Project:	ISOLDE
Reference number:	101112274
Project duration:	01.05.2023 - 30.04.2026
Work Package:	WP1: Requirements and Specifications
Deliverable	**D1.1**
Title	**Demonstrators Requirements and Specifications**
Type of deliverable:	Report
Deadline:	31.08.2023
Creation date:	25.11.2023
Authors:	Patrick Pype, Bogdan Kiszka, Antonio Sciarappa, Cosmin Moisa, Danut Rotar, Mikai Hurdugaciu, Aurel Gontean, Holger Blasum, Darshak Sheladiya, Samuel Ardaya-Lieb, Thomas Hartmann, Dominik Riemer, Michael Gautschi, Yvan Tortorella, Francesco Conti, Alessio Burrello, Daniele Jahier Pagliari, Calin Bira, Dario Pascucci, Antonio Leboffe, Alessandro Marini, Paolo Serri, Catriel De Biase, Salvatore Cognetta, Federico Reghenzani, Giovanni Agosta, Martin Hobelsberger
Involved partners:	AXELERA, CAR-RO, CONS, E4, IFX, LDO, NXP-AT, NXP-CZ, POLIMI, POLITO, SAL, SYSGO, TASI, UNIBO, HM, UNSTPB, FEN

Table of Contents

1 Introduction

1.1 Executive Summary

The purpose of this document is to collect the initial requirements for the demonstrators in the chosen application areas (Automotive, Space, Smart home, IoT), as worked out in T1.1.

It contains both functional and non-functional requirements, at least from the following subjects: architecture (components, interfaces, accelerators), measurable metrics and goals (performances, power consumption, radiation tolerance, security metrics to be used), SW requirements (stacks, tools, system SW features, SW for demonstrators use cases), and standard to be used/considered.

The main goal of this document is to have a sound starting basis for further iteration and refinement in the different WPs of the project. Some of the requirements have already been worked out in full detail, while others still contain some open questions or issues to be finalized. For example, regarding verification and validation, further refinement and updates are still needed. Such activity will be carried out in D1.3 where the final requirements will be frozen.

This document supports the technical report associated to the deliverable D1.1 “Demonstrators Requirements and Specifications” reporting to the WP1 “Requirements and Specifications” leader (E4 Computer Engineering S.p.A.) for the Project ISOLDE.

The document is prepared in the frame of the Task 1.1 – Collection of requirements of the demonstrators & high-level SoCs specifications (WP1).

1.2 Definitions and Acronyms

Abbreviation	Description		Abbreviation	Description
AI	Artificial Intelligence		LLVM	Low Level Virtual Machine
AMD	Advanced Micro Devices (Company)		LTE	Long-Term Evolution (4G LTE)
ARM	Processor company, formerly Advanced RISC Machines		MIPI	Mobile Industry Processor Interface
ASIC	Application Specific Integrated Circuit		ML	Machine Learning
BSD	Berkeley Standard Distribution		MTC	Machine-type communication
CA	Consortium Agreement		MTH	(Timing) Mid Term
CCAM	Connected, Cooperative and Automated Mobility		NFC	Near Field Communication
CFI	Control Flow Integrity		NR	New Radio (standard)
CFR	Control Flow Reconstruction		OBI	On Board Interconnect
CLIC	Core Level Interrupt Controller		OS	Operating System

CLS	Controllable Local System	OSI	Open Standards Interconnect
CPU	Central Processing Unit	PCI	Peripheral Component Interconnect
CSI	Camera Serial Interface	PCIE	PCI Express
CU	(Timing) Continuous updates	PDK	Physical Design Kit
CUDA	Compute Unified Device Architecture		
DM	(Timing) Direct Meetings	PHY	PHYsical layer (implementation)
DRAM	Dynamic Random Access Memory	PKI	Public Key Infrastructure
EC	European Commission	PMOD	Peripheral Module (interface)
ECC	Error Check and Correct	PQC	Post Quantum Cryptography
EDA	Electronic Design Automation	PV	Photovoltaic
EMS	Energy management system	QSPI	Quad Serial Peripheral Interface
EoL	End-of-Life	R&I	Research and Innovation
EPI	European Processor Initiative	RISC	Reduced instruction set computer
EY	(Timing) Each Year	RISC-V	Specific Open-Sourced RISC ISA
FC	(Timing) Final Conference	RTL	Register Transfer Language
FIFO	First In First Out (Buffer)	RTO	Research & Technology Organisation
FLOP	Floating Point Operations per Second; unit to measure computing performance	SDC	Synopsys Design Constraints File
FMC	FPGA Mezzanine Card	SDIO	Serial Digital Input/Output
FPGA	Field-Programmable Gate Array	SGA	Specific Grant Agreement
FPU	Floating Point Unit	SHAKTI	RISC-V CPU from RISE, India
GAFA	Google, Amazon, Facebook, Apple	SIL	Safety Integrity Level
GCC	GNU Compiler Collection	SIMD	Single Instruction Multiple Data
GFLOPS	Giga Floating Point Operations per Second	SME	Small/Medium Enterprise

GPU	Graphic Processing Unit	SMGW	Smart Meter Gateway
GTH	Gigabit Transceiver type H (Xilinx/AMD)	SNS	Smart Networks and Services
GPP	General Purpose Processor	SoC	System on Chip
HBM	High Bandwidth Memory	ST	(Timing) Start of TRISTAN project
HEMS	Home Energy Management System	SW	Software
HPC	High Performance Computing	TC	(Timing) Targeted Conference
HSI	Hardware-supported instrumentation	TF	(Timing) Targeted Fair
HW	Hardware	TID	Total Ionizing Dose
I3C MIPI	I3C successor to I2C	TNID	Total Non-Ionizing Dose
ICT	Information and Communication Technology	TRL	Technology Readiness Level
IDE	Integrated Development Environment	TSN	Time Sensitive Networking
IEC	International Electrotechnical Commission	UART	Universal Asynchronous Receiver/Transmitter
IEEE	Institution of Electrical and Electronic Engineers	VHDL	VHSIC Hardware Description Language
IOMMU	I/O Memory Management Unit	VHSIC	Very High-Speed Integrated Circuits
IoT	Internet-of-Things	VP	Virtual Prototype
IP	Intellectual Property	VRP	Variable Precision
ISA	Instruction Set Architecture	VXP	Variable Extended Precision
ISO	International Standards Organisation	WI	Work Item
ISS	Instruction Set Simulator	WP	Work Package
JTAG	Joint Test Access Group	X86	(Intel originated) X86 instruction set
KDT	Key Digital Technologies	XNG	XtratuM Next Generation
KDT-JU	KDT Joint Undertaking	XPU	X Processor Unit (CPU, GPU …)
KPI	Key Performance Indicator		

2 Space Demonstrator

2.1 Overview of the Demonstrator

The space use case demonstrator focuses on testing the feasibility of performing inference of deep learning models on a RISC-V processor aboard a satellite. The demonstrator will use different data sources, such as hyperspectral imaging data and telemetry data produced on board, to monitor the health status of the satellite itself.

The demonstrator will showcase the capabilities of the RISC-V-based space processor in supporting image processing tasks. The data consists of hyperspectral images, where each pixel represents the reflectance of light within a fixed spectral range. To ensure computational efficiency for space applications, the model will process the image on a pixel-by-pixel basis, determining whether each pixel corresponds to the target material. The algorithm will perform data normalization, feature extraction and leverage neural network-based models for target classification. Furthermore, we will explore potential opportunities for parallelizing the processing, whenever applicable.

The demonstrator will also showcase the capability of the RISC-V-based space processor to support Machine Learning (ML) applications, specifically for telemetry analysis. The ability to detect anomalies and to identify trends in satellite data is crucial for ensuring mission success while preventing costly downtimes. By leveraging advancements in ML algorithms, we can train models to recognize patterns in telemetry data and to alert operators on potential issues before they become critical.

One possible scenario involves using sensor data from various subsystems within the satellite to monitor performance and to identify deviations from the expected behavior. For instance, the demonstrator could utilize accelerometer readings to track changes in the satellite's orientation or gyroscope measurements to assess spin rate fluctuations. Additionally, it could analyze temperature and power consumption data to ensure that the satellite's systems are operating within safe margins.

To implement this use case, the demonstrator will employ a combination of feature engineering and model training techniques. Feature engineering involves extracting relevant information from raw telemetry data, such as time-series data, and transforming it into features suitable for model training. This may include techniques like normalization, smoothing, and extraction of relevant frequency components.

Once the features are prepared, the demonstrator will utilize the already (on-ground) trained ML algorithms to identify patterns and anomalies in real-time data, which can then be used to generate alerts and notifications for operators on-ground or for on-board autonomous failure isolation and recovery system. The latter mechanism is called Fault Detection, Isolation and Recovery (FDIR) system and is currently based on traditional Out-Of-Limit (OOL) techniques, where an alert is sent only if the telemetry exceeds the pre-fixed upper and lower limit values. Injecting the ML models in this system will enhance the FDIR capabilities in anomaly detection.

It is important to notice that the current state of space technology poses challenges for performing the entire processing on-board a satellite (training and inference). As a result, the primary focus lies in optimizing the inference phase, which is well-suited for on-board execution, leaving the training phase to the beefy servers and workstations on-ground.

2.1.1 Description of the space processor

The space processor employed in the demonstrator is based on the CVA-6[1] RISC-V architecture, preferably multi-core. RISC-V processors are known for their energy efficiency, making them well-suited for satellite applications where power consumption is critical. By utilizing RISC-V, the processor ensures optimal performance while adhering to the stringent resource constraints of space missions, with the possibility to benefit from the improvements coming from the community of RISC-V developers[2].

2.1.2 Functionalities and capabilities

The demonstrator showcases the ability to perform inference of a deep learning model directly on the RISC-V-based hardware present onto the satellite. This capability enables processing data onboard the satellite itself, reducing the need for extensive data transmission back to earth for analysis.

2.1.3 Key features and benefits

The primary benefit of demonstrating deep learning model inference on board of a satellite lies in the ability to move a significant portion of the computation to the edge. By processing the data locally on the satellite using the RISC-V processor, only the relevant results and features extracted need to be transmitted to earth. This approach significantly reduces transmission time, data bandwidth, power consumption and associated costs. Moreover, it enhances the overall efficiency of satellite missions, as critical decisions can be made in real-time, thus opening new possibilities for remote sensing applications and space-based research.

[1] GitHub - openhwgroup/cva6: The CORE-V CVA6 is an Application class 6-stage RISC-V CPU capable of booting Linux

[2] Technical Working Groups - Home - RISC-V International (riscv.org)

2.2 Functional and Non-Functional Requirements

In this Section we collect the initial functional and non-functional requirements for the Space use case relative to the components developed in the various WP (Cores, Accelerators, Software). These requirements will be revised, refined and consolidated in the future D1.3 deliverable.

2.2.1 Cores

Regarding the cores developed in WP2, the space demonstrator will require:

- The CVA-6 processor core, possibly in a multi-core configuration
- CVA-6 improvements and extensions developed in ISOLDE

2.2.2 Accelerators

Relatively to the accelerators developed in WP3, based on their maturity level and their potential performance improvements to the final application, the space demonstrator may require all or only a subset of the following items:

- Root-of-Trust Unit (T3.1).
- Custom arithmetic units for AI and their power monitors (T3.2, T3.3) for accelerating. AI inference and evaluating its performance.
- SIMD/Vector unit and relative scratchpad memories (T3.2, T3.4) for accelerating the use case application.
- Tensor Processing Unit for AI (T3.4) for accelerating AI inference on board.
- Accelerators for post-quantum cryptographic accelerators (T3.5).
- Neuromorphic accelerators (T3.6).

2.2.3 Software

Concerning the software stack developed in WP4, the space demonstrator may require all or a subset of the following elements, also depending on their maturity level and degree of utilization of the accelerators:

- Accelerated AI libraries and related software toolchains (T4.1, T4.2) to deploy the AI application using the relative WP3 accelerator and the CVA-6 processor. Based on the requirements and the hardware chosen, different libraries and optimization techniques will be employed to enhance the performance of the space demonstrator.
- System software supporting the space demonstrator (T4.1) to effectively run the use case application. The foreseen demonstrator will involve neural network inference and/or execution of heavy algorithms; to optimize the execution, a system software dedicated to the target will be developed supporting the space demonstrator needs. The main requirements of the system software are to:
  - Provide a way to execute cyclic activities (with or without OS).
  - Provide a Command and Control front end to the operator exploiting one of the target input/output board communication links, where the Command and Control SW shall be able at minimum to start/stop an algorithm execution as well as monitor the execution status.
  - Provide a SW abstraction layer to leverage the HW accelerator.
- Analyze hypervisor support SYSGO (T4.1).
- Compiler and compiler extensions (T4.2) to compile the application enabling utilization of the WP3 accelerators; more precisely:
  - A compiler extension that will automatically manage the performance tradeoff of floating point/fixed point arithmetic;

    - An automatic way to estimate the WCET of the real-time tasks during the compilation process and, possibly, perform optimizations targeted at improving the WCET.
- Software support for Tensor Processing Unit and Root-of-Trust unit (T4.2) for deploying the AI application using the Tensor Processing unit and the Root-of-Trust unit.
- System-level RISC-V simulators (T4.3) of extra-functional properties. Jointly with the target hardware functional simulators, the extra-functional simulator will be used to estimate the power, energy, and latency of the space demonstrator while prototyping the first version. The data will be used to finetune the application further and to fulfil the use-case constraints.
- Hardware analysis of the CVA-6-based SoC, including an identification of multicore interference channels and a study of main critical configuration settings having an impact on multicore interference. Support on RVS RapiTime tool for the actual target for measurement of relevant hardware metrics and Worst-Case Execution Time (WCET) in contention scenarios (T4.3).
- Software power and timing analysis tools (T4.3) for collecting performance measurements of the use case, using novel ways to estimate the WCET (including probabilistic analysis) to overcome the traditional issues of static analyses.
- Hypervisor virtualization support for analysis, monitoring, and testing (T4.3). Port of XNG hypervisor providing support for software timing analysis tools developed by RAPITA. Provide hypervisor services allowing access to necessary on-chip resources used by timing analysis tools, including the different Hardware Event Monitors in the platform and critical configuration settings relevant to contention. Also, provide a building environment in order to ease the development and building of the tools which will support XNG hypervisor. XNG hypervisor tests integration to allow the diagnosis of faults at runtime by employing the fault injection tool developed by UPV. This enhances the safety of the system by identifying potential faults and avoiding major failures.
- ML based power/performance trade-offs modeling of CVA 6 core used for the space demonstrator to find best trade-off in order to increase autonomy when performing on-board processing.

2.2.4 Other requirements

No other requirements have been identified so far.

2.3 Measurable metrics

In this Section we collect an initial version of the measurable metrics for the space use case, indicating the requirements that the final product should satisfy in the case of its actual utilization on satellites. While it will not be possible to validate some of these requirements during the execution of the project, since this can only be done with a final chip available and under particular experimental conditions, we can still identify an initial list, with the purpose to verify if some items need to be further investigated or if some appropriate estimates can be conceived. As for the previous section, these requirements will be revised, refined and consolidated in the future D1.3 deliverable.

Note: for verification we will use the notation

- *[T] for those requirements that need to be tested on hardware*
- *[R] for those requirements that can be verified by checking the project documentation*

*2.*3.1 Performance Requirements

SPACE-001	**Chip Core**	Verification: [R]
The chip shall contain, in its HOST part, two different cores which may be configured to work either in Lock Step or in parallel (multicore) configuration.		

SPACE-002	**Chip Cluster SIMD**	Verification: [R]
The chip shall, in its CLUSTER part, work in a configuration scalable up to at least 8 (To be confirmed) cluster x 16 (To be confirmed) cores, which may be configured to work either with internal redundancy or purely in parallel. *Comment: the configuration to be implemented in the final ASIC solution is determined based on the outcome of the FPGA prototyping activities.*		

SPACE-003	**Cluster FLOP Performances**	Verification: [R]
The CLUSTER part of the chip shall assure at least 500 GFLOPS		

SPACE-004	**HOST DMIPS Performances**	Verification: [R]
The processor core used in the HOST part of the chip shall guarantee at least 1.7 DMIPS/MHz Dhrystone MIPS (Million Instructions per Second). *Note: It is expected to reach at least 1700 DMIPS in a lockstep configuration @1GHz and 3400 DMIPS in parallel configuration.*		

SPACE-005	**HOST FLOP Performances**	Verification: [R]
The processor core used in the HOST part of the chip shall guarantee at least 0.3 FLOP/cycle. *Note: It is expected to reach at least 270 MFLOPS in a lockstep configuration @1GHz* *and 540 MFLOPS in parallel configuration*		

2.3.2 Design Requirements

SPACE-006	Chip Architecture	Verification: [R]
The chip architecture shall be the RISC-V CVA6 core.		

SPACE-007	Processor Features	Verification: [R]
The chip architecture shall support at least the following features: • multicore architecture (i.e., SMP); • cache coherency management; • advanced interrupt support (e.g., platform interrupt controller); • virtual memory management; • virtualization management (i.e., hypervisor support).		

3.3.3 Power Requirements

SPACE-008	Power Consumption	Verification: [R]
The chip shall not exceed a power consumption of 4 W (TBC) in an operational environment of +125 C° @ EOL.		

SPACE-009	Current Consumption	Verification: [R]
The chip shall not exceed a power consumption of 750 mA (TBC) in an operational environment of +125 C° @ EOL.		

SPACE-010	Input Voltage	Verification: [R]
The chip shall work with an input voltage of 3V (TBC) in an operational environment of +125 C° @ EOL.		

SPACE-011	Exceeding Maximum Rating	Verification: [R]
All the EEE in which exceeding absolute maximum rating conditions might cause instantaneous or very short-term unrecoverable hard failure (destructive breakdown) shall be reported in the design document with a table like the following one:		

PARAMETER	MIN	MAX	UNIT
VDIG-3Vx (Part name #1)	0.3	4.5	V
VDIG-3Vy (Part name #2)	20	150	mA
…	…	…	…

SPACE-012	Working near Maximum Rating	Verification: [R]

All the EEE that are stressed from operating near absolute maximum levels that affect long-term reliability of the device shall be reported in the design document with a table like the following one:

PARAMETER	MIN	MAX	UNIT
VDIG-3Vx (Part name #1)	0.3	4.5	V
VDIG-3Vy (Part name #2)	20	150	mA
…	…	…	…

3.3.4 Radiation Requirements

Space radiation environments (e.g., Galactic Cosmic Rays, Solar Flares, Van Allen Belts) will induce effects on electronic devices and materials. The particle of concern for such radiations are mainly ions, protons, electrons and gamma rays (as secondary's of electrons, generated inside the satellite structure), which can cause effects such as:

- Single Event Effects (SEE).
- Displacement Damage (Total Non-Ionizing Dose).
- Total Ionizing Dose (TID).

The Radiation Source - Particle type – Effects cross table is schematically shown in the Figure 1 below:

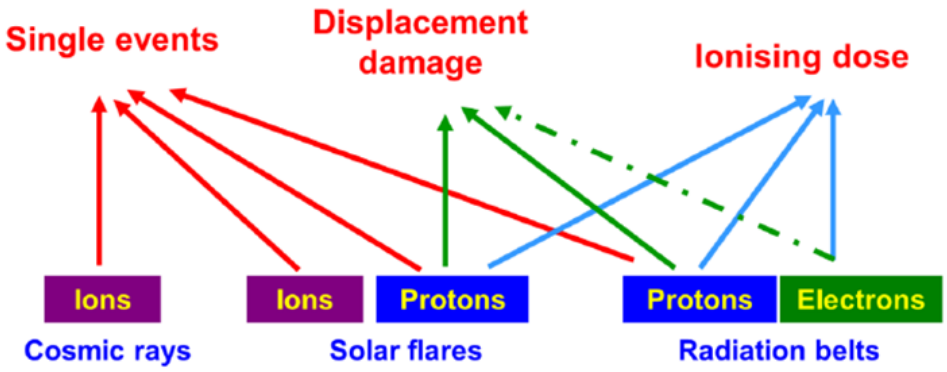

Figure 1: Radiation Source – Particle Type – Effects cross table

SPACE-013	Mission Parameter	Verification: [R]

The chip shall withstand a 12.5 years mission in a LEO orbit environment.

SPACE-014	Total Ionizing Dose	Verification: [R]

The chip shall withstand a minimum Total Ionizing Dose of 5 krad(si).

SPACE-015	Destructive Single Event	Verification: [R]

The chip shall withstand a $LET_{threshold}$ > **15 MeV.cm²/mg**. DSE free to proton.

SPACE-016	Non-Destructive Single Event	Verification: [R]
The chip shall withstand a $LET_{threshold}$ > **0.4 MeV.cm²/mg**. Not sensitive to direct ionization by protons.		

SPACE-017	Radiation Design Margin	Verification: [R]
The chip shall take into account a Radiation Design Margin, RDM = 1.20 to be applied on number given in requirements for TID and TNID at End-of-Life levels		

SPACE-018	Radiation Hardness Assurance	Verification: [R]
The chip design shall take into account the ECSS-Q-ST-60-15C “Radiation Hardness Assurance Requirements for EEE Parts”		

2.4 FPGA Prototype Requirements

In this Section we collect an initial version of the requirements that will be used to derive the FPGA for the integration of the space demonstrator in WP5. Once again, these requirements will be revised, refined and consolidated in the future D1.3 deliverable.

2.4.1 Vendor/Toolchain Preferences

There are several FPGA vendors with AMD/Xilinx and Intel/Altera being the most prominent ones. Usually, each vendor provides its own toolchain for code synthesis, simulation, timing analysis, floor planning, programming, verification, live introspection, debugging and so on. Since there is no cross-vendor toolchain covering the whole scope and mastering a toolchain requires time and training, the toolchain (and vendor, thus) can be seen as a requirement for choosing an FPGA. A preliminary survey among all partners involved in the space use case, however, showed that the majority of the participants are in favor of Xilinx and on the use of the relative toolchains.

2.4.2 Programming Language Requirements

FPGAs are typically programmed in VHDL, Verilog and SystemVerilog, or by exploiting high-level synthesis (HLS), where tools are used to map high-level code (e.g., C++) to the available hardware. While the first three formalisms are industry standards and typically available for any FPGA, the latter is much more customized to the used FPGA and its hardware capabilities as well as to the source programming language. The survey carried out among the partners involved in the space use case showed that SystemVerilog is the most commonly used high level formalisms for design and verification.

2.4.3 FPGA Requirements

As an initial proposal, the FPGA to be used in the demonstrator shall include at least the following features:

Programmable System Integration

- Up to 9M system logic cells.
- Up to 16GB in-package HBM DRAM.
- Up to 500Mb of total on-chip integrated memory.
- Integrated 100G Ethernet MAC with KR4 RS-FEC and 150G Interlaken cores.
- Integrated blocks for PCI Express® Gen3 x16 and Gen4 x8.

System Performance

- Up to 38 TOPs of DSP compute performance.
- Up to 128 transceivers operating at 32.75Gb/s or 58Gb/s.
- 2,666Mb/s DDR4 in the mid speed grade.

Power Management

- Energy-efficient machine learning inference.
- Voltage scaling options to tune for performance and power.

A device based on the previous features will be capable of pushing the system performance-per-watt envelope, enabling breakthrough speeds with high utilization. High system performance and multiple power reduction innovations, make this type of FPGA the suitable choice for computing intensive applications. The FPGA shall be chosen also on the capacity to provides scalability and simplicity on package.

2.4.4 External Interface Requirements

FPGAs may have dedicated hardware blocks to implement industry-standard interfaces, which frees the user from having to implement said interfaces via logic units. On the one hand, this saves space, since an ASIC implementation is generally smaller than its logic unit equivalent. On the other hand, the implementation has been tested and its timing has been verified, which shortens the overall design time. For the space use case, external interfaces will be needed for Ethernet and PCIe as described in the section above.

2.4.5 Functionality Requirements

While basic FPGAs might only consist of logic units and block RAM elements, especially mid- and high-end FPGAs have dedicated hardware units tailored for specific tasks. Examples are floating-point units and cryptographic units. Some specific components that may be integrated in the space demonstrator can require specific hardware units: in particular, the vector-accelerator would need 32- & 64-bit floating point units.

2.5 Verification and validation requirements

This will be worked out in D1.3, which will give a refinement, update and additional requirements, especially from the verification and validation point-of-view.

3 Automotive Demonstrator

3.1 Overview of the Demonstrator

Eye detection (see example in Figure 2) is an essential feature for driver monitoring systems acting as a base functionality for other algorithms like attention or drowsiness detection.

Figure 2: An example of eye detection

Multiple methods for eye detection exist. The machine learning based methods involve a manual labeling process in order to generate training and testing datasets. This use case presents an eye detection algorithm based on convolutional neural networks trained using automatically generated ground truth data and proves that we can train very good machine learning models using automatically generated labels. Such an approach reduces the effort needed for manual labeling and data preprocessing and it is applicable in image processing.

The algorithm and the relative processing engines will be implemented on an FPGA based setup and will be validated using laboratory and real in-car environment setup.

Algorithms will be fed with images from cameras with IR image sensors and illumination and the results can be tracked /analyzed on a PC, Intel Core i7-10700, 16 GB, Intel UHD Graphics 630, 512 GB, M.2 PCIe as depicted in Figure 3 and Figure 4.

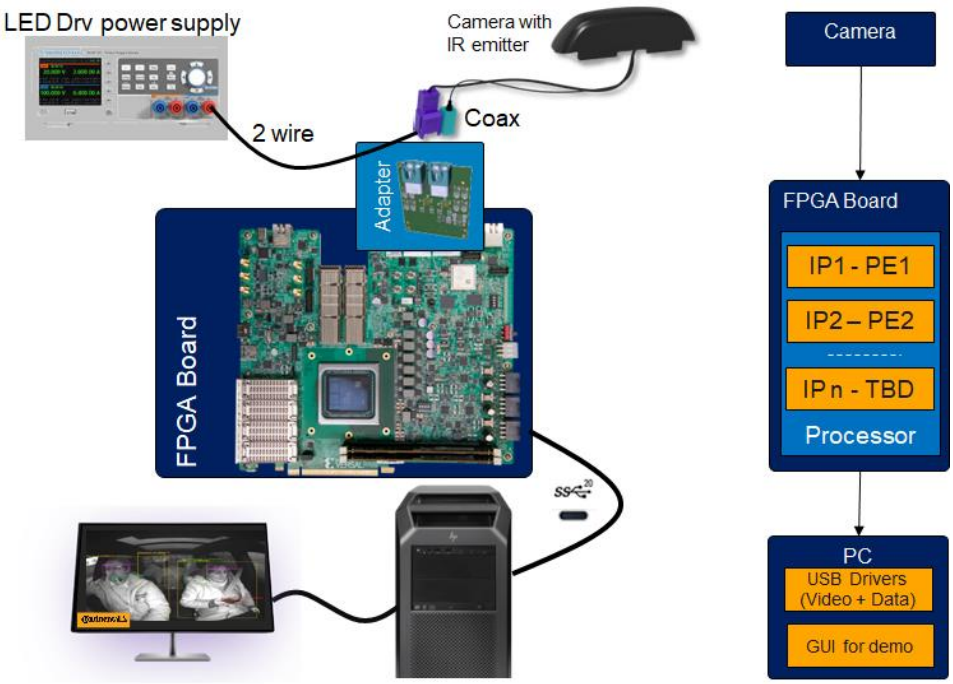

Figure 3: Block diagram of the demonstrator setup

Figure 4: Possible position of the camera in a car

3.1.1 Description of the automotive processor

The automotive application (in our case Driver Monitoring App) can be decomposed in:

- Compute-intensive modules – e.g., machine learning inference, digital signal processing, cryptography.
- Regular modules, i.e., any module which doesn't belong to the above mentioned category (a.o. simple MCU's, ARM Mx (M4) based)

Given the such a taxonomy, the compute-intensive modules shall be executed on dedicated hardware accelerators, while the regular modules will get executed by general purpose processor (GPP).

This approach leads to a heterogenous hardware setup (GPP, hardware accelerators, network to interconnect them) which can be a challenge to be programmed.

In our proposal, we hide the hardware complexity behind unified RISC-V ISA modules (both standard and custom) while providing a holistic view of the software world.

This approach also needs of a software toolchain (compiler, assembler, etc.) to translate the application into a mix of custom RISC-V instructions intended for the hardware accelerators and RISC-V vanilla instructions. The overall view is depicted in Figure 5.

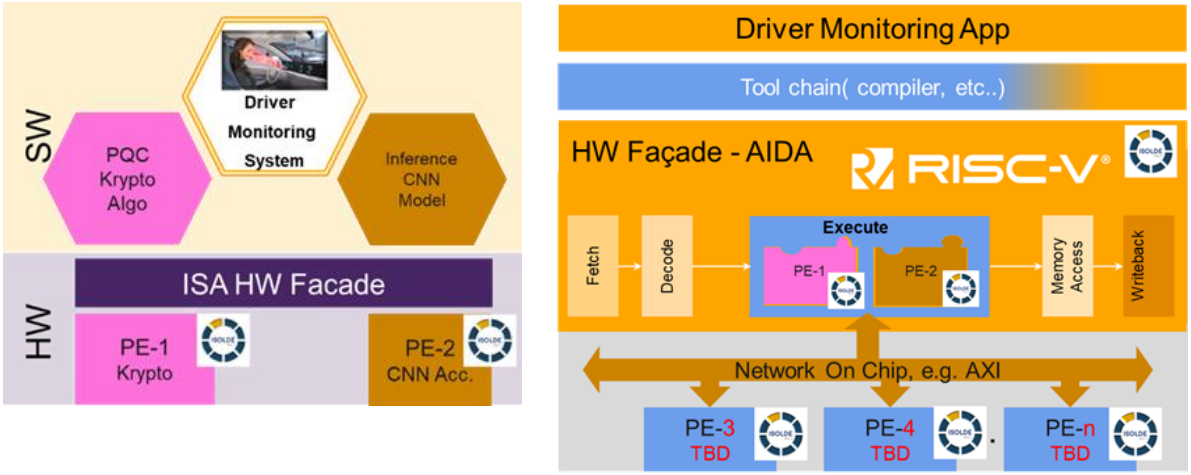

Figure 5: Software toolchain overview

Figure content:

AIDA - Application-specific RISC-V Hardware Accelerator

defines a higher-level ISA which makes the HW accelerators easier to use from the SW perspective

PE = Processing Engines, a.k.a. HW accelerators

PQC = Post Quantum Cryptography

Software toolchain will be able to translate ONNX models/C++ applications into binary code to be executed by AIDA

Development effort will be concentrated in:

- ONNX Front-end
- RISC-V backend

while leveraging all the optimizations already available in LLVM expressed graphical in Figure 6.

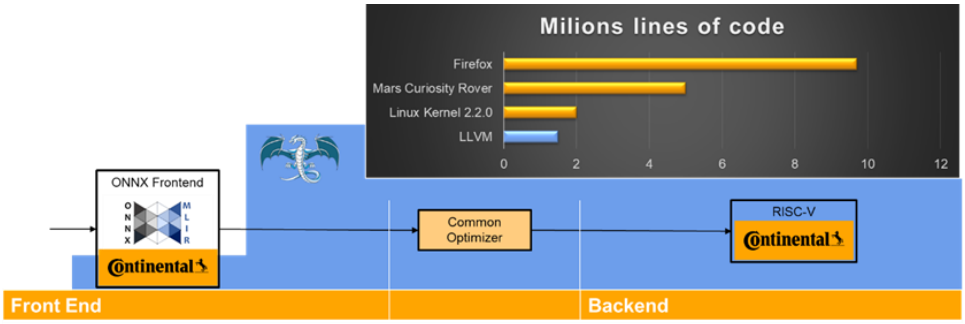

Figure 6: Effort comparison (Code Length) for different implementations

3.1.2 Functionalities and capabilities

To perform eye detection in real-time, a system of multiple Convolutional Neural Networks (CNN) will be used. The models are trained using ground truth data generated using a fully automated solution which does not require any type of manual labeling or intervention.

The system will contain between 2 and 6 CNN models, each one of them responsible for simplification of the task to output the final location of the eyes.

The first CNN of the system will be a regression model responsible for computing an initial location of both eyes (2 bounding boxes).

The last CNN will be a classification model that will be used combined with a small searching mechanism in order to fix the initial bounding box in the correct position.

Between the previously mentioned, other CNN models may be used to reduce and simplify the task of the final classification model. This will make the execution of the entire system faster from a software point of view together with the hardware accelerator.

3.1.3 Key features and benefits

Key benefits for DMS:

- Best performance/watt for an in-cabin driver monitoring system.
- Exploration for HW solutions for in-cabin sensing.

3.2 Functional and Non-Functional Requirements

3.2.1 Cores

In the HW Facade we aim to modify the core pipeline, therefore we will probably use NOEL-V.

Stretch goal: benchmark the several HW Facade implementations (listed in priority order):

- NOEL-V
- In-house HLS RV32I core (RISC-V core implemented in C++)
- CVA6

3.2.2 Accelerators

We will analyze as many as possible the use of such accelerators provided by the following partners:

- UZL: RISC-V Floating-Point accelerator.
- XPERI: AI/ML accelerator IP.
- POLIMI: Customizable floating point arithmetic unit for mixed-precision computing. Configurable hardware accelerator for the BIKE post-quantum key encapsulation mechanism.
- UNIBO: Tensor Processing Unit.
- POLITO: Hardware accelerators for parallel processing.
- TUI: vector/SIMD units.
- SAL: Event-Based / Sparse convolution accelerator.
- SAL: Resource efficient PQC.
- BSC: PQC accelerator implementing Classic McEliece, Crystals Dilithium and Crystals Kyber.
- UNSTPB: XPU.
- HM: Bytecode-Interpreter Accelerator for portability and strong isolation

3.2.3 Software

We will use the following compilers:

Procedural compiler: The LLVM Compiler Infrastructure

ML compiler: onnx-mlir

3.2.4 Other requirements

N/A

3.3 Measurable metrics

The following metrics will be investigated:

- Peak Performance (GOPS/s)
- Peak Power Consumption(W)

3.3.1 Performance Requirements

We aim at throughput: 30fps@1280x720

3.3.2 Power Requirements

Power estimation due to running the demonstrator is bellow 5W

3.3.3 Thermal Requirements

N/A

3.3.4 Radiation Requirements

N/A

3.3.5 Other Requirements

N/A

3.4 FPGA Prototype Requirements

The Continental team owns Xilinx boards as ZCU 102/104. As a possible enhancement, we target the VMK180, and, if feasible, a Xilinx Versal board. As a back-up solution, there is the VCU118 board.

Verilog and System Verilog are the preferred HDL languages; however HLS (C++ and Vitis) represent the main used language and environment.

3.4.1 Memory

Our demonstrator does not require any special memory request, as the on-board memory is good enough for our implementations.

3.4.2 I/O

Camera interface: MIPI/CSI

10 GPIOs for debug (other than JTAG) and control

High speed USB 3.x to interface the FPGA board to a PC for the evaluation of the algorithm results and of the image recording.

3.5 Verification and validation requirements

This will be worked out in D1.3, which will give a refinement, update and additional requirements, especially from a verification and validation point-of-view. Some initial views are given below.

3.5.1 Requirements traceability

Minimalistic for this proof of concept.

3.5.2 Verification plan

Bring-up of the system setup.

3.5.3 Validation plan

Validate the results against CUDA implementation running on GPUs.

4 Smart Home Demonstrator

4.1 Overview of the Demonstrator

The goal is to show the usability of RISC-V application platforms for an SME in the energy sector. Figure 7 describes the setting for smart home energy management demonstrator.

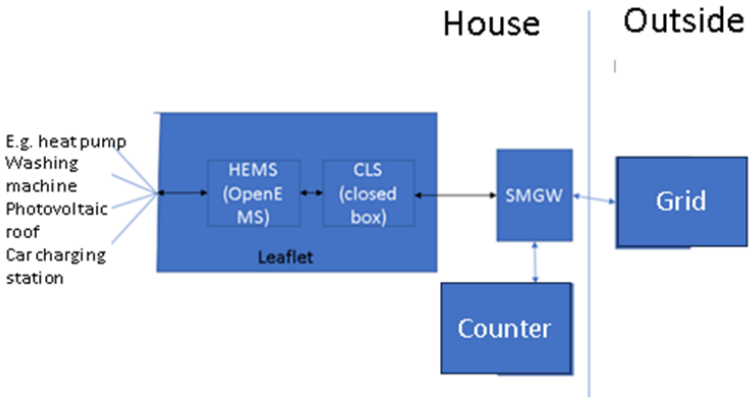

Figure 7: Setup of the demonstrator

The application is an end consumer house network in which various electrical producers (e.g., PV on the roof and/or balcony module, wind turbine, etc.) and consumers (e.g., heat pump, heating rod, charging column for electric car, washing machine, light, etc.) communicate with the Leaflet device (see Figure 7). The goal is to optimize the local use of the generated energy using prognosis, e.g., the charging column battery, operation of the heat pump, are operated when electricity is provided from your own PV system. The decisions / optimizations are made here by a HEMS (Home Energy Management System), in our case based on OpenEMS (https://openems.io/), which is located on the Leaflet device developed by Consolinno. Since it is not cost-effective to strive for complete self -sufficiency in the event of a home network, there is still a connection to the public power grid via the supply system operator. In detail, the house network is separated from the supply system operator by a smart meter gateway (SMGW), which also accesses the counter. The SMGW communicates (e.g., via IEC 61850 protocol) with that of the Leaflet device, which also has a CLS unit (Controllable Local System).

The HEMS unit in Figure 7 operates on the knowledge of all connected devices in the system, while the CLS unit only must know the aggregated value of the generation and consumption in order to communicate, whether fed or consumed, and in what amount this happens.

4.1.1 Description of the smart home processor

We target three implementation stacks, one on i.MX8 (ARM), two on RISC-V CVA-6, preferably multi-core. Concerning the two RISC-V CVA-6 stacks, one is a pure open source demonstrator that can be made fully available (e.g., on Github) as sample payload and the other CVA-6 stack is a demonstrator using a special kind of small code-size hypervisor (separation kernel, https://en.wikipedia.org/wiki/Separation_kernel), which is good at providing strong isolation of execution environments, and controlled execution between them. The separation kernel provided in this use case is called PikeOS (https://www.pikeos.com), and amongst others, has been evaluated against the security standard Common Criteria EAL5 (https://www.bsi.bund.de/SharedDocs/Zertifikate_CC/CC/Betriebssysteme/1146.html). The three targeted implementation stacks are shown in Figure 8.

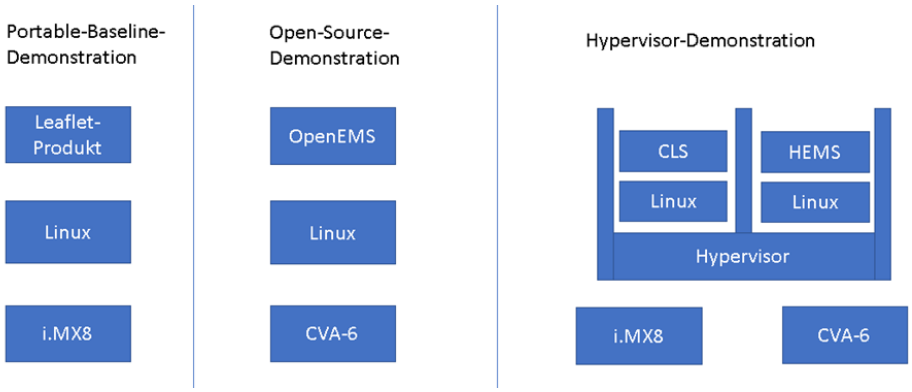

Figure 8: The three implementation stacks covered in this use case

4.1.2 Functionalities and capabilities

CVA-6 shall be used to run a demonstration of the OpenEMS energy management system. CVA-6 shall be able to run mixed-critical systems, such as the PikeOS hypervisor.

We want to run prediction and/or optimization algorithms for energy management, that could be either classical or AI-enhanced (e.g., spiking neural networks, we will also look at applicability of tools like Apache StreamPipes).

Extensions to be developed for Apache StreamPipes include lightweight adapters for smart home applications, e.g., KNX (https://www.knx.org/) and BACNet (https://bacnet.org/), and the evaluation of lightweight adapters in RISC-V architectures. These will be implemented in form of an edge client, which will also provide an execution environment for the afore-mentioned ML algorithms.

4.1.3 Key features and benefits

Our demonstrator pursues the following goals:

- Demonstration that a RISC-V platform is used to carry out the HEMS. The aim here is to be ready to get ready as soon as a European RISC V-application processor has reached the critical mass for a tape-out, or to contribute with our use case to generate this critical mass (open source demonstration, see Section 5.1.1).
- Demonstration that it is possible for an SME to develop at the same time with target platforms i.MX 8 and RISC-V (portable baseline demonstration, see Section 5.1.1).
- Demonstration that through virtualization is possible to develop a portable mixed-critical system with possible target platforms i.MX8 and CVA-6 (see Section 5.1.1 hypervisor demonstration). This goal is attractive because it enables device consolidation. This virtualization is motivated to comply with a possible need for a strong separation of zone boundaries, such as defined e.g. in BSI-TR-03109 (https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03109/TR-03109_node.html).
- Demonstration that Apache StreamPipes will be able to (pre-)process data in a lightweight RISC-V environment and to evaluate machine learning models using a developed edge client (developed in WP4).

4.2 Functional and Non-Functional Requirements

4.2.1 Cores

The demonstrator shall run on a general-purpose CVA-6, ideally multicore.

4.2.2 Accelerators

If available and if the integration is feasible, we want to make use of AI acceleration. This is an option, not a hard requirement.

4.2.3 Software

We want to use a stack consisting of Linux, Java, OpenEMS as well as a version with virtualization consisting of PikeOS, Linux, Java, and OpenEMS.

4.2.4 Other requirements

N.A.

4.3 Measurable metrics

Hereinafter are described the measurable metrics and goals of the smart home demonstrator. Because it is hard to clearly define the measurable metrics at this point in time, it may still change during the project within the use case itself.

4.3.1 Performance Requirements

The smart home demonstrator shall run on the Genesys2 FPGA development board (https://digilent.com/reference/programmable-logic/genesys-2/start) and shall show the online running of the OpenEMS energy management system.

4.3.2 Power Requirements

Currently not defined.

4.3.3 Thermal Requirements

The demonstrator shall run at room temperature (10°C - 40°C).

4.3.4 Radiation Requirements

No radiation hardness is needed.

4.3.5 Other Requirements

None

4.4 FPGA Prototype Requirements

Hereinafter are described the requirements that will be used to select the FPGA for the integration of the demonstrator in WP5.

4.4.1 FPGA

For single core we will use Genesys2 (https://digilent.com/reference/programmable-logic/genesys-2/start) for the implementation of CVA-6. For the multicore the analysis is still in progress.

4.4.2 Memory

We envision 512 MB of RAM, 8 GB of non-volatile FLASH memory.

4.4.3 I/O

For the simulation, e.g., via serial interface, we need a console interaction with the system (e.g., Linux terminal).

4.5 Verification and validation requirements

It is planned to validate the demonstrator by producing a prototype on FPGA and to run selected OpenEMS scenarios/benchmarks.

4.5.1 Requirements traceability

There is no planned special provision for requirements traceability.

4.5.2 Verification plan

There is no verification plan beyond running demonstration scenarios.

4.5.3 Validation plan

It is planned to juxtapose results with regards to the chosen OpenEMS scenarios on the RISC-V platform to functionality of the i.MX8 platform.

5 Cellular IoT Demonstrator

The cellular IoT demonstrator (cIoT) aims to combine 5G cellular IoT standards for wireless communication with embedded HPC to build a modem that can connect to the internet through the cellular network, and at the same time offering enough compute capabilities for embedded applications. Target markets of the proposed system are wearables, industrial monitoring, environmental sensing and monitoring, smart-city, and connectivity in automotives.

The HPC part will be tackled with a powerful vector engine attached to a 64b RISC-V host processor, while the real-time constraints of cellular communication protocols are addressed with tightly coupled, dedicated accelerators in a communication subsystem. For cellular IoT (cIoT, AKA machine-type communications MTC) applications there are the additional requirements of low power (10 years of battery life), low cost (SoC integration) and extended coverage (high sensitivity).

Throughout the project, the partners involved will work towards a 22nm CMOS prototype and demonstrate functionalities on a FPGA platform.

5.1 Overview of the Demonstrator

The demonstrator consists of two main blocks, an embedded HPC subsystem that is handling general high-performance tasks such as image and video processing, and a communication subsystem that is specialized for digital baseband operations and allows for wireless connectivity through the cellular mobile network.

The embedded HPC subsystem consists of a 64b host processor, a powerful vector unit, and an IOMMU that will allow to offload communication tasks to the communication subsystem.

The communication subsystem on the other hand will consist of smaller cores that run real-time critical tasks, including protocol software for modern IoT protocols such as LTE Cat1Bis, the control of a set of accelerators, and the RF-subsystem.

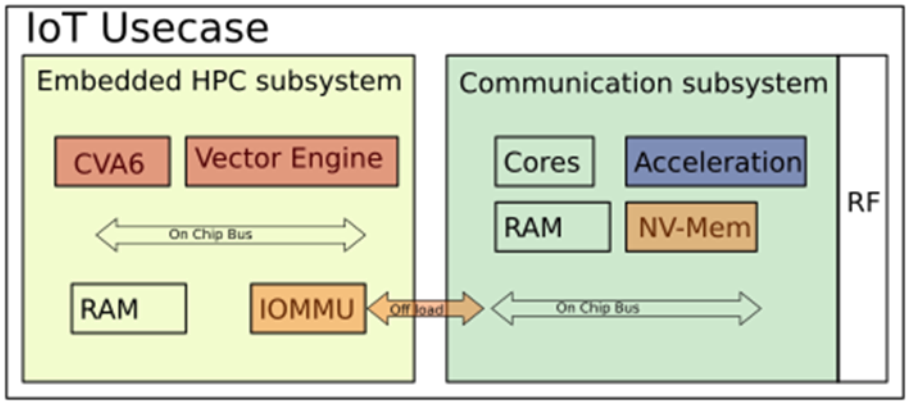

Figure 9.: Cellular IoT use case diagram with contributions from WP2 (red), WP3 (blue), and WP5 (orange).

5.1.1 Description of the IoT processor

The processor of the embedded HPC subsystem shall be a 64b core, with a tightly coupled vector engine. Both components will be developed as part of WP2.

On the communication subsystem a set of smaller cores will be used to build a multi-core subsystem that allows for concurrent execution of many software tasks, and control of multiple accelerators.

5.1.2 Functionalities and capabilities

The cIoT demonstrator system has two main functionalities:

- Wireless communication through LTE Cat1Bis, NR
  - The system must be capable of handling all real-time constraints of the protocol software.
- HPC capabilities to meet performance requirements for wearables, drones, etc.
  - The system must be capable of processing acquired data and transmitting it over the communication subsystem.

5.1.3 Key features and benefits

While the requirements on the communication subsystem are given by the cellular protocol, the requirements on the HPC subsystem differ from application to application. Splitting the system in two parts will allow to scale the complexity of the HPC system to the applications requirements, and at the same time will not interfere with the real-time requirements of the communication subsystem. An efficient way of offloading data to the communication subsystem is required.

5.2 Functional and Non-Functional Requirements

5.2.1 Cores

The demonstrator will use a 64b RISC-V architecture, likely the CVA6 core.

5.2.2 Accelerators

To meet the real-time requirements given by the cellular communication protocol, a set of accelerators, that are developed in WP3 are used. The most performance critical tasks are synchronization, equalization, and channel decoding. The accelerators will be connected to a tightly coupled data memory and will be controlled by a RISC-V multicore system that is running the protocol software.

5.2.3 Software

The HPC system will likely run a Linux distribution and the communication subsystem will run a real-time OS.

5.2.4 Other requirements

Since wearables are one target application of the proposed system, low power consumption is required when fabricated. In particular, a very low deep sleep power consumption is important to allow for a long-lasting battery. Furthermore, to decrease the cost and size of the IC, while guaranteeing a certain level of security, it is desirable to have nonvolatile on chip memory for code storage as well as a file system.

5.3 Measurable metrics

Hereinafter are described the measure metrics and goals of the cIoT demonstrator.

5.3.1 Performance requirements

The throughput requirements of Cat1Bis shall be fulfilled:

- 10mbps downlink
- 5mbps uplink

5.3.2 Power requirements

Deep sleep power consumption shall be in the order of a few uA.

5.3.3 Other requirements

When integrated in an advanced 22nm technology, the size of the die shall not exceed 1cm^2.

5.4 FPGA Prototype Requirements

Hereinafter are described the requirements that will be used to select the FPGA for the integration of the demonstrator in WP5. The preferred board for the cIoT use case is the Xilinx Versal VMK180 for the following reasons:

5.4.1 Vendor / Toolchain Preferences

ACP has been using Xilinx FPGA boards for 10+ years but has no experience with other vendors. Given the experience and success with previous Xilinx boards, it is preferred to stick with Xilinx boards and the Vivado toolchain.

5.4.2 Programming Language Requirements

The demonstrator will be built on top of existing components that are written in SystemVerilog and VHDL. While SystemVerilog is superior for system design, VHDL has its advantages for arithmetic operations that are used in accelerators. To be able to reuse as many components as possible, multi-language support is a must.

5.4.3 FPGA Requirements

The communication subsystem will require a large amount of LUTs and Block RAMs to implement all the processors, accelerators, memories and RF interface. A first estimate shows that roughly 500k-1M system logic cells are required for the communication subsystem while the rest of the system is likely to require another 200-400k.

In terms of BlockRams the communication subsystem will roughly need 4MB of memory to implement all RAMs, caches and buffers.

The cIoT demonstrator aims to integrate nonvolatile on-chip memory for code and file storage. This is typically not available on FPGA boards but can be emulated with DDR memory that is not powered down.

5.4.4 External Interface Requirements

A set of external interfaces are required. Most notably, 3 GTH transceivers and an FMC connector to interface the RF board. Further, an ethernet adapter, 2 UART interfaces, and a PMOD (or equivalent) are required.

5.4.5 Functionality Requirements

Xilinx MPSoC Development boards come with a processing system (PS), consisting of a multicore ARM core that can run a Linux distribution, and a Programmable Logic (PL) part, that

can be programmed according to the needs of the use case. While the PS is not strictly required for the cIoT use case, it offers useful functionalities for debugging, and tracing.

The Xilinx Versal VMK180 board offers all the desired functionalities and will be used for most of the project. However, sub functionalities can also be demonstrated on smaller, cheaper boards such as the Genesys2 board.

5.5 Verification and validation requirements

5.5.1 Requirements traceability

Currently no requirements for traceability are defined.

5.5.2 Verification plan

Individual components will be verified in RTL simulation and then combined in a system. A set of integration tests will be performed to make sure that all components are correctly integrated.

5.5.3 Validation plan

Will be refined in D1.3.

6 Conclusions

The initial requirements for the demonstrators in the four different application domains have been collected in this deliverable D1.1. The main goal of this document is to have a sound starting basis for further refinement and updates, based on activities in the different WPs composing this project. All partners are fully aware that there are still some open issues and specific needs to be discussed further. For example, some requirements on verification and validation still need to be worked out in more detail. All these updates and refinements will be documented in Deliverable D1.3.