL B for the DDRAM controller part and for systematic safety SC 3 / ASIL D can be achieved. For isolation purpose protection units (XMPU) exists for the specific target ports.

6.3.2.2 Versal Functional Isolation Techniques

The Versal device includes several protection units of two types:

- XPPU (Xilinx peripheral protection unit)
- XMPU (Xilinx memory protection unit)

The XPPU provides peripheral protection for

- PMC APB programming Interface
- NPI - NoC Programming Interface
- LPD APB, AXI Programming Interface
- FPD AXI Programming Interface
- LPD Peripherals
- PMC Peripherals

The XMPU provides memory protection for the memory access of

- DDRAM
- XRAM (Accelerator RAM)
- OCM
- PCM RAM

and can differentiate up to 16 regions of an address range for each XMPU.

Also, a trusted isolation resource is available with ARM Trustzone when using the Linux OS running on the Cortex-A72 while the isolation is differentiated in secure and non-secure accesses where the trusted firmware is provided with the Xilinx OS-Linux build flow.

A Memory protection also exists with MMUs: The Cortex-A72 cores includes a MMU and a System-MMU is integrated in the FPD which is required for hypervisor OS management, if needed.

6.3.2.3 Versal Safety Channel Architecture

A Functional Safety Channel is a set of hardware and software resources that implements the entire Safety Function. A Functional Safety Channel has a Sensor, Logic Solver and an Actuator. A centric safety channel can be defined in the LPD with the RPU processing and the R5 can be initialized to run in the lock-step or split mode for the Solver task. Sensor Inputs may be sourced from peripherals in the LPD or PL and Actuator Outputs may be sent to peripherals in the LPD or PL.

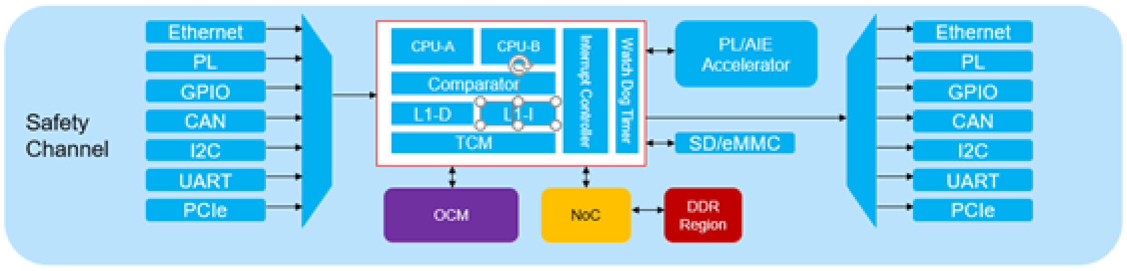

Figure 21 – LPD Centric Safety Channel

A centric safety channel can be defined in the FPD with the APU processing using the Cortex-A72 running a SMP OS like Linux or AMP like Baremetal or FreeRTOS with a logic Solver task. Sensor Inputs may be sourced from peripherals in the LPD or PL and Actuator Outputs may be sent to peripherals in the LPD or PL.

The FPD centric safety channel certification is still under evaluation at Xilinx (Q2/2021).

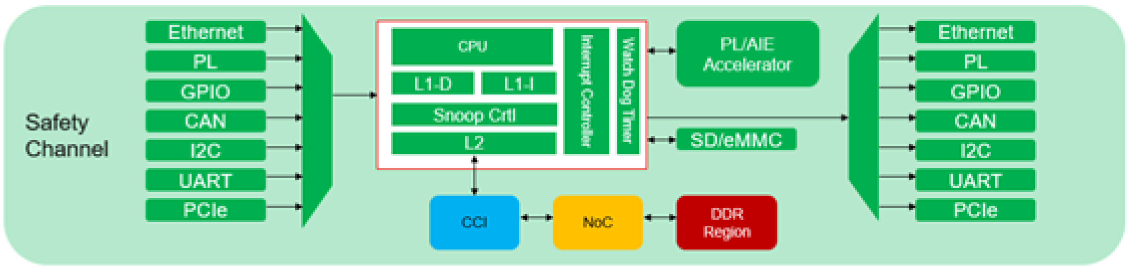

Figure 22 – FPD Centric Safety Channel

A centric safety channel can be defined in the PL with a MicroBlaze soft-IP processor implementation in the PL running a Baremetal or FreeRTOS with a logic Solver task. Also, a hardware redundant core implementation is available for the MicroBlaze system. Again, sensor Inputs may be sourced from peripherals in the LPD or PL and Actuator Outputs may be sent to peripherals in the LPD or PL.

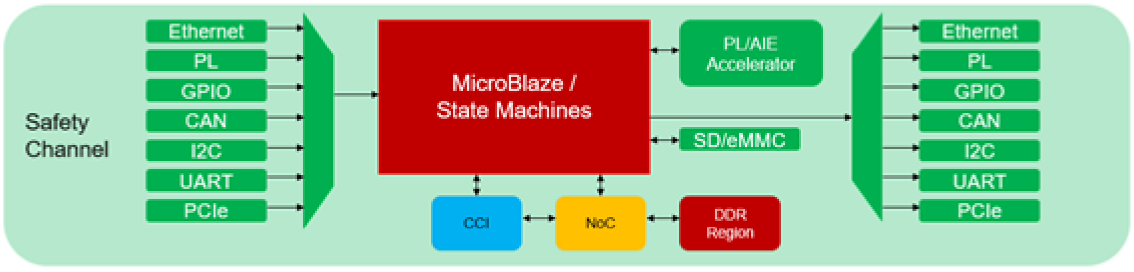

Figure 23 – PL Centric Safety Channel

Multiple domain safety channels architectures can exist with multiple safety channel systems as described above with LPD, FPD and PL based solvers.

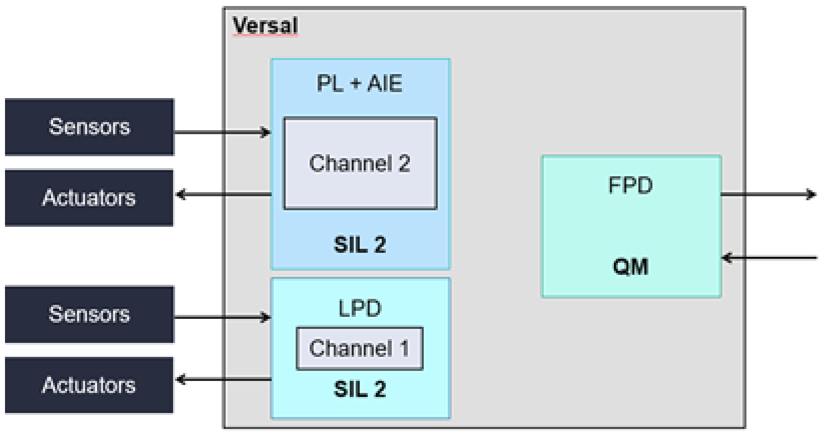

Figure 24 – Deample: A Dual Safety Channel in Versal

Also, heterogeneous safety channels solvers can be created with solver architectures of co-processing i.e., APU + RPU coprocessing or using the AIE as accelerators via NoC. A barrier between multiple safety channels can exist physically like a FPD / LPD separation or can exist in temporal diversity which is available when running two independent OS scheduled with a hypervisor running on the APU in the FPD.

6.3.2.4 Versal Diagnostics

Diagnostics are required for functional safety systems and are used to detect a fault on Functional Safety. A safety requirement is guaranteed based on the diagnostics. The diagnostics implementation is available for the Versal in hardware and software:

- Internal Hardware (intrinsic to the element)
- Packaged Software (embedded test libraries, test applications)
- Architecture (External Redundancy)

The LPD Diagnostics provide the following services:

- Lockstep CPU (Cortex-R5)
- Error Checking Code (ECC) for TCM, OCM, Caches, DDRAM
- Windowed Watchdog Timers
- Temperature & Voltage Monitoring Satellites
- Bus switch Timeout, Parity, Port Isolation
- Protection Units (XPPU, XMPU)
- Software Test Library
- Check Register State (Parameters)
- Check Peripheral function
- Check the checkers (Fault injection)
- End to End Data Integrity (a.k.a Black Channel) – Customer Driven

The FPD Diagnostics provide the following services:

- Hypervisor capable CPU (Cortex-A72)
- Error Checking Code (ECC) for OCM, Caches, DDRAM
- Watchdog timer
- Temperature Monitoring Satellite
- Software test libraries
- Check Register State (Parameters)
- Check the Checkers (Fault injection)
- End to End Data Integrity (a.k.a Black Channel) – Customer Driven

The AIE Diagnostics provide the following services:

- MBIST (for program and data memory)
- ECC for program memory and data memory
- TMR (triple mode redundancy) for critical system registers

Versal test libraries (STLs) are provided for safety requirements with ASIL C for OS running in the LPD using the Cortex-R5 in the RPU or a MicroBlaze in the platform management (PMC). When using the STL test libraries running on the AIE engines the safety standard ASIL B can be achieved. Versal STLs support for the processing units

1. RPU (R5 core)
2. PSM (MicroBlaze), Xilinx provided firmware
3. PMC (MicroBlaze),
4. AIE (core functional tests including its interfaces)

6.3.2.5 Versal Tool Chain

Xilinx provides Vivado for the hardware design flow and Vitis for the software design flow. Vivado has been certified from the TÜV Süd in an earlier release and is expecting a certification for Functional Safety Applications according to the standards ISO26262 & IEC61508 in this year. The Vitis environment provides the Xilinx toolchains for the embedded flow and acceleration flow mainly based on C/C++ language programming.

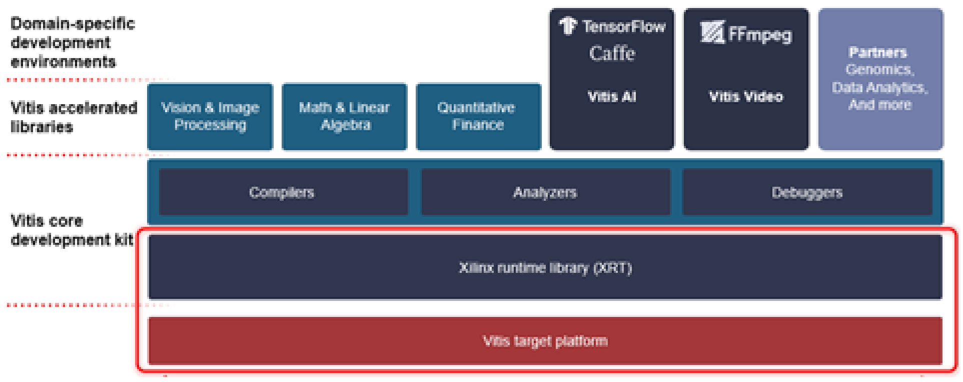
Domain-specific development environments
Vitis accelerated libraries
Vitis core development kit
Vision & Image Processing
Math & Linear Algebra
Quantitative Finance
TensorFlow
Caffe
Vitis AI
FFmpeg
Vitis Video
Partners Genomics, Data Analytics, And more
Compilers
Analyzers
Debuggers
Xilinx runtime library (XRT)
Vitis target platform

Figure 25 – Deample: Vitis Unified Software Platform

The Vitis release 2020.2 is the first release where the certification process is ongoing for specific firmware, like the platform management. The tool NoC Compiler is required to define separation of memory and port accesses and so also for the isolation methodology in hardware. The NoC compiler is part of the Vivado toolchain. This static compilation provides the NoC initialization at boot time where properties can be assigned like defining exclusive groups or a separate hardware routing.

FMEDA Tools (Failure Modes, Effects and Diagnostic Analysis)

1. FuSa metric computation
2. FMEDA data integrity checks
3. FMEDA in certification process at Xilinx => FIT rates => safety manuals
4. FMEDA tools at the development site (optional)

There are various tool methods that can be deployed to fulfill the proposed Methodology for the Fractal project so a definition of the actual scope must be created.

6.3.3 Versal ACAP Certification in Fractal

With all the safety related features that are available, there are typical FMEDA Workflow phases that are described in the Xilinx ecosystem. The task at hand is to apply this to the proposed methodology in section 6.1.2. To show that this process can be followed with sound coverage should enable a certification of a Fractal node design.

The proposed concept towards a potentially certifiable platform for FRACTAL use-cases would be a statically hardware partitioned architecture that will support a separation on safety channels fulfilling different classes of safety. Such setup will be forwarded in section 7.4 as to explicitly describe the scope of the features to be used. This is subject to coordination with the use case requirements.

7 Methods for the integration on FRACTAL platforms

This chapter introduces methods that will be used to integrate use cases on FRACTAL platforms in WP7 and WP8. Some sections are specific to a use case.

WP7 will integrate the FRACTAL building blocks, technologies and methods. The performance, safety and security requirements will guide the integration and verification activities. The verification task will assess the metrics regarding these quality attributes and set up a coordination task to gather and analyze the insights and KPIs of the various FRACTAL nodes.

WP8 will aim at homogenizing the requirements from use case providers in order to provide a unique framework for dealing with all the real-world situations considered in the use cases in term of Fractality. The objective of this work package is to define the needs of the use cases by identifying domain-specific requirements and will assess at the end of the project whether the technical objectives of the project have been reached.

Moreover, WP8 will ensure that developments in the field of:

- Artificial Intelligence and Autonomous Decisions
- Safety and security insurance
- Low power and high performance on the edge implementation

To do this, a coordination process between use case providers, technology providers and integrators will be needed in order to ensure that the development activities are in the direction of fulfilling the pillars of the project.

7.1 Operational integration

To be able to bring FRACTAL to market, FRACTAL must be viewed as a Product and this project must define the characteristics of that product and how it is built and delivered. Project workflows described in section 4 introduce the many components that the distinct partners are developing to be integrated into the FRACTAL platform and that UCs will use; thus defining the elements required in a FRACTAL node, both software and hardware.

On the other side, the distinct use cases provide a focus on the variability that a FRACTAL node must support. By the end of the project, a Fractal node should be able to be constructed to satisfy each use case (WPs 7 and 8). Operational Integration refers to the process that must be undertaken to build the FRACTAL node that an end user wants. That process includes putting the Fractal node built in a production environment.

However, first a common view of what a Fractal node is and which are its parts. Here Feature Models can help to define what a FRACTAL node will be. A feature is defined as a distinguishing characteristic of a product, usually visible to the customer or user of that product. The analogy of a car will be used to explain what a product and a feature model is. When we want to buy a car, we have to select among the distinct features available. For instance, the user can choose the version, the engine, the color, the air conditioning system, the transmission, etc. Sometimes some selections force the selection of another feature. For instance, a given product version can force a given engine and restrict the number of other options available. This is, features can have distinct values and also have relationships. Even, some features can be optional (e.g. sunroof).

The diagram in Figure 26 shows a starting point for the FRACTAL feature model.

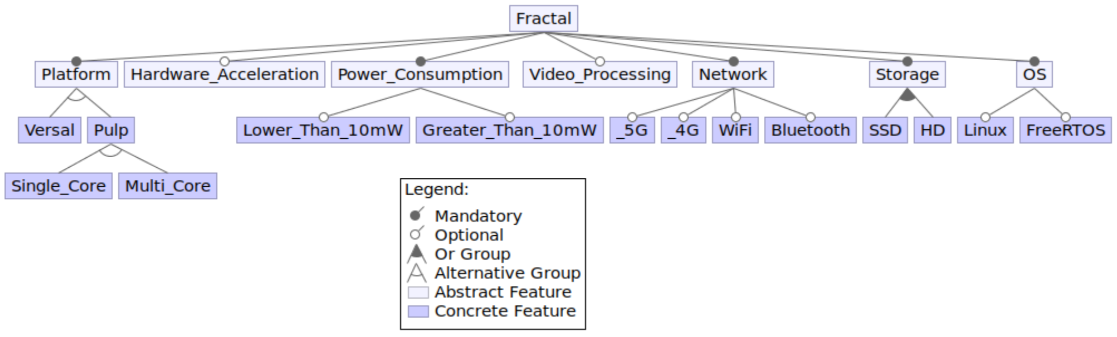

Figure 26 – Sample initial feature model for Fractal

Notice that the selection of a feature implies the selection of the corresponding product components underneath. It this sense, the selection of a FRACTAL feature can force the selection of a given SW/HW component or the exclusion of another. For instance, a feature of Low Power consumption may exclude the Versal Platform (feature constraints).

In order to build this model, every component (HW and SW) being developed in the context of the project has to specify what need it does solve and what feature it adds to Fractal. Features will be organized in a tree like the one presented before.

A "feature catalogue" contains all the available feature options for all the products. The selections that a user performs to build a product conforms the "Bill-of-Features" for that product. Given a valid "Bill-of-Features", the process to build that FRACTAL node should be detailed.

In the context of the project each Use Case should be able to express their needs for the Fractal Node through the Feature Model. According to the workflow of the project presented in section 4, WP3 defines the node architecture over which the other WPs will build their contributions. Thus, on WP3 an initial feature model will be constructed, that will be refined with the participation of the Use Cases on WP 7 and 8 (following subsections detail the methodologies on these WPs). Then the construction process for the selected Fractal node should be defined. This is, questions like the following ones must be answered:

- Which platform should be bought
- How are required HW components installed (indirectly selected on the feature model)
- How is OS installed
- How are required SW components installed (indirectly selected on the feature model)
- How to add software for the use case
- How to test that the system is operational
- How to pass certification tests
- …

In the end, the industrialization and productization of Fractal are achieved, enabling companies to sell products that conform to Fractal for other use cases.

7.2 SW integration (PULP, VERSAL), RTOS for PULP

7.2.1 Posix compatible RTOS integration to the PULP

Background:

In FRACTAL the general (low risk -- fast to deliver) approach is to integrate the middleware layers (by WP4 and WP5) top of the Linux operating system. For use-cases this will offer a simple and powerful system to implement the application logic. Also, the acceleration/protection functions that physical HW offers are easy to integrate to the Linux low-level layers. The drawback is that Linux requires adequate amount of processing power and energy. Due high-power nature Versal platform this is not issue as such, but in low-end nature of PULP it limits the options.

On this task a Posix compatible OS (Nuttx) is integrated to PULP. From software point of view, will be limited – less memory, no supervisor abstractions, it lacks high-level languages such as Java and Python. However, unlike most of the Realtime operating systems (RTOS) it has Posix threads and sockets, standard c-api's and standard dev-tools. Most of the cases source-code developed to the Linux can just be compiled in.

As a result, the Nuttx will be a compatible Fractal subset. In its scope, it does not require any special porting for the Fractal SW.

As a result, this offers a mechanism to push certain Factual applications to the extreme low-power -- low-cost targets.

Implementation strategy:

The Nuttx integration work will be implemented as a spiral model. At first a minimum possible implementation will be developed. Result will be released and based on available options and Fractal requirements the next development round will be implemented. Results of the first few rounds will be more demonstrative, but on later rounds more practical results will be delivered.

First target: Select a (Pulp) RISC-V platform and integrate Nuttx to it.

There exists a deprecated implementation of Nuttx for PULP. Official Nuttx has dropped that out due limited platform support. As a result, we study of how much the deprecated port is out of Nuttx baseline and what are the options to release the result.

While the availability of Fractal Pulp HW is not clear at this stage, some similar/existing/available (Pulp) RISC-V platform will be used for first round. According studies the development board of the Greenwaves GAP8[15], would be easiest to obtain for the first platform.

Second target: Select either: Another HW Platform or some middleware layer(s) or another Nuttx feature.

After each target new target is set.

N-1 target: Branch a version for the task 7.2.2.

N target: When adequate amounts of targets integrated, integrate (some) Use Case to platform.

Due the uncertainty of the requirements that middle-ware, capability of the Fractal Pulp platforms, the applications should have a platform back-up plan -- this Fractal-Pulp-Nuttx platform will be available later that Linux based platforms. This will be a topic to collaborate with Use Cases.

7.2.2 Asymmetric Linux-Rtos multiprocessing to the FRACTAL project

In symmetric multiprocessing (SMP) all processor cores are utilized by same operating system. In asymmetric case (AMP) cores are utilized by different operating systems. There are some benefits of this sort of systems. Here are briefly the two cases:

First: The Real time matter. Linux as such is not a real-time operating system. At certain stage – that is not fixed – Linux begin to lose real time deadlines (Practical tests have demonstrated that response jitter below 1ms is intolerable). One solution to solve this issue, is the Real time version(s) of Linux. There the certain parts of kernel are modified for real time purposes. Drawback on this is that these changes are branches of the main line Linux. While the Linux evolves, the compatibility of this branch weakens and over time the new feature backporting becomes more difficult. This yields to practical problems -- typically related to security and compatibility. Another solution is to have a parallel real time operating system (RTOS). This RTOS

[15] https://greenwaves-technologies.com/gap8_gap9/

process the real time tasks, while Linux is the actual computing platform. And, having this RTOS running on same processor chip, eases the system design.

Second: The low power operation. Linux is a complex system that requires complex HW to run -- lot of moving parts that consume energy. Driving system to standby and woke up on event is a better solution, however the woke up takes in average more energy than normal operation and often the woke-up events require only little processing or even are false. Again, obvious solution is to introduce a second low-power OS that stays awake, while Linux is standby. This OS can verify the woke-up events and some cases process them. And in cases where "The Force" is needed, RTOS can woke-up Linux OS and delegate the event. Like the above case, the system design is easier, when the low-power OS and application OS are in same physical chip.

Two cases above and combination of them are valuable tools for certain application types in Fractal context. In task we seek to run Nuttx (RTOS) in one core and Linux in other cores. The Posix compatibility offers yet another potential compatibility benefit: Due both OSes are in same processor context; they may share same security, data and even binaries.

Challenges on this task are related to the context switching, security and shared resources matters below the OSes.

As in previous task the implementation will done according to the spiral model. At first an adequately good platform is selected, where some version of previous task Nuttx implementation is integrated together with some Linux. The second target will be a Fractal improved version of that. And so on.

Security features, low-power operation, application requirements need to be collaborated with other Fractal partners -- this can be also in focus of Versal platform.

This text is to be updated later during the project (presumably in D2.4).

7.3 Electronic System-Level HW/SW co-design

Systems based on heterogeneous parallel architectures (*Heterogeneous Parallel Systems* - HPSs) have been recently exploited for a wide range of application domains, especially in the *System-on-Chip* (SoC) form factor (e.g., Xilinx ZYNQ/VERSAL and Altera Cyclone V SoC families). Such systems can include several processors, memories, and a set of physical links among them. By definition, the set of processors in the same architecture is heterogeneous. This implies that it is possible to exploit, at the same time, the following processing classes[16]:

[16] Frank Vahid and Tony Givargis. 2001. Embedded System Design: A Unified Hardware/Software Introduction. John Wiley & Sons, Inc.

- *General-Purpose Processors* (GPP): x86/x64, ARM, MicroBlaze, NiosII, Leon3, etc.
- *Application-Specific Processors* (ASP): *Digital Signal Processor* (DSP), *Graphics Processing Unit* (GPU), *Network Processor* (NP), *Artificial Intelligence Processor* (AIP), etc.
- *Single-Purpose Processors* (SPP): AES encoder/decoder, JPEG encoder/decoder, UART/SPI/I2C controllers, AI engines; in general, every ad-hoc developed digital HW component (a.k.a. co-processors or accelerators).

Finally, such processors can be adopted in the form of soft, hard or fuse (i.e., *hardwired*) IP cores or as discrete integrated circuits (IC) mainly depending on the final system form factor (i.e., on-chip, on-FPGA, on-board) and scope (complete product or platform).

HPSs are often used to implement *Dedicated Systems* (DS). DSs are digital electronic systems with an application-specific HW/SW architecture. They are specifically designed to satisfy a priori known application requirements, both functional (F) and not functional (NF). A DS could be then embedded in a more complex system and/or it could be subjected to hard/soft real-time constraints. When DSs are based on HPS they are called *Dedicated Heterogeneous Parallel Systems* (D-HPS).

Apart from possible differences in terminology and composition, for this kind of systems one consideration is always true: they are so complex that the adopted *HW/SW Co-Design Methodology* plays a major role in determining the success of a product. Moreover, in order to cope with such a complexity, the selected methodology should allow the designer to start working at the so-called *Electronic System-Level* (ESL) of abstraction. This means to be able to start the design activities from an executable model of the system behavior based on a given *Model of Computation* (MoC) that would be unifying for HW and SW, and that could be described by means of a proper specification/modeling language. In fact, in the past years, a remarkable number of research works have focused on the system-level HW/SW co-design of D-HPS. In such works, the most critical issues have been always related to the *System Specification* and *Design Space Exploration* activities. In the first activity, the designer models the behavior of the desired system (specifying also possible NF requirements), the available basic HW components, and the target HW architecture. The second activity is then related to the approach, automated or not, used to find the best HW/SW partitioning and mapping for the final system implementation. The main differences among the various approaches are related to the different amounts of information and actions that are directly requested to the designer and that are so heavily influenced by his/her experience. In particular, a lot of approaches explicitly require as input the HW architecture to be considered for mapping purposes. Very few others try to fully addresses the problem of both to "automatically suggest an HW/SW partitioning of the system specification" and to "map the partitioned entities onto an automatically defined heterogeneous parallel architecture". In the context of the latest category, during other ECSEL RIA projects (e.g., MegaM@RT2, AQUAS, FITOPTIVIS, COMP4DRONES), it has been defined and improved a model-based ESL

HW/SW co-design methodology (and related prototypal toolchain), called HEPSYCODE[17], targeting heterogeneous parallel dedicated systems. HEPSYCODE reference ESL HW/SW co-design flow is briefly described in the following, together with a proposal about possible adaptation/integration opportunities of HEPSYCODE in the context of the FRACTAL project.

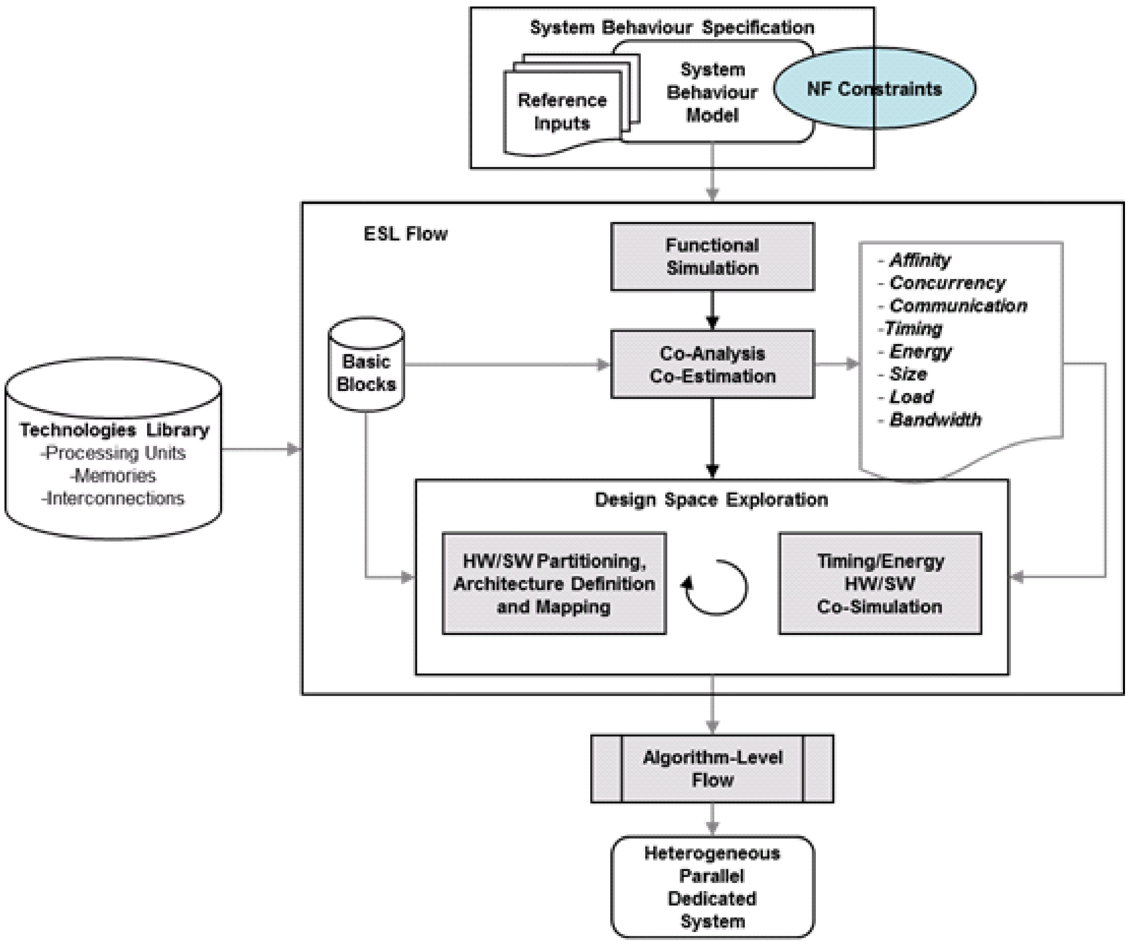

Figure 27 – The reference ESL HW/SW co-design flow

[17] http://www.hepsycode.com

7.3.1 Reference ESL HW/SW Co-Design Flow

The HEPSYCODE ESL HW/SW co-design flow is shown in Figure 27: it shows the main co-design steps and the related items that are briefly described in the following paragraphs. More details about the whole methodology can be found in [18,19,20,21].

7.3.1.1 System Behavior Specification

The entry point of the reference co-design flow is the *System Behavior Specification* (SBS). It is composed of *System Behavior Model* (SBM), *Reference Inputs* (RI), and *Non-Functional Constraints* (NFC).

SBM represents the behavior of the system to be implemented, i.e., the functional requirements. It is based on a CSP-like (*Concurrent Sequential Processes*) MoC[22,23] and described by means of the *SystemC* language[24].

RI is a set of input-output tuple pairs, possibly timed, that represents the expected outputs from the SBM when specific inputs are provided to it. RI is of critical importance since it has to be as much as possible representative of the actual operating conditions of the system (a.k.a. *Golden Inputs*).

NFC represents requirements related to aspects orthogonal to the behavior. In fact, they specify a set of constraints that have to be satisfied while still following a correct behavior. They are currently related to one or more of the following issues:

- Timing Performance Constraints
  - *Time-To-Completion constraint (TTC)*
    - Time allowed to complete the processing related to RI

[18] L. Pomante. "System-level design space exploration for dedicated heterogeneous multi-processor systems". IEEE Int. Conf. on Application-specific Systems, Architectures and Processors, 2011.

[19] L. Pomante, D. Sciuto, F. Salice, W. Fornaciari, C. Brandolese. Affinity-Driven System Design Exploration for Heterogeneous Multiprocessor SoC. IEEE Transactions on Computers, vol. 55, no. 5, May 2006.

[20] Pomante, L., Muttillo, V., Santic, M., Serri, P. SystemC-based electronic system-level design space exploration environment for dedicated heterogeneous multi-processor systems. Microprocessors and Microsystems, 72, 2020.

[21] Ciambrone, D., Muttillo, V., Pomante, L., Valente, G. HEPSIM: An ESL HW/SW co-simulator/analysis tool for heterogeneous parallel embedded systems, 2018. 7th Mediterranean Conference on Embedded Computing, MECO 2018 - Including ECYPS 2018, Proceedings.

[22] C.A.R. Hoare. Communicating sequential processes. Communications of the ACM, 21(8):666–676, August 1978.

[23] http://www.usingcsp.com

[24] SystemC, http://www.accellera.org

  - *Time-To-Reaction (TRC)*
    - Time allowed to complete the processing related to specific processes of the SBM
- Energy Consumption Constraints
  - *Energy-To-completion (ETC)*
    - Energy consumption allowed to complete the processing related to RI
- Mixed-Criticality Constraints
  - Isolation
    - Do not map processes with different criticalities on the same processor
  - Isolation with hypervisor technologies (HPV)
    - Do not map processes with different criticalities on the same HPV-based SW partition
- Architectural Constraints
  - Set of available processors and physical links
  - Min and max number of available processors and physical links instances
  - Available area (for PCB/ASIC) or an equivalent metric for FPGA
  - Reference template architecture
    - HW
      - Distributed/shared memory
      - Homo/hetero-geneous mono/multi-core processors
    - SW
      - Available (hyper)scheduling policies.

7.3.1.2 Technologies Library

In order to list and describe the basic HW elements available to automatically build the final architecture, a proper *Technologies Library* (TL) provides a characterization of available processors, memories and physical links. Such a library contains information like processing classes (i.e., actually only GPP, DSP, and SPP), operating frequencies, maximum load (for GPP and DSP classes), capacity (for SPP and memories), max bandwidth (for physical links), relative cost (considering the cost related to obtain a component and/or the effort needed to use it), and so on. Such information is then exploited during the different steps of the co-design flow.

7.3.1.3 Functional Simulation

The first step of the proposed co-design flow is the *Functional Simulation* where SBM is simulated to check its correctness with respect to RI. Such a simulation allows also to take into account timed inputs, i.e., there is a concept of simulated time, but it doesn’t consider the time needed to execute computation and communication (i.e., 0 simulated time). If SBM is not correct (i.e., wrong outputs or critical conditions such as deadlocks) it should be properly modified and simulated again. The early detection of anomalous behaviors allows the designer to correct the specification avoiding a late discovery of problems that could lead to time-consuming design loops.

7.3.1.4 Co-Analysis and Co-Estimation

This step aims at extracting as much as possible information about the system by analyzing the SBM while considering the provided TL. This step is composed of *Co-Analysis* and *Co-Estimation* activities.

During *Co-Analysis*, SBM is analyzed to evaluate three metrics: *Affinity*, *Communication* and *Concurrency*. The first one represents how much a process is suitable to be executed on a specific processor class (i.e., GPP, ASP, SPP). The second one is the evaluation of the number of bits that the different processes pairs have exchanged during the simulation. The third is related to how much concurrency has been found during the simulation in the activities of processes and channels.

Co-Estimation provides a set of estimations about *Timing*, *Energy*, *Size*, *Load* and *Bandwidth*. Timing is related to the estimation of the number of clock cycles needed, by each processor in the TL, to execute each single statement composing the SBM processes[25]. Energy is related to the estimation of the Joule consumed, by each processor in the TL, to execute each single statement composing the SBM processes[26]. Size represents the number of ROM/RAM bytes needed for SW implementations and equivalent gates (or similar metrics for FPGA) for HW ones[27]. Finally, by exploiting Timing data and considering the TTC constraint, it is also possible to estimate the Load associated with the execution of the SBM processes when mapped on a single instance of each processor in TL, and the Bandwidth needed to the different processes to communicate while fulfilling the TTC constraint. The extraction of these data from the SBM is an important step that allows, during the following design space exploration, the identification of the number and type of processors and physical links needed to satisfy the NFC.

7.3.1.5 Design Space Exploration

Finally, the reference co-design flow reaches the *Design Space Exploration* (DSE) step that is constituted of two iterative activities: *"HW/SW Partitioning, Architecture Definition and Mapping"*, and *"Timing/Energy HW/SW Co-Simulation"*. All the data (i.e., metrics and estimations) extracted in the previous steps are then used to drive the DSE by considering all the NFC. The "HW/SW Partitioning, Architecture Definition and Mapping" activity is based on a genetic algorithm that allows exploring the design space looking for feasible mapping/architecture items suitable to satisfy imposed

[25] V. Muttillo, G. Valente, L. Pomante, V. Stoico, F. D'Antonio, and F. Salice, "CC4CS: an Off-the-Shelf Unifying Statement-Level Performance Metric for HW/SW Technologies", In Companion of the 2018 ACM/SPEC Int. Conf. on Performance Engineering (ICPE '18).

[26] V. Muttillo, P. Giammatteo, V. Stoico, L. Pomante. An Early-Stage Statement-Level Metric for Energy Characterization of Embedded Processors. Microprocessors and Microsystems, 2020.

[27] Brandolese, C.; Fornaciari, W.; Salice, F. An area estimation methodology for FPGA based designs at systemc-level. Design Automation Conference, 2004. Proceedings. 41st, 2004 Page(s):129 – 132).

constraints. Then, the "Timing/Energy HW/SW Co-Simulation" activity considers suggested mapping/architecture items to actually check for Timing/Energy NFC satisfaction. If the suggested mapping/architecture item does not meet such constraints, the designer should perform again the DSE by changing some exploration parameters, by modifying the starting SBM, by enriching the TL with new elements, or by relaxing some constraints.

7.3.1.6 Algorithm-Level Flow

When the mapping/architecture item proposed by the DSE step is satisfactory, it is possible to implement the system. For this, the SW-mapped processes are typically transformed in C/C++ code, with the support of a possible embedded and/or real-time OS, while the HW-mapped ones are transformed in synthesizable HDL code or implemented by means of existing COTS component depending on the final system form factor. It is worth noting that such transformations will be done automatically or manually depending on the language and the coding style adopted to describe the SBM. This step is fully based on existing commercial algorithm-level methodologies and tools.

7.3.2 HEPSYCODE in the FRACTAL project

Given the ever-increasing opportunities provided by the advancement in the HW/SW technologies, there is a strong need for ESL methodologies and tools able to keep as much as possible smaller the design-productivity gap in the field of HW/SW dedicated systems. According to this scenario, the HEPSYCODE methodology, briefly described in the previous sub-section, can be adapted/integrated in the FRACTAL methodological framework. In particular, in order to support the FRACTAL designer during the mapping of a set of functionalities on a FRACTAL node and the related FRACTAL node customization, it is possible to:

- Adapt/integrate HEPSYCODE to fully exploit SoCs and processors relevant for the FRACTAL project (e.g., Xilinx ZYNQ/VERSAL, RISC-V) and the possible ASP/SPP provided by them (e.g., GPU, AI engine).
- Adapt/integrate HEPSYCODE to consider NFC relevant for the FRACTAL project (e.g., mixed-criticality to support safety, power/energy).

7.4 Methodologies for VERSAL platform

The main scope of work on the Versal node platform safety design will need to align the provision of the existing toolflow from the vendor Xilinx' ecosystem with the thirteen steps as forwarded in section 6.1.2.

PLC2 aims to provide a base design in due time for the Versal development platforms (VCK190) which are part of the certification evaluation process at Xilinx. This helps to simplify the separation methodology and the tool usage where a supported OS (Linux) is running on the APU that already provides control concepts of the actual Versal ACAP safety features to allow for integration of the services envisioned in WP3 and WP4. FRACTAL node specific functionality can add and further integrate into the PL and on AIE engines. Such additions should not impact the overall safety considerations. The basic Xilinx-side proposed workflow is listed here:

Design

1. H/W Partitioning
2. Failure modes identification
3. Safeness Estimation
4. Define requirements for phase STL and FSV

STL

5. Define SW requirements from FMEDA
6. STL definitions > design > implementation
7. Review the results from FSV

FSV (FuSa Verification)

8. Define Fault simulation environment
9. Fault simulation and analysis
10. Review process to STL phase

FuSa

11. Review and coordinate all the above phases
12. Methodology of maintenance and integration flow
13. Safeness metric generation

The actual protection and isolation concept will be prepared but will still allow platform operation with a to-be-defined set of operation modes with respect to power, function availability and more to support the adaptive node concepts. The objective is to provide a unified boot and initialization setup which can be used for all Versal based FRACTAL UCs. On top the development of this base platform needs to also address the services architecture along the WP4 requirement.

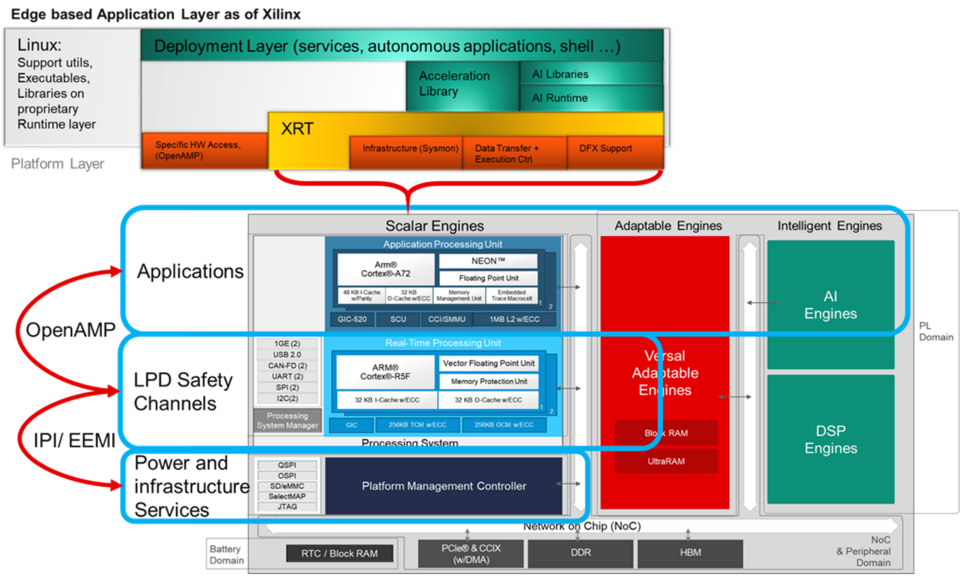

Figure 28 – Versal top level reference platform

The development of this base platform project will be guided by the steps of the proposed Fractal safety related methodology from section 6.1.2. Due requirement mapping and documentation will be carried out.

7.5 Methodologies for PULP platform

The Parallel Ultra Low Power (PULP) platform is a RISC-V based open-source platform for energy efficient computing. While there are multiple PULP systems with different capabilities, within the FRACTAL project the reference PULP system is the PULPissimo system available under:

https://github.com/pulp-platform/pulpissimo

The platform is fairly mature and has been used as part of various ASIC tapeouts so far, and comes with FPGA images for popular XILINX boards such as the Genesys II, allowing partners to hit the ground running. As part of FRACTAL, the goal is to improve and adapt this basis system for several use cases with additional capabilities developed by FRACTAL partners as part of technical developments WP4/5 and 6.

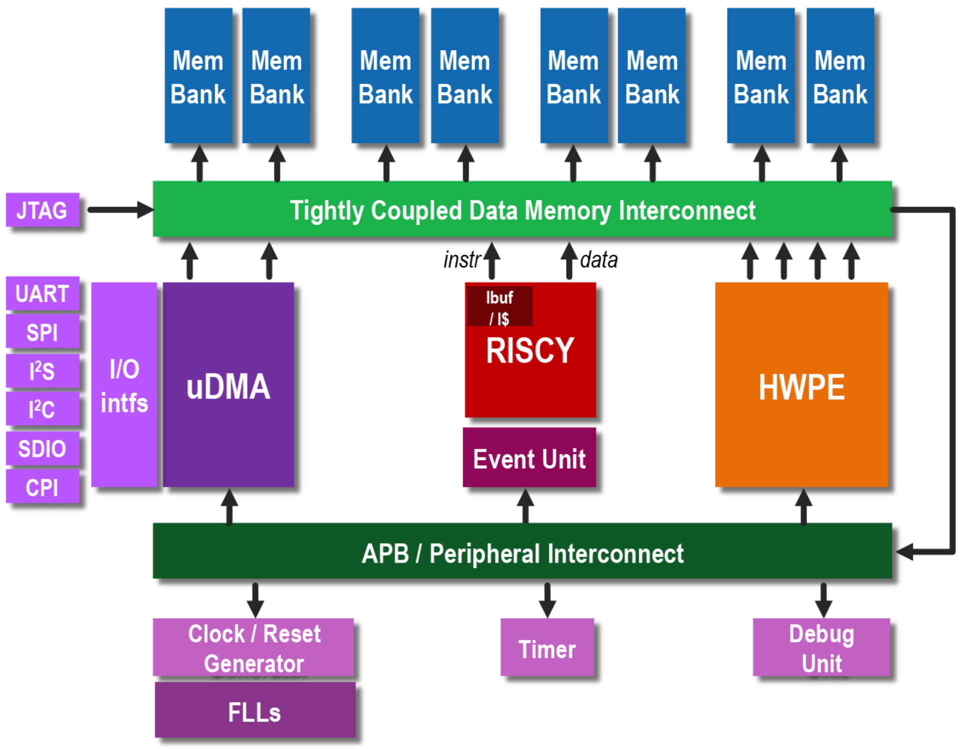

Figure 29 – PULPissimo block diagram

From the onset, the positioning of the PULP platform has been complimentary to the XILINX Versal platform. While Versal is addressing high performance, industrial maturity and established processor cores, for FRACTAL, PULP was designed to address low power IoT operation, with a flexible and open architecture allowing partners to easily integrate changes to the platform using an open-source ISA.

There are already a number of training videos and material available on the PULP platform WWW site under:

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

For example, the following (3.5h) training video will explain the design and features of PULPissimo:

https://www.youtube.com/watch?v=27tndT6cBH0

There are several different ways in how changes to PULPissimo as part of developments in FRACTAL can be implemented:

- **Adapting software** to be compatible with PULPissimo. At the moment, PULPissimo supports several basic operating systems such as FreeRTOS and seL4 and there have been successful ports of various other operating systems and libraries. PULP based system rely on standard GCC/LLVM based SW development flow and also include a custom software development kit. The following tutorial explains the PULP software development kit: https://youtu.be/Ydd9TlKQiO4

- Adding **accelerators and peripherals** to the PULPissimo system to enhance its hardware capabilities. In addition to standardized AXI/APB compatible peripherals, PULPissimo also supports accelerators that can access a shared scratchpad memory with the RISC-V processor greatly reducing the overhead of data transfers. The following tutorial is on the HW/SW co-development needed for such modifications: https://youtu.be/B7BtaYh3VqI
- Adding **instruction set extensions** to the RISC-V core in the system, which could improve the performance of the system significantly. RISC-V has a very clear methodology for such extensions, and ETH Zurich has significant experience with them. In fact, the default processor core in PULPissimo (RI5CY/CV32E40P) has already a large number of extensions for digital signal processing. However, SW tools must be made aware of these modifications, so that code can be generated that maps to these additional instructions.

Some FRACTAL partners have also expressed interest in more sophisticated systems from the PULP platform like the HERO (heterogeneous Research Platform) or the 64bit Ariane/CVA6 system with Linux support. ETH Zürich will guide partners that want to explore these options, but will concentrate their main effort around the PULPissimo.

The flexibility offered by the platform and the experience of ETH Zurich on actual ASIC implementations of the platform will help partners to assess the cost and benefit of various improvements and changes that will be explored throughout FRACTAL. Such evaluations will be of help for partners even in cases where a direct use case implementation will not be feasible due to technical reasons. For example, the feasibility of a low power mode can be tested and developed on a PULPissimo system and the results could be used to determine the effect of such an improvement on different systems as well.

7.6 Methodologies for hardware accelerators

Naturally, Deep Neural Networks (DNN) perform many computations on a huge amount of input data for generation of the outcome. The use of classic general-purpose computer designs has been shown to be not very efficient for execution of such networks due to the sequential concept of computation that these architectures apply. Therefore, building a specialized hardware that provides better performance and is energy efficient is required.

However, DNNs are a central part of machine learning which means they must be trained before they infer. During the training phase the DNN’s parameters (weights and bias) are derived. As a process training is much more complex than inference, where inference computes the outputs based on the previously trained weights and biases. Since training of DNN is a onetime process and requires much more computation power it can be performed on high performance machines that have no restriction on energy consumption. On the other hand, inference is usually applied on many devices located near the sensors and runs repeatedly with different input data patterns. Thus, specialized hardware needs to be customized for inference execution to satisfy performance requirements and energy efficiency goals.

The following subsections describe the methodology used for building and evaluation of the hardware accelerators in FRACTAL project. After listing the main components required for building of the hardware accelerator, the metrics that need to be observed for evaluation and optimization of the performance are described. All these steps are performed with the help of Catapult High-Level Synthesis tool, which enables the hardware designer to evaluate performance parameters at an earlier stage of the design before generating the hardware implementation details. The tool itself has integrated features that makes the performance evaluation a straightforward process. Once the design satisfies the performance requirements defined in the project the High-Level Synthesis tool will automatically generate the implementation outcome which can be for ASIC or FPGA technology.

7.6.1 Hardware Accelerator Architecture for Deep Neural Network

DNN accelerators typically consist of an array of processing elements (PE) and memory blocks interconnected by a Network on Chip (NoC). The PEs are simple Multiply-Accumulate (MAC) units capable to perform multiplication of inputs and weights and add the resulting products to the partial sum.

Memory of hardware accelerators are organized hierarchically consisted of register files (RF), global buffers, and main memory. RFs are the smallest and fastest memory units located on PEs. They hold data immediately available to the MAC unit. The type of data stored in RF can be weights (weight stationary), partial sums (output stationary), or a combination of both types (row stationary). Which type of data will be located on RF depends on the data flow model in use. There are hardware accelerators as well without RFs (no local reuses) in case the size of the chip area is critical. A global buffer is an intermediate memory layer located on-chip to hold fragments of the weights and inputs. Its location and size enable a global buffer to respond faster and efficiently when its content is reused by PEs. The last layer is the main memory, usually in form of off-chip DRAM memory, that holds all weights and input data.

Hardware accelerators for DNN can either be implemented on Field Programmable Gate Arrays (FPGA) or as Application Specific Integrated Circuits (ASIC). FPGAs are used for development of prototypes since the design for such technology is simple and the time to develop the product is shorter. On the other hand, ASICs can be optimized for much more parameters, have lower energy consumption and better computation performance. However, for ASIC technology the production costs are much higher combined with longer development cycles compare to FPGA solution.

7.6.2 Metrics for building optimized HW accelerator

To build an efficient hardware accelerator for DNN it is important to take into consideration a set of metrics that are key for achieving an optimal solution. Mainly the efficiency of the accelerator is associated with the number of operations per Watt, but this metric is a composition of accuracy, throughput, latency, energy consumption, cost, flexibility, and scalability[28]. *Accuracy* indicates the quality of inference outcome and should be high enough to perform correct classification. The accelerator should have capability to process enough data in a given period of time so its *throughput* can achieve real-time performance. Also, the time between input and output should be short if it is required to have low *latency*. When an accelerator is used within an edge device the *power consumption* and *energy efficiency* must be taken into consideration during design phase to ensure that the accelerator operates within the boundaries of the power envelope. It has been observed that frequent access to the main memory for data read/write is one of the main sources of energy consumption and compared to the demand of the various arithmetic operation performed in the accelerator it is few magnitudes higher. *Cost* is constrained by the required hardware volume and the size of the market. Obviously, it is important the designed hardware accelerator is attractive from a financial point of view. If the accelerator is more generic, it can execute a wider variety of different inference tasks, a feature which makes the accelerator more *flexible*. The last metric, *scalability*, refers to how well the accelerator can adopt from perspective of throughput and energy consumption when its number of components increases. Thorough upfront evaluation of all the mentioned metrics will allow the hardware designer to evaluate if the accelerator will be a beneficial and viable solution for a given application.

7.6.3 High Level Synthesis

Nowadays, hardware design processes have become quite complex and sometimes manual coding is even impossible due to increased complexity of the needed hardware solution. High-level synthesis (HLS) aims to generate synthesizable register transfer level (RTL) implementation of the hardware derived from its high-level specification. Its goal is to automate all the intermediated processes performed between specification of the hardware and the RTL level. The approach raises the level of abstraction on functional and implementation details and thus eliminates all the steps that were performed manually in the past. The whole concept is like the approach of compiling high-level code into assembler one. HLS also makes the verification process faster compared to the time required to perform the same process on Register Transfer Level (RTL), which sometime can take so long that the process itself becomes impractical. By elimination of the manual steps and reduction of the verification time HLS brings a lot of benefits:

[28] Vivienne Sze et al. „Efficient processing of deep neural Networks: A Tutorial and Survey" – Proceeding of the IEEE, Vol:105, Issue: 12, Dec 2017

- The automated generation of RTL implementation is less prone to errors due to the predefined rules applied for its generation.
- Well-designed automation tool can generate RTL that outperforms in quality compared to human design RTL level.
- Gives hardware designer more time to spend on design space exploration rather than on implementation.
- HLS simulation runtime is few times faster than RTL simulation runtime.
- HLS makes the design platform independent by enabling its implementation on a wide range of platforms.

The process of generating the RTL hardware description from its specification is a multi-level task[29] as shown in Figure 30. The process starts with describing the desired functionality of the hardware in a high-level language. The HLS tool takes the input, compiles it and generates a formal model. This model is then converted into a structured network of components (functional units, memories, controllers, and interfaces). During this stage the HLS firstly allocates the resources that are needed for computation. Next, the HLS schedules the execution order of derived operations. Thirdly, it binds the allocated resources to the corresponding operations derived from the formal model. These three tasks are interrelated and for optimal results they should be done in conjunction. The output is an RTL description of the hardware consisted of control and data path. It is important that HLS takes as input untimed high-level hardware description and transforms it into a fully timed hardware implementation.

The most popular languages used in HLS tools for functional description are SystemC and C/C++. The resulting RTL description is generated in form of popular Hardware Description Languages (HDL) like VHDL or VERILOG. However, not all C/C++ code structures can be converted into an HDL. Usually, non-synthesizable segments of the code are sections used for system calls, input/output structures, and pointers. Therefore, it is required from the implementer to distinguish between code structures that can and cannot be synthetized.

[29] Philippe Coussy et al. "An Introduction to High-Level Synthesis" – IEEE Design & Test Computers, 2009

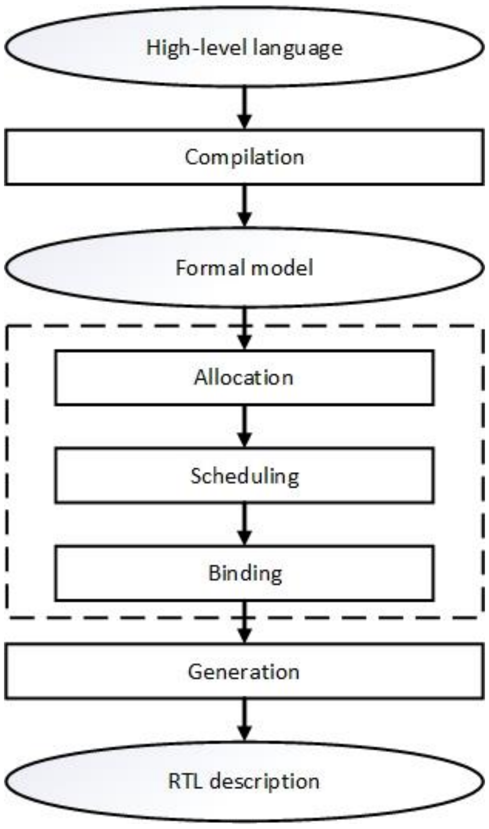

Figure 30 – HLS design flow

Mainly, HLS generates the HDL based on a set of predefined transformation rules. However, these rules can be overruled with pragmas within the source code if a better outcome can be produced. The usual techniques used in HLS design for performance improvement of the hardware are pipelining, unrolling, and in-lining[30]. Pipelining enables parallel execution within the loop, unrolling generates more processing elements in parallel, while in-lining removes the hierarchy. All these techniques can be implemented in form of pragmas or within the HLS tool.

7.7 Integration of Speech-based Signal Processing algorithms

The number of connected devices has increased rapidly. The Internet of Things (IoT) framework and the diffusion of related enabling technologies, such as Device-to-Device (D2D) communications, cloud and edge computing and big data analysis have strongly improved the feasibility of connecting and communicating through a large number of mobile nodes, often in non-ideal environmental conditions. Moreover, the

[30] Adam Taylor "Porting Vivado HLS design to Catapult HLS Platform" – White paper

importance of audio speech processing is demonstrated by the use of this type of approach in a wide variety of commercial applications. This growing interest in speech and speaker recognition is witnessed by the widespread use of applications, such as speaker verification and authentication procedures, gender recognition and language recognition. Other common applications of speech processing techniques lie in the range of accessibility solutions: the most remarkable examples of this kind are the speech-to-text and text-to-speech functionalities. Moreover, in numerous practical cases a speaker talks in an environment in which many smart devices (e.g., mobile phones) are present, for example during a seminar presentation, a conference call, or during a lecture in a classroom. For this reason, within the FRACTAL project, understanding audio context represents an important tool that can be extremely useful in several realistic scenarios. As the quality of audio signals is deeply influenced by the environmental conditions, an exhaustive study of the performances of the most common speech processing techniques in variable noise conditions and at different source-receiver distances is required. For this reason, many literature works address the issue of speech processing in challenging environmental conditions, proposing noise robust audio processing techniques, able to provide good performances even if noise is corrupting the audio signal. To the purpose of enhancing the recognition performances in challenging environmental conditions, in the FRACTAL project we propose a noise and distance robust Speech/speaker identification method, which embeds a smart pre-processing method employing Voice Activity Detection (VAD), capable to boost the system accuracy in terms of correct classification rate.

The problem of Speaker Recognition is to recognize the identity of the speaker who sounds closest to speech analyzed from an audio sample produced by an unknown speaker. It can be tackled in two different scenarios: Closed-Set scenario, when the recognized speaker belongs to a given, a-priori known set, and Open-set scenario (also called out-of-set speaker identification, if the identity of the test subject could also belong to a speaker who is not part of the predefined known speaker group. In the following we tackle the speaker identification problem in both the aforementioned situations. It is worth noticing that the gender recognition and the language recognition approaches uses similar methods (just the data sets change).

The acquired audio signal is divided into short segments called frames, during which speech can be considered as stationary. Each frame has a length of T= 25 [ms] and it is selected so that the time distance between the centers of two adjacent frames is equal to 10 [ms] (i.e., two consecutive frames are overlapped for one third of their duration). After the framing block, the features are computed. We employ the first 13 Mel-Frequency Cepstral Coefficients (MFCC) and the respective 13 Delta-Delta (i.e., the second-order derivative of the MFCC coefficients) for a total of 26 features. Once they have been extracted, the features will be used to train a supervised classifier. The first step in order to obtain a good classification system is to pre-process the signal by removing frames that do not contain useful information. When dealing with audio speech signals, this step translates into discarding the audio frames that do not contain speech utterances. We call this Voice Activity Detection

(VAD). To achieve this goal, many techniques have been proposed in the literature. One of the simplest consists in applying a Band Pass Filter (BPF) centered over the speech bandwidth (e.g., from about 50[Hz] to 3500[Hz]). This action will remove unwanted frequency components that do not fall within the voice bandwidth, but it does not ensure that the remaining signal components are actually related to speech utterances. In FRACTAL we employ an alternative approach to voice activity detection which aims at smartly taking into account only actual speech frames. We call this pre-processing method SmartVAD, which consists in a short-time spectral analysis pre-processing filtering system. It is based upon two important indicators: *i)* Spectrum Flatness Index (SFI) and ii) Energy Ratio Index (ERI). The rationale behind these two parameters is that a speech frame exhibits a spectrum having most of its energy within the 1st [kHz] and which should not be flat. Finally, a threshold criterion is applied such that an audio frame is not discarded only if the value of the considered indicators satisfies the threshold condition. These are important indicators of a voiced frame, and they motivate the choice of the proposed VAD indicators, which effectively represent the nature of the considered signal.

Concerning artificial intelligent functions implementing the recognition phase, we employ a Support Vector Machines model. SVM is a supervised learning scheme that uses a binary approach to assign samples to a class, by dividing the feature space into different regions, one corresponding to each category. The SVM algorithm works in two separate phases: training phase and testing phase.

Training Phase: we define the feature vector for a given frame out of F frames for each recording. We define as well, the vector containing all the classes (i.e., the names of all speakers in the following but in general can be the gender of the speaker and/or the language). The main idea of the SVM algorithm is to separate the feature space by means of one single hyperplane: in this project we employ both the One-Against-All (OAA) method that constructs a SVM for each class considered recognition function, and the One-Against-One (OAO) approach, which builds more models.

The main difference between the two types of classification schemes is that, whereas the OAA approach consists in training one model for each considered known speaker, the OAO approach performs a supervised training of one single model, by assigning a different class to each speaker identity. In other words, in the OAA approach, the single SVM for a class is trained by employing all the feature vectors of a given training set belonging to the given class with positive labels and all the other feature vectors belonging to the other classes with negative labels, so performing a sort of binary partitioning. The OAO approach, instead, consists in training one single SVM for all the considered classes, by assigning different label values to different classes, for a total number of labels equal to the number of classes to identify, thus performing a so-called multi-class partitioning. Starting from the audio signals of the training set, the single SVM, built for a class, can be obtained by computing the aforementioned hyperplane that can be expressed as a function of its orthogonal vector. Analytical descriptions of the SVMs have been omitted for the sake of brevity.

Decision (Testing) phase: every time that a speaker must be recognized, the SVM outputs a matrix, called Probability Matrix (PM), one for each trained hyperplane. The number of PMs is equal the number of the considered speakers for the OAA case, whereas it corresponds to the number of all possible combinations of all the considered speakers taken in pairs, for the OAO case. Each element of this matrix is the a-posteriori probability of a given feature vector belonging to the class identified by the binary label. From all the PMs a decision matrix is computed. It is a binary matrix where each element $\{-1, +1\}$ is obtained by associating the given frame to the binary label which has the highest probability. From the decision matrix we determine the scoring vector S, which represents the scoring of the a given. It is a measure of the likelihood of the input speech utterance to belong to such a speaker. Finally, from the scoring vector the Maximum Likelihood Index (MLI) is inferred. It is simply the index of the speaker, among the predefined set, who has the highest score value (i.e., it has the highest a-posteriori probability to have produced the input speech signal). The recognized speaker is then determined by employing a decision rule that change in case of Open-Set or Close-Set scenario. In other words, for what concerns the open-set scenario (which is the one targeted within the FRACTAL project), the maximum score obtained by the speakers belonging to the known speaker set is compared to a predefined threshold. If the highest score is above the considered threshold the recognized speaker is the one who produced the maximum score, otherwise the classifier chooses for an Unknown speaker.

8 Conclusions

In this deliverable, first, we have introduced challenges for developing the FRACTAL node as a system of systems, and a set of key concepts that are needed to define a framework.

This document has introduced the framework of FRACTAL for taking AI and Safe Autonomous decisions, from different perspectives, starting from the framework to the services, libraries, strategies and algorithms.

This methodological framework also relates to the safety aspects to be considered in FRACTAL nodes, that can be addressed by a series of different safety standards depending on the target applications. Apart of the diversity of application specific standards to be taken into account we also have considered safety communications, safety aspects of Artificial Intelligence and how the building block approach can fit into the FRACTAL framework.

Integration aspects have also been taken into account, how different SW items, HW platforms, different applications… are integrated together in a unified platform.

A general methodology for FRACTAL system development has been presented that will be elaborated in other deliverables. We described the overall workflow of the project with a focus on the identification of the key enabling technologies, the task of this deliverable, based on the use case descriptions.

9 List of Figures

10 List of Tables

11 List of Abbreviations

ACAP	Adaptive Compute Acceleration Platform (relates to VERSAL)
ACE	AXI Coherency Extensions
ADMA	Accelerator coherence port Direct Memory Access controller
AES	Advanced Encryption Standard
AHB	Advanced High-performance Bus
AI	Artificial Intelligence
AIE	AI Engine
AIP	Artificial Intelligence Processor
AMBA	Advanced Microcontroller Bus Architecture
AMP	Asymmetric Multi-Processing
APB	Advanced Peripheral Bus
API	Application Programming Interface
APU	Accelerated Processing Unit
ASIC	Application Specific Integrated Circuits
ASIL	Automotive Safety Integrity Level
ASP	Application-Specific Processors
AXI	Advanced eXtensible Interface
BBRAM	Battery Backed Random Access Memory
BIST	Built-In Self Test
BPF	Band Pass Filter
CAN	Controller Area Network
CAN FD	Controller Area Network Flexible Data-Rate
CCF	Common Cause Failure
CCI	Cache Coherent Interconnect
CENELEC	Comité Européen de Normalisation Électrotechnique (English: European Committee for Electrotechnical Standardization)
COTS	Component Off The Shelf
CPS	Cyber Physical System
CPU	Control Processing Unit
CSP	Concurrent Sequential Processes
CUDA	Compute Unified Device Architecture
DDR	Double Data Rate
DDRAM	Double data rate Dynamic Random Access Memory
DHPS	Dedicated Heterogeneous Parallel Systems
DL	Deep Learning
DNN	Deep Neural Network
DRAM	Dynamic Random Access Memory
DoA	Description of Action
DS	Dedicated Systems
DSE	Design Space Exploration

DSP	Digital Signal Processor
DT	Decision Tree
ECC	Error Correction Code
EDDL	European Distributed Deep Learning Library
EN	European Norm
ERI	Energy Ratio Index
ESL	Electronic System Level
ETC	Energy-To-completion
EU	European Union
FIT	Failure In Time
FMEDA	Failure Modes Effects and Diagnostic Analysis
FPD	Full Power Domain
FPGA	Field Programmable Gate Arrays
FSV	FuSa Verification
FUB	Functional Unit Block
FuSa	Functional Safety
GAN	Generative Adversarial Network
GCC	GNU C Compiler
GIC	Global Interrupt Controller
GNU	GNU is Not Unix
GPIO	General Purpose Input/Output
GPP	General-Purpose Processor
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HEPSYCODE	HW/SW CO-DEsign of HEterogeneous Parallel dedicated SYstems
HERO	Heterogeneous Research Platform
HLS	High-level synthesis
HPC	High Performance Computing
HPS	Heterogeneous Parallel Systems
HPV	Hypervisor technologies
HW	Hardware
IC	Instruction Count or Integrated Circuit (context dependent)
IEC	International Electrotechnical Commission
IoT	Internet of Things
IP	Intellectual Property
ISA	Instruction Set Architecture
JPEG	Joint Picture Action Group
KPI	Key Performance Indicator
LEDEL	Low Energy DEep Learning Library
LLVM	Low Level Virtual Machine (compiler)
LPD	Low Power Domain
LR	Logistic Regression

MAC	Multiply-Accumulate
MBIST	Memory Built-In Self Test
MFCC	Mel-Frequency Cepstral Coefficients
ML	Machine Learning
MLI	Maximum Likelihood Index
MLP	Multilayer Perceptron
MMU	Memory Management Unit
MoC	Model of Computation
MPSoC	Multi Processor SoC
NDA	Non Disclosure Agreement
NF	Non-Functional
NFC	Non-Functional constraints
NoC	Network on Chip
NP	Network Processor
NPI	NoC Programming Interface
OAA	One Against All
OAO	One Against One
OCM	On Chip Memory
ONNX	Open Neural Network eXchange
OS	Operating System
PCB	Printed Circuit Board
PCM	Phase Change Memory
PE	Processing Element
PL	Programmable Logic
PM	Probability Matrix
PMC	Platform Management Controller
PS	Processing System
PSM	Processing System Manager
PULP	Parallel Ultra Low Power
QM	Quality Management
QoS	Quality of Service
R&D	Research & Development
RAM	Random Access Memory
RAMS	Reliability, Availability, Maintainability and Safety
RF	Register File
RI	Reference Inputs
RIA	Research and Innovation Actions
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
RPU	Real time Processing Unit
RTC	Real Time Clock
RTL	Register Transfer Level

RTOS	Real Time Operating System
SBM	System Behavior Model
SBS	System Behavior Specification
SC	SIL capable
SC 3	SIL 3 capable
SECDED	Single Error Correction Double Error Detection
SFI	Spectrum Flatness Index
SIL	Safety Integrity Level
SMMU	System Memory Management Unit
SMP	Symmetric Multi-Processing
SoC	System-on-Chip
SOTIF	Safety of Intended Functionality
SPI	Synchronous Peripheral Interface
SPP	Single-Purpose Processors
STL	Software Test Library
SU	Statistics Unit
SVM	Support Vector Machine
SW	Software
TCM	Tightly Coupled Memory
TL	Technologies Library
TMR	Triple Mode Redundancy
TRC	Time-To-Reaction
TTC	Time-To-Completion constraint
TÜV	Technischer Überwachungsverein (German safety and standards institution)
UART	Universal Asynchronous Receive/Transmit
UC	Use case
V&V	Verification and Validation
VAD	Voice Activity Detection
VCA	Video Content Analysis
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WDT	WatchDog Timer
WP	Work Package
XAI	Explainable AI
XMPU	Xilinx Memory Protection Unit
XPPU	Xilinx Peripheral Protection Unit
XRAM	Accelerator RAM
YOLO	You Only Look Once

The short names of FRACTAL partners are not considered as abbreviations: ACP, AITEK, AVL, BEE, BSC, CAF, ETH, HALTIAN, IKER, LKS, MODIS, OFFC, PLC2, PROINTEC, QUA, ROT, RULEX, SIEG, SIEM, SML, THA, UNIGE, UNIMORE, UNIVAQ, UOULU, UPV, VIF, ZYLK.