Project number: 826276

CPS4EU

Cyber Physical Systems for Europe

D2.5 Simulation tools and experimental platforms - v1

Reviewer (name – company): A; Dupret (CEA)

Dissemination level: Public

Version	Date	Author (name – company)	Comments
V1.0	2020/11/30	Julien Schmitt (VSORA)	
		Benjamin Candillon (VSORA)	
		Minh-Thuyen Thi (CEA)	
		Siwar Ben Hadj Said (CEA)	
		Michael Boc (CEA)	
		Jean-Baptiste Doré (CEA)	
		Nicola di Pietro (CEA)	
		Florian Greff (Thales TRT)	
		Guillaume Vivier (Sequans)	

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

EXECUTIVE SUMMARY

This document is the first report of task 2.3 of the work package 2 (WP2).

WP2 deals with the communication aspects in cyber physical systems (CPS). Task 2.3 deals more specifically with prototyping of solutions envisaged in the two other tasks. In the first deliverable (D2.1) we presented requirements from other packages (in particular WP6 and WP7) and exposed two main axes of development. The first one consists of a communication system relying on existing technologies: the challenge here is to integrate within a board various communication protocols like 4G communication (LTE), Wi-Fi, Bluetooth, Ethernet. This aggregation is done via the PIARCH board presented in this document.

The second axis of development explores new technologies centred on the 5G standard and its evolution, with a particular interest in the ultra-reliable and low-latency communication (URLLC) feature introduced by this norm. This implies new algorithms for signal synchronization and new solutions for the network-level issues: deliverable D2.3 presents the innovative solutions for CPS communication modules and networking developed by WP2 of CPS4EU.

Regarding the signal processing aspect, the study carried out in Task 2.2 led to the design of a MIMO system using energy detectors for millimeter Wave applications. This module implements advanced signal processing algorithms by using neural networks as well. In the context of Task 2.3, we propose to continue this study until the implementation on a real target: this phase of the development in a commercial project is challenging since new issues like the computation capacity of the real target, the quantization of data and the management of the memory could lead to major modifications in the original algorithm. The communication between different teams working on the project (the DSP team and the implementation team) generates a cycle of development converging to a satisfying solution but this increases the time to market. In the framework of Task 2.3, we are exploring a new way of development where DSP engineers can determine at early stage of development their computing and memory requirements and study the impact of the quantization. Finally, we will run the simulations on a remote FPGA using a new cloud computing offer.

To achieve very low latency, the network is also a key component: we present here the results and the validation process to implement the IEEE Ethernet Time Sensitive Networking (TSN), in particular, the norm IEEE 802.1AS.

The present document is a first release exposing the issues and the protocol of validation on experimental platforms. Preliminary results are also presented but they will be completed in the final version of the document at the end of the CPS4EU project.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Table of content

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

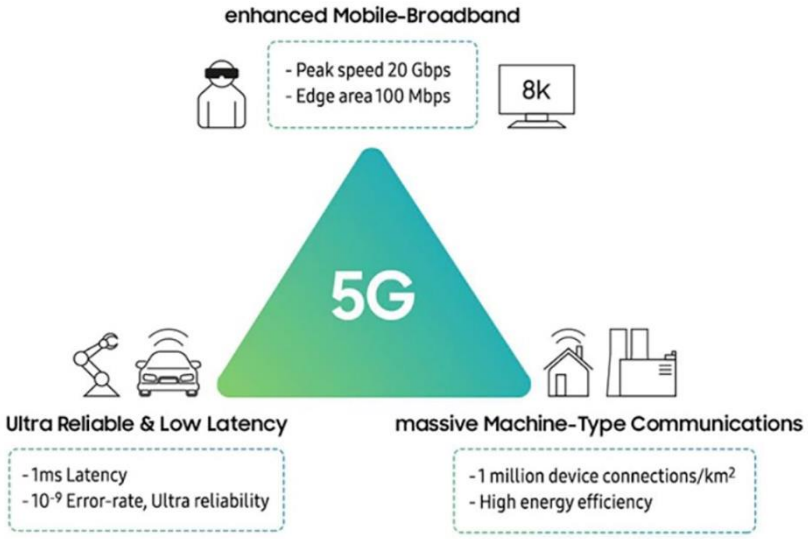
1. INTRODUCTION

1.1. Purpose

The purpose of this document is to present the validation process of the achievements of WP2, which aims to define the communication modules needed for the next generations of Cyber Physical Systems. The first step of the project was an analysis of the requirements coming from verticals. As described in the first deliverable D2.1 “Specification and architecture of the communications modules”, we identified two main families of modules for CPSs: the first one is based on existing technologies; the second one consists of algorithmic and networking solutions that go beyond the state of the art, devised because of the limitations of the first family.

The first family of communication modules that we identified in WP2, is based on well-known technologies: Wi-Fi, Bluetooth, Ethernet, high-performance LTE (cat 4), low-power LTE (Cat M1, Cat NB1). The work presented in D2.1 highlighted the need of a pre-integrated platform: not all requirements (in terms of communication capability, power consumption, security, interfaces …) can be met in a single solution. The platform PIARCH developed in WP6 will use modules from 4G modems. The validation of the system will be done within this group (WP6) and we present here only the board and the preliminary test and validation.

The weak point of the 4G technology integrated in the PIARCH board, is the latency and the reliability of the communication at least for the most extreme use cases of CPS. This point is addressed in a new wave form: the 5G technology.



The 5G technology is proposed by the 3GPP consortium and targets to improve three features:

- Enhance the mobile broadband with peak speed up to 20 Gbps
- Address the IOT devices
- Introduce the Ultra Reliable & Low Latency Communication feature (URLLC)

In the context of the CPS4EU project, we are focusing especially on the last point. In this document, we will not discuss about URLLC-enabling algorithms: the solutions developed by CPS4EU are described in D2.3: “Proposition for 5G, including URLLC evolution” and will be further discussed in D2.4. Here, we will focus on the simulation process and describe how we can simulate some selected innovative solutions on a real DSP. This simulation will be done on different platforms and finally we will run the simulation using a DSP mapped on remote FPGAs.

Moreover, in the complete chain of transmission, efficient networking is also a key factor to meet the latency requirements. In the work of WP2, we address this point by implementing the IEEE time-sensitive networking recommendations, and in particular the norm IEEE 802.1AS. We will present in this report preliminary results about this implementation.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

1.2. Definition, acronyms, and abbreviations

Acronym / abbreviation	Description
3GPP	3rd Generation Partnership Project
AI	Aritificial Intelligence
AGU	Address Generator Unit
ALU	Arithmetic and Logic Unit
API	Application Programming Interface
AWS	Amazon Web Service
BMCA	Best Master Clock Controller
CPS	Cyber-Physical System
CPU	Central Processor Unit
CUC	Centralized User Configuration
DMA	Direct Memory Access
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GCC	GNU Compiler Colection
GM	Grand Master
IMSI	International Mobile Subscriber Identity
LTE	Long Term Evolution
LTE – M	Long Term Evolution – Machine Type Communication
MIMO	Multiple Inputs Multiples Outputs
MTC	Machine Type Communication
PReLU	Parametric Rectified Linear Unit
PCI	Peripheral Component Interconnect
PCIe	PCI express
PPP	Point-to-Point Protocol
PIARCH	Pre-Integrated Architectures
ReLU	Rectified Linear Unit
RTL	Register Transfer Level
SDN	Software-Designed Networking
SIMD	Single Instruction Multiple Data
SSH	Secure Shell
TCM	Tightly Coupled Memory
TLM	Transfer Level Model
TS	Time Synchronization
TSN	Time Sensitive Networking
UART	Universal Asynchronous Receiver Transmitter
URLLC	Ultra-Reliable & Low Latency Communication

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

1.3. List of Figures

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2. EXISTING TECHNOLOGIES (LTE / WI-FI / BT)

2.1. Purpose

The goal of this validation platform is to provide a preliminary proof of concept of the integration, configuration and usage of the connectivity module from WP2, which provides LTE connectivity. This work can then be leveraged in the development of the Secure CPS-to-X Connectivity Pre-Integrated Architecture (SC2XC PIARCH) in WP6, and of use cases (namely WP7).

To give a better idea of the context, Figure 1 shows our reference use case, which will make use of the connectivity module. In this use case, we want to use it to enable remote monitoring and maintenance of a connected and possibly autonomous vehicle, from the cloud. More precisely, it will be used by a secure gateway, at the edge of the vehicle, to periodically send Health and Usage Monitoring System data from the vehicle to the cloud, and occasionally receive Over-The-Air updates from the cloud. Additional protocols such as Bundle Protocol (IRTF DTN Research Group, 2007) will be used to enable disruption-tolerant communications, but this is out of the scope of WP2.

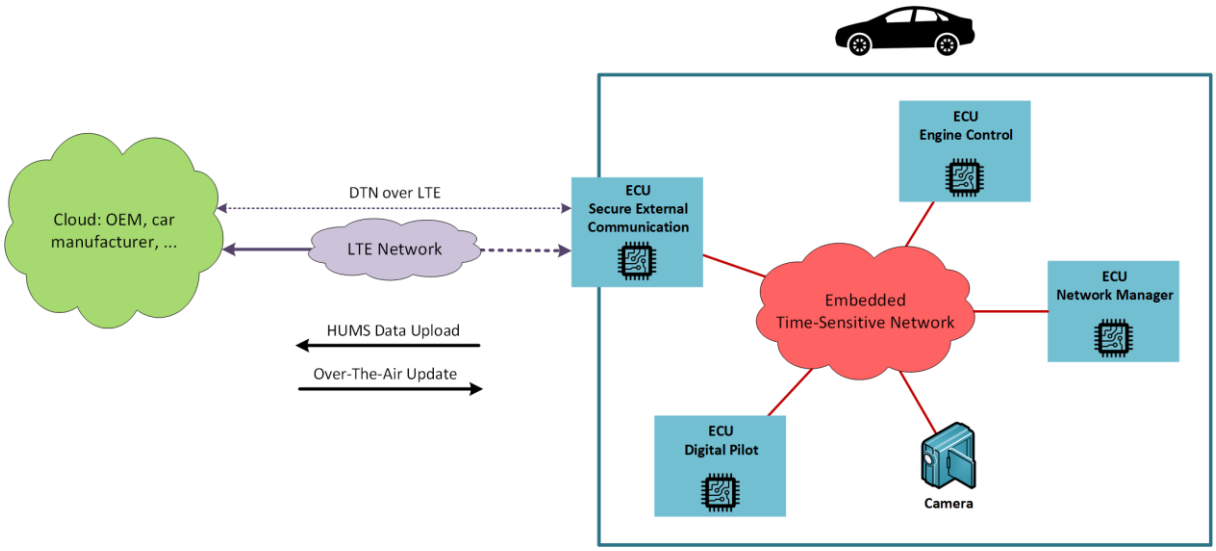


Figure 1 - Safe and secure remote monitoring and maintenance use case

In WP2, we study how we can make use of the connectivity module to send data from a Linux application but also from a microcontroller, to a PC located in a remote network. We also study the configuration and integration possibilities to enable different use cases, in terms of energy consumption, application development, security, and so on.

The knowledge gathered from this work is transferred to WP6 and leveraged in the development of the SC2XC PIARCH. This PIARCH contains all the support to build a secure gateway subsystem for a target use case. It provides LTE, Wi-Fi, Bluetooth/Bluetooth Low Energy and TSN (Time-Sensitive Networking) connectivity. Regarding the connectivity module, it basically consists in “packaging” it to ease its integration and usage, i.e. to facilitate the reproduction of what has been achieved in WP2.

The SC2XC PIARCH will then be used to build the “Safe and secure remote monitoring and maintenance” automotive use case in WP7, as explained above (Figure 1). The PIARCH also enables the use of the connectivity module in other use cases in and beyond CPS4EU.

This is how we aim to transfer WP2 knowledge and components all the way up to use cases and partners projects, as depicted in Figure 2.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

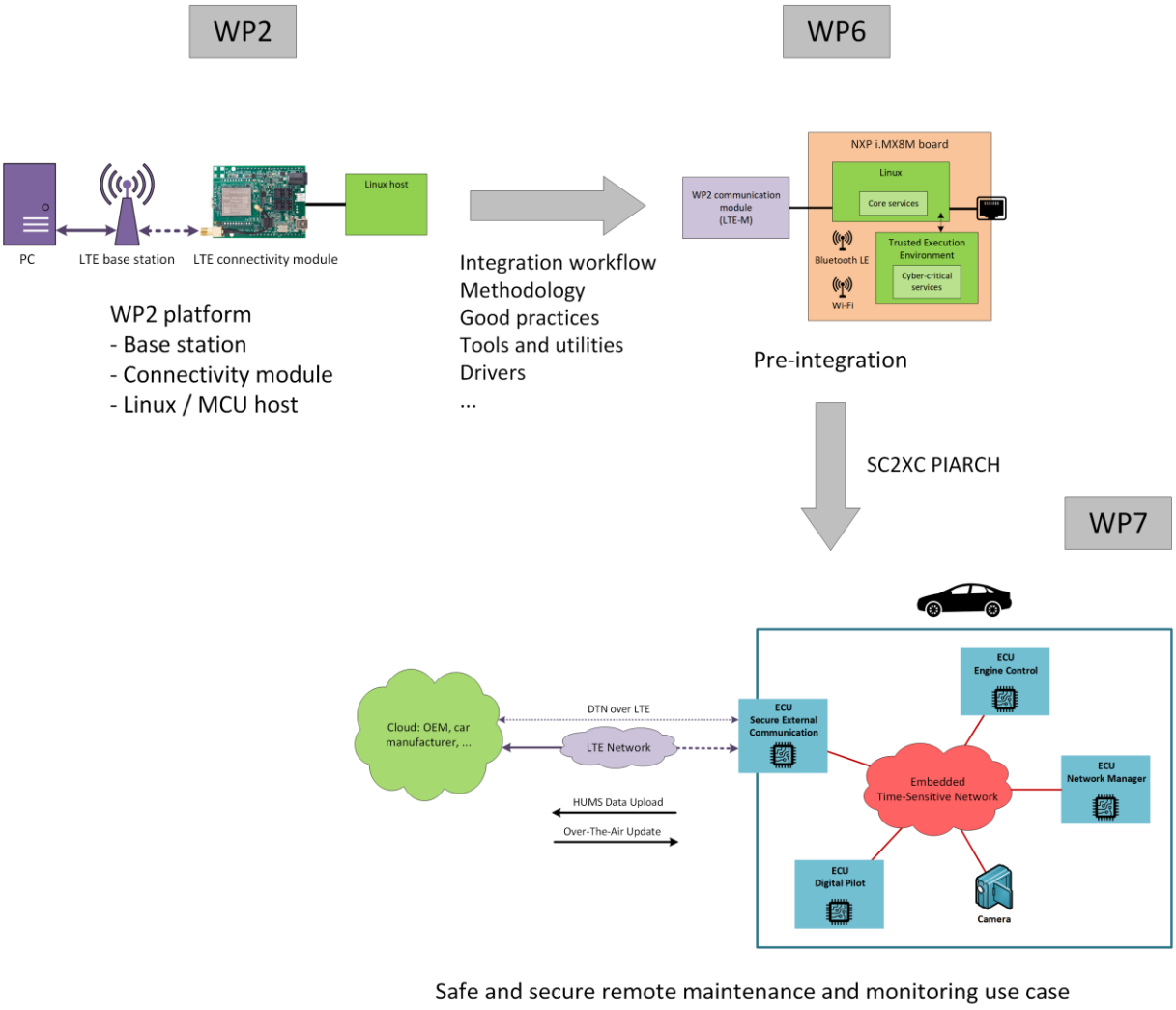


Figure 2 - WP2 module integration in WP6 and WP7

2.2. Technical objectives

The current goal of this validation platform is to experiment and validate the following:

- Configuring the base station and registering the module to the network,
- Sending and receiving data from Linux and, in the future, from a low-power microcontroller,
- Forwarding data from the base station to a core IP network,
- Handling various possible errors, for example when the module has to be reset,
- Configuring the module and base station to make use of LTE-M (LTE Cat-M1) or NB-IoT (LTE-NB1) technologies, depending on the use case,
- Making use of deep sleep mode to reduce power consumption.

Security is not part of the current study, as it is handled by the PIARCH itself, via the use of a Trusted Execution Environment to perform cyber-critical operations.

2.3. Hardware

2.3.1. LTE module

The LTE modem is developed by Sequans. For the pre-integration of PIARCH, it was decided to adapt an existing module, developing some specific feature to the CPS, and to support integration activity. It was decided to consider the Monarch family for the low-end communication requirements (cat-M or NB-IoT).

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Monarch is a single-chip LTE Cat M1/NB1 solution designed specifically for narrowband IoT applications, including sensors, wearables, and other low data, low power M2M and IoT devices. Monarch complies with the ultra-low-power and reduced complexity feature requirements of the 3GPP mMTC, defining narrow-band, low data rate LTE technology for machine-type-communications (MTC). Monarch achieves a very high level of integration whereby baseband, RF transceiver, power management, and RAM memory are integrated into a tiny 6.5 x 8.5 mm package, running Sequans carrier-proven LTE protocol stack, an OMA lightweight M2M (LWM2M) client for over-the-air device management, and a rich set of AT commands.

The Monarch module is a hardware module that can be integrated directly to a customer board. It has been certified in all major operators in the world and supports 17 bands in a single sku.

Figure 3 - Monarch GM01Q module

Since in CPS4EU project, the pre-integrated architecture board is based on a on-the-shelf evaluation board from NXP (NXP I.MX 8M), no specific HW design was foreseen. Therefore, it was decided to move forward with a Monarch evaluation board.

Indeed, there are various derivative of boards build on the Monarch module. Evaluation kits, or extension boards on commonly used MCU. One can mention

- GM01Q-STMOD expansion board for plug-and-play use on STM32 discovery kits
- Nimbelink evaluation kit
- Skyworks integrated SIP (for an even more thin Monarch based module)

For STMOD extension

EVK for Monarch-SIP

Nimbelink EVK

Figure 4 - Several derivative of boards based on Monarch solution

For CPS4EU we considered Nimbelink based EVK, which offers easy control and debug through two separated USB ports, as well as possible integration with Arduino or Raspberry boards.

Continuous improvements of the performance of the module are being done by Sequans, with a particular attention given to power consumption, radio performance, security and reliability, as well as end user ease of use. With that respect, we started to conceive a unified environment that would allow fast replacement of Sequans module, for instance from a cat-M to an NB-IoT one, or even to support a cat-4 or cat-6 module. This would then perfectly fit to the concept of pre-integrated architecture for which pre-integration of various modem category can be done.

More practically, to support the integration into the PARCH, we also generated various documentations.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.3.2. Base station

We use an AMARI OTS 100 (now AMARI Callbox Classic, Figure 5) base station from the French company Amarisoft (https://www.amarisoft.com/, s.d.). It acts as a 3GPP compliant eNB (base station) and EPC, allowing functional and performance testing of LTE (up to Cat. 10), LTE-M and NB-IoT connectivity.

Figure 5 - AMARI Callbox Classic base station

2.3.3. Host

As explained above, we are currently focused on making use of the connectivity module from a Linux host. We use the same board as the one used in WP6, i.e. an NXP i.MX8M-based board. This host is connected to the module's UART via USB (Figure 6). We use this board for the sake of consistency; however, any Linux host could be used to conduct the experiments.

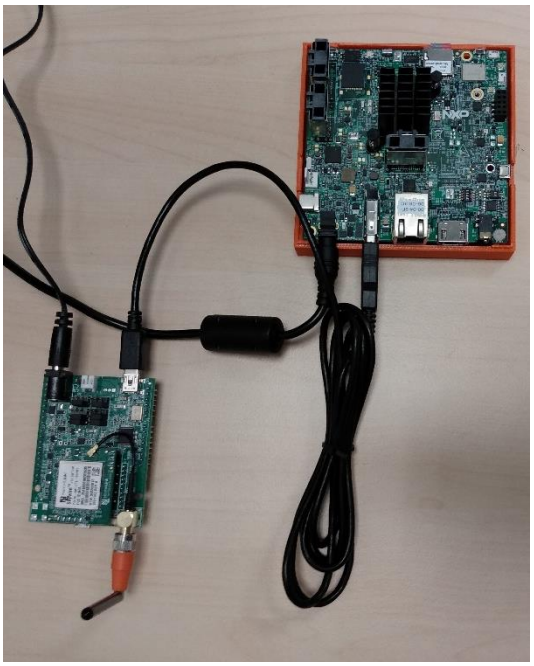


Figure 6 - NXP MCIMX8M-EVKB board and Sequans Monarch GM01Q_EVK connectivity module

In the future, we aim to enable the use of a low-power microcontroller (with another OS such as Zephyr (https://www.zephyrproject.org/, s.d.)) for when energy consumption is at stake. The i.MX8M System-on-Chip provides us with such a microcontroller (ARM Cortex-M4F).

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.3.4. SIM cards

We use configurable SIM cards (USIMs) from the French company Open Cells Project (https://open-cells.com/, s.d.).

2.3.5. Remote PC

This element is normal PC located in the same IP network as the base station. It is used to perform end-to-end communications between the LTE module and a remote IP network / cloud.

2.4. Technical achievements

We have managed so far to register the module to the LTE network, use various AT commands to configure it and send data, as well as mounting an IP interface in Linux to send data via Point-to-Point Protocol (PPP) over UART. This is an overview of the current process:

a. USIMs configuration

Open Cells Project provides a tool to configure the USIMs. We set their operator code to <opc> and their authentication key (Ki) to <key>. We will also use their International Mobile Subscriber Identity (IMSI).

b. Base station configuration

We configure the Amarisoft base station database to accept the target USIM. We provide the authentication algorithm (milenage), IMSI, <key> and <opc>.

c. AT commands from the host

We use the module UART to connect to it from the host and send AT commands. Various AT commands can be used to check the USIM PIN code, put it in airplane mode, enable multi-sim support, and so on. Once the module is ready to connect to the network, we use the following AT command to enable full functionality and try to register to the network:

AT+CFUN=1

Then, we can use the following AT command to check the registration status:

AT+CEREG?

This command must return 1 if we are correctly registered to the home network, or 5 if we are registered in roaming mode. In the case the Mobile Country Code (MCC) and Mobile Network Code (MNC) of the IMSI do not match those of the base station, the module will be registered in roaming mode.

d. Sending data

The module provides TCP/UDP/IP, HTTP(S), FTP, TFTP, MQTT stacks and more, in order to send and fetch data thanks to AT commands. Details on how to make use of these commands are given in the module documentation.

We are also able to mount a Linux IP interface using Point-to-Point Protocol (PPP) over UART. In order to do so, *pppd* (PPP daemon) and *chatscript* utilities must be installed on the host. The PPP options and chatscripts files (*connect* and *disconnect*) must be sent accordingly to the use case. We then run the PPP daemon, *pppd*, in order to mount the *ppp0* interface. This interface can then be used to freely communicate with the base station. Again, more details on this process are given in the module documentation.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

3. SIMULATION OF A MIMO DECODER ON A DSP MODEL

3.1. Presentation of the project

In this section, we will present some initial details and the main features of our testbed implementation of the MIMO decoder for sub-THz xHauling developed in the framework of Task 2.2. The final and detailed results obtained through this experimentation will be reported in D2.6 at the end of the project's lifetime. A full description of the considered MIMO communication problem and the corresponding MIMO signal decoding system are described in D2.3. For the purpose of this document, we just need to know that this system involves an Artificial Intelligence processing. We will focus here on the simulation tools exploited to evaluate the system on a realistic implementation and we will describe the different validation platforms (from native processing to an implementation on a remote FPGA).

We assume that the neural network is trained before implementation and we consider only the inference part of the system. This includes a description of the different layers of the neural network and the weights associated to each layer.

To simulate this application, we have to build a whole project composed of 3 distinct blocks represented in the diagram below:

- An application – blue part
- A system – orange part
- A test bench – green part

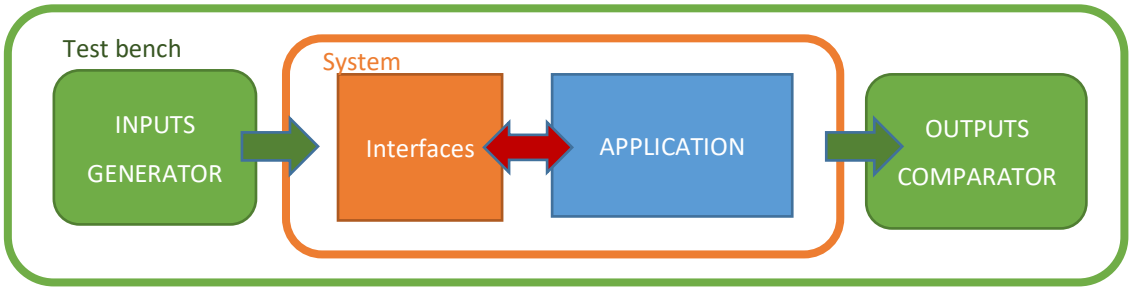


Figure 7 - Whole neural network system simulation

In some cases, the *system* regrouping the *application* and the *interfaces* are not separated. These blocks can be described in different languages (matlab, python, C++, tensorflow, etc …)

Application

The application is the description of the different layers composing the neural network. This is generally expressed in a framework dedicated to AI like Keras, tensorFlow or Caffe. Using these frameworks, we have generally also the weights (coefficients computed by the training process) which belong in the previous diagram to the interface level. This description can be executed with all kinds of layers (convolution2d, fully connected, *matmul*, various activation functions (ReLU, ReLU6, PReLU, sigmoid) …).

The description of the layers can also be expressed using VSORA's library. This C++ library contains the most used layers in various public projects (among supported project by VSORA, we can mention imageNet (image classification), YOLO (objects detection), DeepSpeech (speech recognition)).

The use of VSORA's inference library is mandatory to compile the application on a VSORA's DSP. Nevertheless, we do not have to write the code in C++ using this library: as mentioned previously, most projects are written within a framework. VSORA developed a compiler (called "*graph compiler*") which accepts an entry point expressed with these languages and transforms it into a code using the VSORA's API. In the CPS4EU project context, the framework used is tensorFlow and the entry points are called "pb-files".

The graph compiler is not just a translator from one language (here the tensorFlow language) to C++ (with the inference library). Its separates also the weights included in the pb files from the layers description. In some cases, it can also move or merge some layers to take the full benefit of the target.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

System

The system level contains the application and the interface. As mentioned previously, when using a native framework, the system level is merged with the application level. There are 2 main reasons to make a distinction between the system and the application and both are relevant only when compiling on a lower target (ie not in native mode):

- The first reason is that the application and the interfaces are not executed by the same entities. The application is executed by the DSP – Core (the SIMD part of the DSP) to take the advantage of the full processing capacity. The interfaces are managed typically by DMAs or by the host processor. We will detail more precisely the architecture of the DSP in a next paragraph.
- The second reason is that the interfaces support also the memory management. This point is not obvious when running the project on a PC since the memory is almost infinite (from the application's point of view). On a real target, memory management is essential. The DSP-Core contains a local memory (TCM) in small quantity. This memory has a low density and its cost in the final silicon is significant. That's why data and weights are stored in an external DRAM (dense memory) and we format and load them dynamically into the DSP core's TCM using a dedicated process.

The graph compiler generates the interface expressed in python. Python is a language widely used in the AI ecosystem. It has a large range of external libraries which make it easy to use.

Test bench

The test bench is the final and highest level in the project. It contains the system with additional blocks. This level is dedicated to the simulation and does not exist in the real life. Typically we will instantiate a data generator which will feed the system with data to process. At the end of the processing chain, we can analyze the results.

The test bench can be expressed in various languages. In the context of the CPS project, we will use the Matlab environment. Matlab is the generic tools for mathematical applications. It is easy to use and we can manipulate data at a high level (the matrix container is typically built-in).

The data generator can also have various implementation: it can consist simply of a data reader from a file. This is generally the first step since it is easy to implement and we can have a full control on the generated data. However, its uses is limited and cannot manage complex scenarios; it can be replaced then by a true algorithm generating data on the fly.

3.2. Test bench description

As mentioned above, the considered MIMO system model is described in deliverable D2.3 [*Propositions for 5G, including URLLC evolution*] and the test bench focuses on the MIMO detector presented therein. In particular, a Matlab model will generate a signal from a bit stream multiplexed on N antenna as depicted in Figure 8. The noisy signal is received on the N Rx antenna. A baseband conversion is made in Matlab. A set of received samples are then forwarded to the VSORA platform that instantiates the proposed neural-network MIMO decoder. It should be noticed that the neural network is trained offline.

First, the output of the VSORA platform will be compared to the output of a high-performance Keras model of the same decoder with large quantization of the processor. This will validate the flow. Then, an optimization of the architecture will be proposed. It will then be possible to estimate the performance loss (Bit Error Rate) introduced by the internal quantization of the processor and, more generally speaking, by the implementation choices made.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

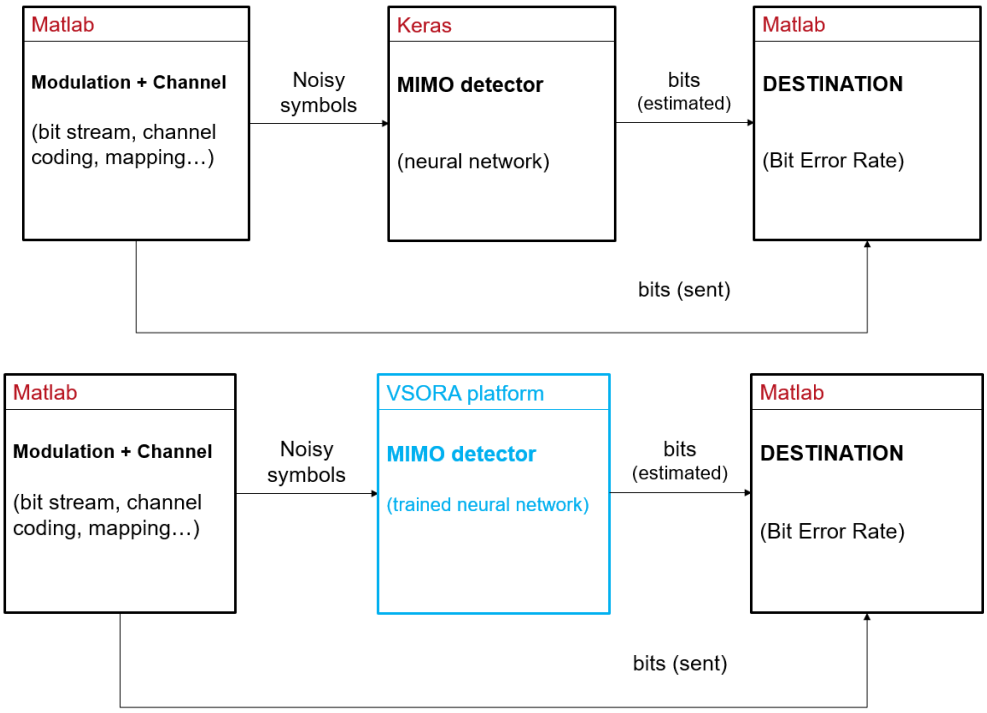


Figure 8 - Illustration of the test bench

3.3. VSORA DSP Architecture

DSP macro description

The DSP developed by VSORA belongs to the family of DSP called SIMD. Multiple data are processed in parallel within Base Units (BUs) which are executing the same instruction (or suit of instructions) given by a sequencer. Address Generator Units (AGU) compute addresses of data to be read or written in the local memory (TCM). Additional DMAs give access to the TCM from outside. These blocks compose the part of the DSP called "*core*". This part is totally slaved by a host processor. The host processor communicates with the core through a mailbox.

The number of base units and the number of DMAs are parameters of the static configuration of the DSP. The more there base units are instantiated in the core, the higher the processing capability will be.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

The quantization of the data, which has an impact on the memory footprint, is also a built-in parameter of the ALU. The quantization follows the IEEE-754 standard with a configurable (static) number of bits for the mantissa and for the exponent. The quantization is a key parameter in the study of the implementation of the algorithm on a real target.

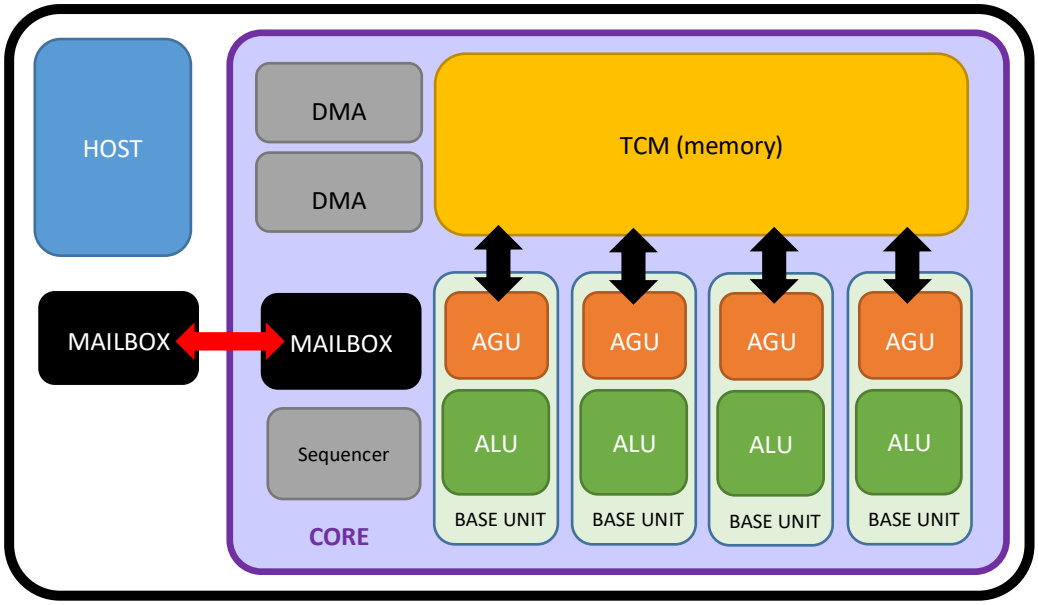


Figure 9 - VSORA's DSP architecture

The Host processor is a commercial processor with its own operating system. The single constraint is to be supported by the LLVM compiler which is the case for the most popular processors. It can be replaced in a first step by the CPU of user's PC, and the operating system by Linux. This allows to focus on the processing study (processing capacity and memory needs, quantization). The CPS4EU project is restricted to this area.

Compilation flow

The philosophy of VSORA's development flow is to keep the same code at all stages of development. All simulation platforms (see paragraph 3.4) run the same code, but compiled with different options.

The graph compiler has a tensorFlow code as entry point and generates a C++ files calling the API of the inference library. This library is developed by VSORA.

The LLVM based VSORA compiler has a C++ code as entry point and generates a binary file of the application for the host processor. In this code, functions running on the core (parallel processing) are extracted and replaced by a message (send through the mailbox to the core) (this takes the form a function call).

On the core side, the code of the vector math functions library is expressed in C and in a language developed by VSORA to express the parallelism and functionalities of the core ("vs-code"). This code is compiled by GCC (well know compiler in the Linux community) and by a compiler developed by VSORA; these binary codes are stored in various locations in the core.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

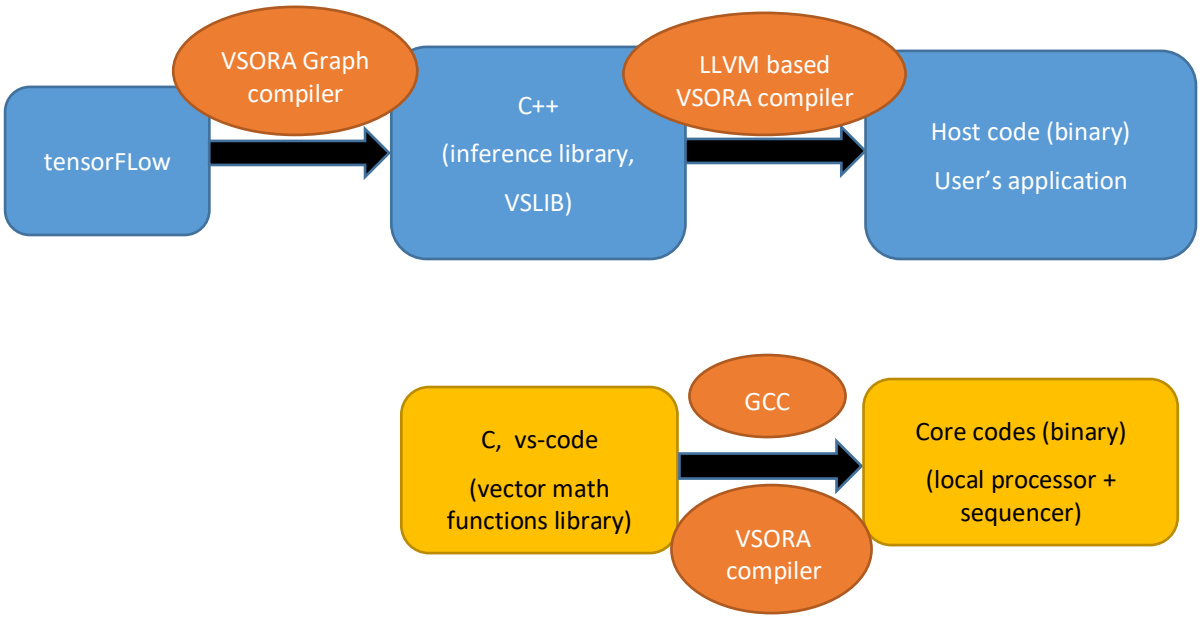


Figure 10 - VSORA's DSP Compilation flow

3.4. Simulation platforms

VSORA's development process is organized around simulation platforms. These platforms represent different levels of the DSP model.

- **Native platform**

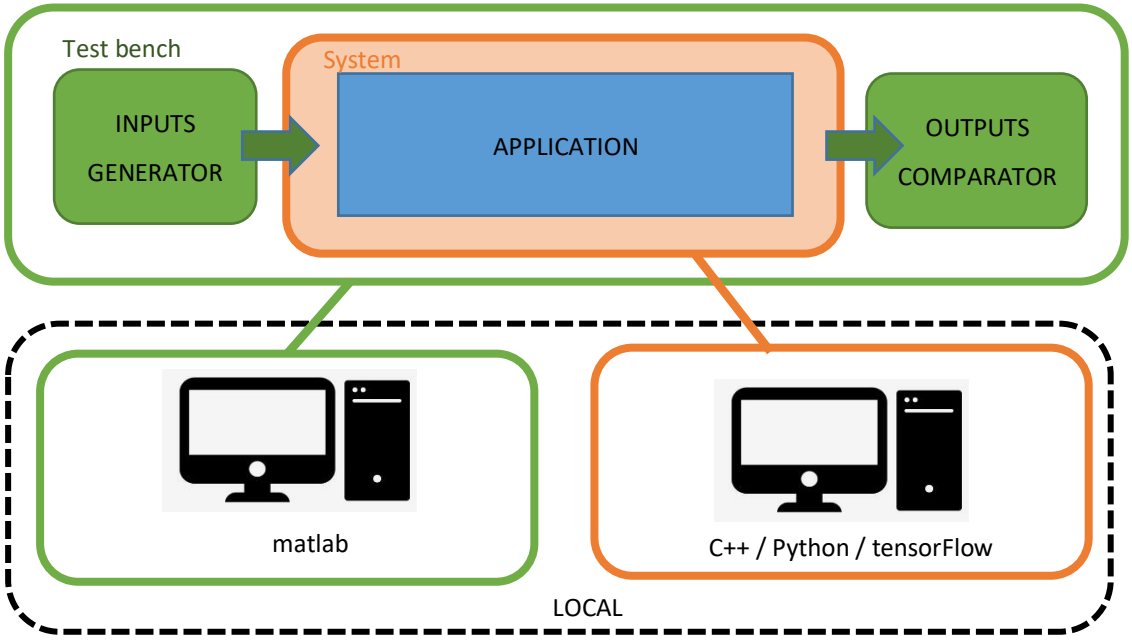


Figure 11 - Native simulation

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

The native platform is the highest level of representation of the system. At this level, we do not have the notion of DSP and the main focus is the application and the algorithm study. The environment simulation is the user's PC or servers.

The application / system must be described using tensorFlow or VSORA's library and encapsulated in a top written with python. The test bench is described with matlab. The communication between matlab's process and the system's process is done with existing communication libraries (python).

This platform is used to develop the algorithm: high abstraction of computation, fast simulation.

- **High Level Platform**

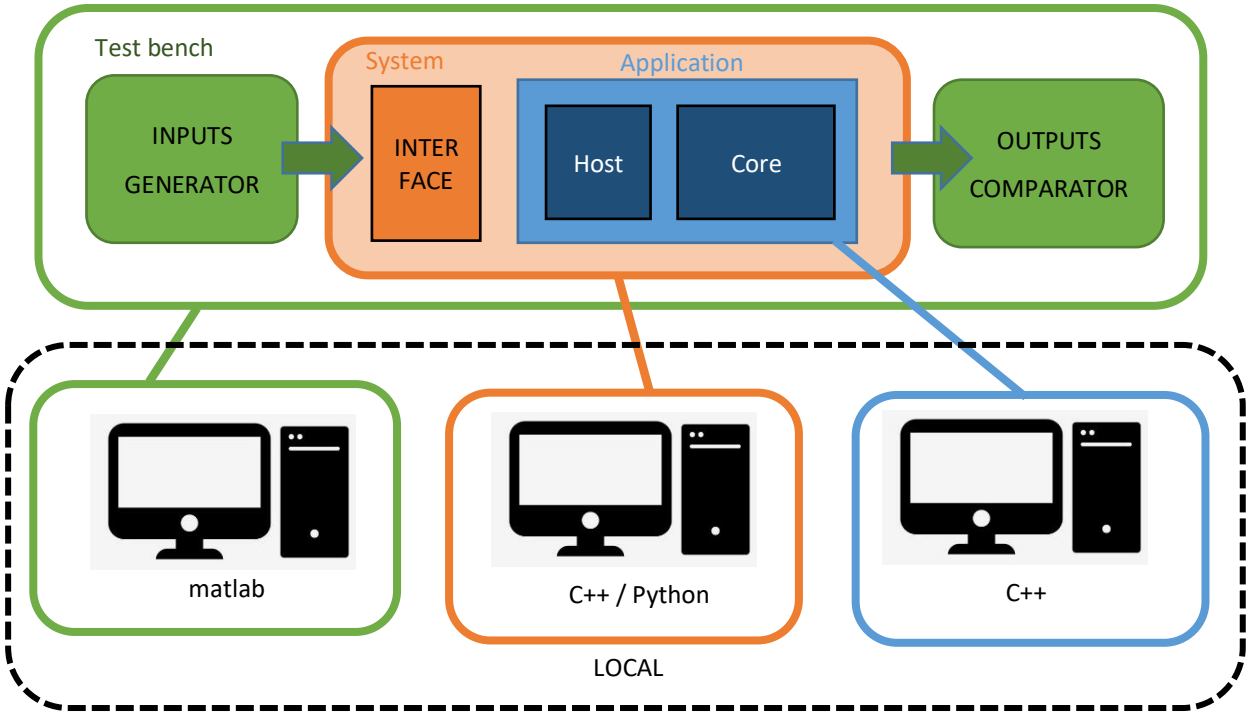


Figure 12 - High level simulation

The high level platform is the first level where the DSP appears. This level allows to validate the compilation process (code transformation, isolation of the core from the host processing).

At this level, the application is compiled with the graph compiler and with the vsora/llvm compiler: the interfaces are isolated, the code running on the core is isolated from the core running on the host. Theses codes are executed within processes and executed locally on a PC.

This platform generates several reports: we can have an estimation of the number of cycles of the simulation, and the peak memory usage. This level of simulations is well suited to evaluate the impact of the quantization and to determine the architecture needed to run the application under real time constraints.

- **Low Level Platform / RTL platform**

The low level and RTL platforms are similar to the high level platform: the differences appears in the models of the DSP instantiated by the application: these models become more precise in the description of the different components: we get more accurate simulations (in term of number of cycles) but the simulation's durations increase dramatically.

In this level, we can also instantiate a TLM or RTL model of the host to have a complete model of the DSP.

These platforms are not used in the CPS4EU project: we will not give more information about them.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

- **FPGA Platform**

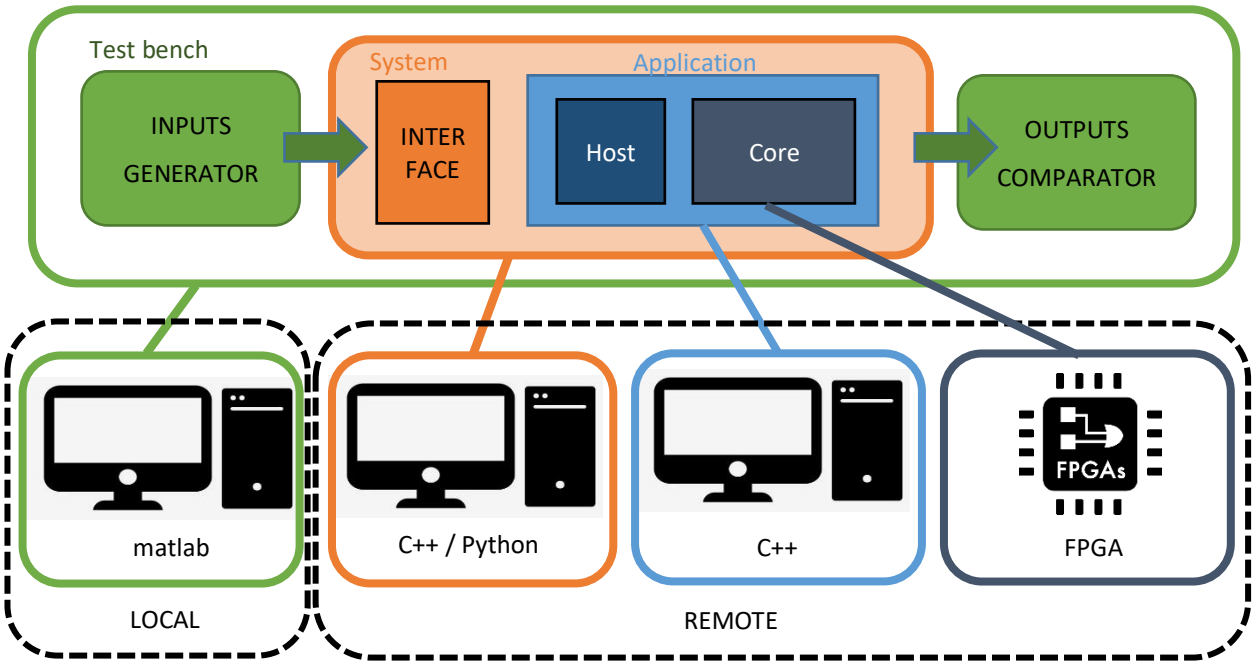


Figure 13 - FPGA simulation

The FPGA platform is the lowest platform of development. The model of the core is mapped in a FPGA. The system is run on remote mother board neighboring the FPGA while the test bench is run locally: matlab is a software under commercial license and cannot be exported remotely.

The FPGA has limited capacity: the core can contain up to 2 Base Units and the memory of the TCM is limited as well. Thus, we could not be able to run the exact configuration determined at the high level simulation. However, this simulation is still relevant: it is a true core system (not only estimation) with real memory conflicts. The simulations are faster than a RTL simulation (giving the same accuracy in terms of processing cycles).

3.5. Remote FPGA infrastructure

Description

VSORA has chosen to use the cloud computing platform from Amazon, called Amazon Web Services (AWS). Instances are available to offer custom accelerations with FPGA for the developers and are easy to use. The user is connected in just a few clicks or commands to the remote FPGA infrastructure. He just requires internet access to create AWS account to connect to FPGA instance.

Today, AWS has deployed FPGA instances in four regions of the world: Ireland (Western Europe), Oregon (USA west coast), Virginia (USA east coast) and Sydney, Australia (Asia-Pacific-southeast). According to the region and the user's location, network efficiency may vary and price as well.

Web interface is available to configure interactively the instance. But the user can create and configure the instance only by using command line interface and scripts.

Architecture

AWS FPGA instance provides access to Xilinx Virtex UltraScale+ FPGAs. The FPGA is controlled by a bus interface (PCI express). Each FPGA contains approximately 2.5 million logic elements. Billing is based solely on time of use.

The instance may contain one, two or eight slots. One slot contains one FPGA. Since resources are limited, it may happen that no FPGA is available at the creation request, but this this is relatively rare.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

For multi-core simulation, each core is mapped on its own FPGA: we need in this case a minimum of 2 slots.

Instance F1 name	2x	4x	16x
Number of slots	1	2	8
Price per hour ($)	1.65$	3.3$	13.2$

Table 1: FPGA board configuration sets

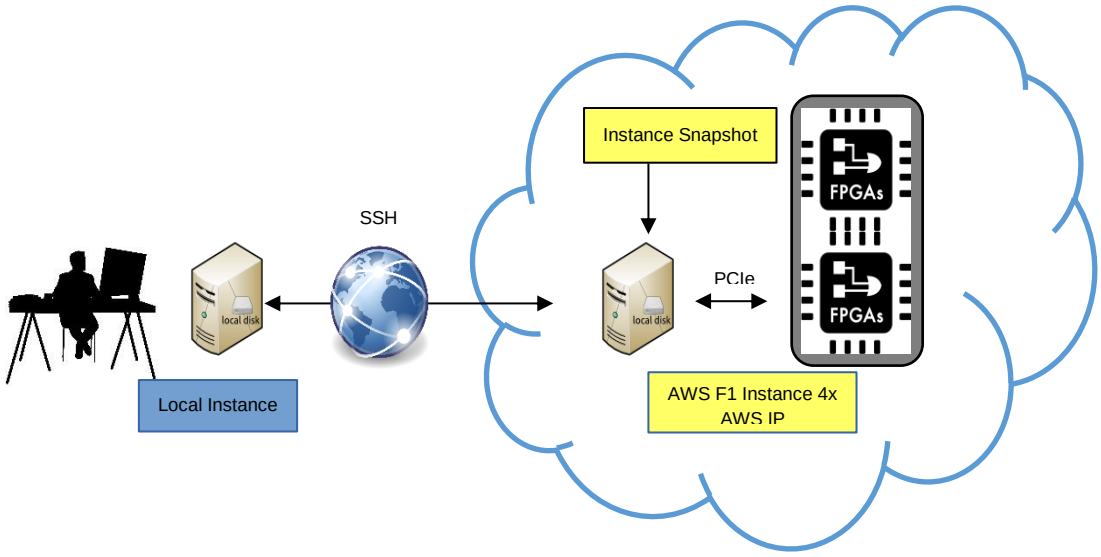


Figure 14 - Remote FPGA infrastructure

Network, security, how it works

First, the user runs command to create FPGA instance with his AWS account identity. This command returns AWS instance parameters which are used to connect to the instance with the Secure Shell protocol (SSH).

The instance is loaded with a snapshot image stored remotely. It contains VSORA libraries and tools to compile and run applications on native, high level, low level and FPGA platforms.

Some TCP or UDP ports specified by the user may be opened to exchange data safely between applications running only on his local instance and the FPGA instance.

Then the user can send source code to the instance and compile his own application for FPGA platform.

AWS Global FPGA Image (AGFI)

When the user is connected to the FPGA instance, he has to load VSORA's logic into the FPGA. VSORA has synthesized several FPGA images (AGFI) with various characteristics. Each AGFI is specified by a number of Base Units, by the ALU type and by the quantization.

Before running any application on the FPGA platform, the user must load a FPGA image to a specific slot number with a simple command from his own instance. Then the user can launch the application compiled on the FPGA instance. This application will send messages to the FPGA through the PCI express bus (this corresponds to the

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

messages send from the mailbox on the host side to the mailbox on the core side. See Figure 9 - VSORA's DSP architecture). A simulation executed on the FPGA is up to 100 times faster than on high level platform.

3.6. Experimental results

At this stage of the project, we do not have any results to present. For the time being, we put in place the framework to enable the use of FPGA on the cloud. The framework has been validated on simple use case, not on the DSP application envisaged in this project. In the same time, we started to work with Task 2.2 towards the implementation of some innovative ideas developed in this task. It is planned to demonstrate on VSORA DSP framework one of these ideas.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

4. ETHERNET TSN TESTBED FOR TIME SYNCHRONIZATION WITH SDN-BASED MANAGEMENT

4.1. Ethernet TSN Testbed for Time Synchronization with SDN-based Management

IEEE Ethernet Time Sensitive Networking (TSN) is a set of standards that extend Ethernet to support real-time communications. These standards are expected to have important roles in the future industrial systems such as Cyber-Physical System (CPS). As a key standard in TSN, IEEE 802.1AS [AS] (hereafter referred to as simply AS) specifies the clock synchronization/time synchronization (TS) for Ethernet network, targeting the precision levels that are required by future communication systems such as smart factory networks or CPS.

In the IEEE 802.1AS standard, the synchronization is performed among the clocks of network devices, which are connected via Ethernet links, switches, and/or bridges. The device with the clock selected as the source of time is called Grand Master (GM). This standard selects the GM using the Best Master Clock algorithm (BMCA). Then the GM's clock provides the time reference to other devices across the network.

4.2. Ethernet TSN Testbed for Time Synchronization

General Architecture

In IEEE 802.1AS, some important parameters are the priorities of devices (priority1, priority2), the role of Ethernet port (portRole), i.e., master or slave. The priority parameters allow the algorithm BMCA to select GM. Through setting the priorities and portRole, our platform allows the algorithm BMCA inside each device to recognize quickly the selected GM. This recognition is especially useful in a large-scale industrial network.

Configuring automatically the AS parameters is a challenging work. First, besides above important parameters, this task requires to take into account various other parameters and network information (e.g., topology, traffic pattern). Second, the future industrial systems require high levels of flexibility and re-configurability, which have some important effects such as low time-to-integrate and on-the-fly configuration. As a result, we utilize Software-Defined Networking (SDN) to propose a solution that manages and configures TSN, focusing on IEEE 802.1AS. SDN is a dynamic network architecture that can support to build a flexible and self-configurable solution. The proposed solution is implemented into our SDN-enabled testbed, then verified and analysed by several performance evaluations. More information about this testbed can be found in [CEA], which is conducted in the context of our research on future CPS.

We implement the proposed solution into our SDN-enabled platform, named NEON [Decremps14], [Labraoui17]. Three important components of NEON are southbound API, controller, and services (Figure 15). NEON southbound API is an SDN southbound, which allows to manage network devices such as access points, switches/bridges, and gateways. NEON controller, similar to other SDN controller, is a logically centralized entity that maintains the information about the network devices and allows to control the whole network. NEON services are the software running on top of the controller, providing network functionalities, such as monitoring and configuring.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

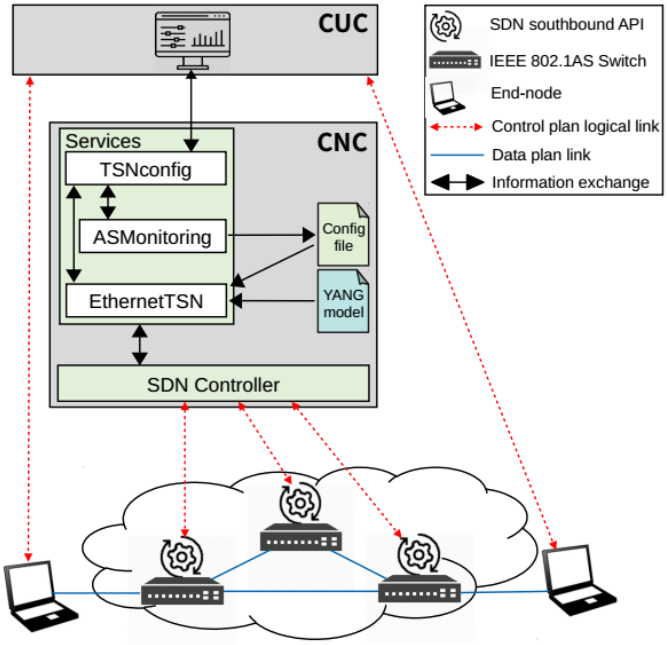


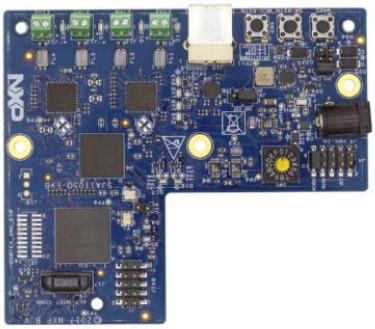
Figure 15 - General architecture of the testbed

The two notable services in this testbed are ASMonitoring and EthernetTSN. The ASMonitoring service, via controller, will monitor the network devices, compute proper values for AS parameters, and generate AS configuration file for each device. The EthernetTSN service prepares and sends TSN configuration parameters to the network devices. This service gets the TSN configuration files and validates them to the TSN YANG models (e.g., IEEE 802.1AS YANG model). The objective is to verify the presence of mandatory parameters and to check whether the proposed values are inside the predefined ranges. Centralized Network Controller (CNC) and Centralized User Configuration (CUC) are main components in the TSN standard IEEE 802.1Qcc, which allow to configure the network devices based on user requirements.

Testbed setup

We implement the standard IEEE 802.1AS into the NXP© SJA1105 Ethernet switches, which are connected to Intel© I210-equipped endnodes (Figure 16). There is also available implementation from NXP; however, our implementation allows us to have full control on the switch. We setup a testbed with two end-nodes and two switches; they connect together in a linear topology (Figure 17). The time synchronization program on the end-nodes is implemented based on linuxptp [linuxptp]. Since linuxptp is an open software, it can be managed by one of our plugins inside NEON southbound API. As currently CUC entity is not implemented in the testbed, we propose that the CNC (i.e., SDN controller and services) takes charge of the endnodes' configuration directly without passing through CUC. After the CUC entity is implemented, it will be responsible for getting user requirements, end-nodes' capabilities, and pushing end-nodes' configurations.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276



Switch: NXP® SJA1105Q

End-node: Intel® Ethernet Controller I210

Figure 16 - Switch and endnodes' network card

The default AS parameters of the testbed are showed in Table I. In IEEE 802.1AS, sync message and follow-up message are the messages that the nodes exchange in order to compute and synchronize their clocks. Path delay is the propagation delay between two nodes. Note that Table 2 shows the values of the interval parameters in binary logarithm, i.e., if the value is -3 then the interval is 0:125 second. In the evaluation, we set the default run time to 5 minutes, and the default GM to endnode 1.

Device	Parameter	Description	Value
Switch	sync interval	interval between two *sync messages*	-3
	path delay interval	interval between two messages that used for measuring *path delay*	-1
	sync receipt timeout	number of *missed sync/follow up message* that determines a missed transmission	1
End-node	sync interval	interval between two *sync messages*	-3
	path delay interval	interval between two messages that used for measuring *path delay*	-1
	sync receipt timeout	number of *missed sync/follow up message* that determines a missed transmission	10

Table 2 : AS parameters of the nodes

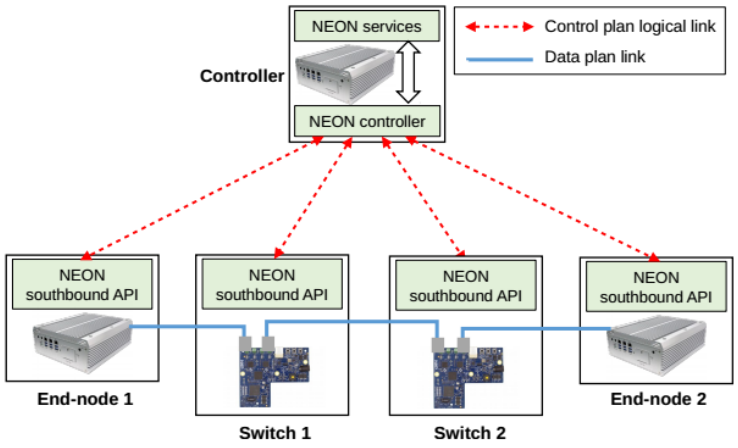


Figure 17 - Testbed setup

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Experiment results

The evaluation results show that the solution is able to provides basic/essential functionalities, e.g., managing devices/topology and configuring AS. The TS process, which is configured by the solution, provides highly precise synchronization, on the scale of hundred nanoseconds.

We first analyse the scenario in which a new device is connected while the network is running TS. Specifically, the network initially includes two switches and end-node 1; we run TS for 1 minute then we connect the end-node 2. After the end-node 2 is connected, the ASMonitoring service detects the new device and generate an AS configuration file. The EthernetTSN service sends the AS configuration to this device, allowing it to join the network and start synchronizing its clock to the GM's clock.

In Figure 18, we depict the offset between the GM and other devices. Note that the offset of end-node 1 is not available because end-node 1 is the GM. The figure shows that the new device (i.e., end-node 2) is synchronized to the GM without impacting other nodes. In the proposed solution, a new device is recognized automatically when it connects to the network; and this device is synchronized immediately if it is AS-capable. In other words, the current solution can provide an initial result of flexibility, i.e., the ability to automatically admit new device into TS process. On the other hand, when a device is synchronized, the offset between it and the GM is on the scale of hundred nanoseconds. This is a tight accuracy for local area network, compared to other works in the literature, which have the offset "on the microsecond scale", as stated in [Volgyesi17].

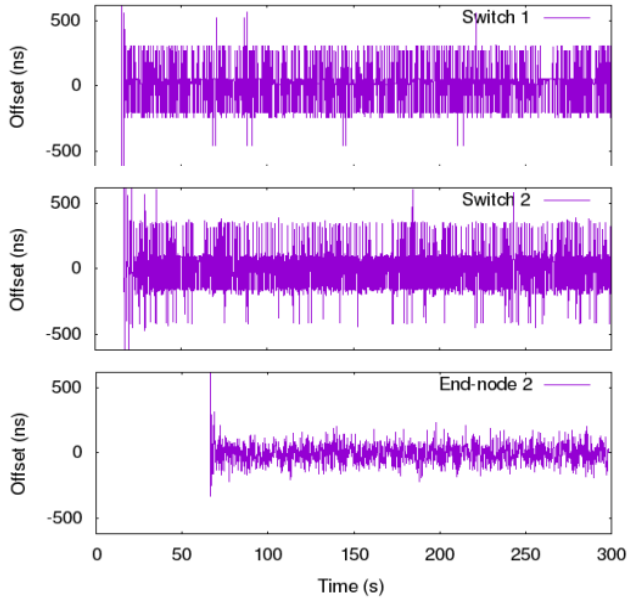


Figure 18 - Time offset between GM's clock and other clocks in scenario of connecting new device

We evaluate another scenario in which the GM role is changed from end-node 1 to end-node 2 (Figure 19). Before changing GM, switch 1 and switch 2 synchronize their clocks to the clock of end-node 1. After changing, the two switches and also the end-node 1 automatically start synchronizing to the clock of end-node 2. The reason of the change can be that, as an example, the clock accuracy of the newly connected end- node 2 is higher than end-node 1, so the ASMonitoring service decides to change then generates corresponding configuration. The EthernetTSN service sends the newly generated AS configuration to the network devices. Figure 19 shows that after changing the GM, the TS process is disrupted for around more than 10 seconds. We define that the TS process is disrupted when the offset is outside the range [−1000ns; 1000ns]. Within the disrupted period in Figure 19, the offset values are in the order of million nanosecond. After changing GM, the two switches obtain again the performance as before the changing. In other words, the current solution can provide an initial reconfigurability, i.e., the ability to reconfigure for new GM.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

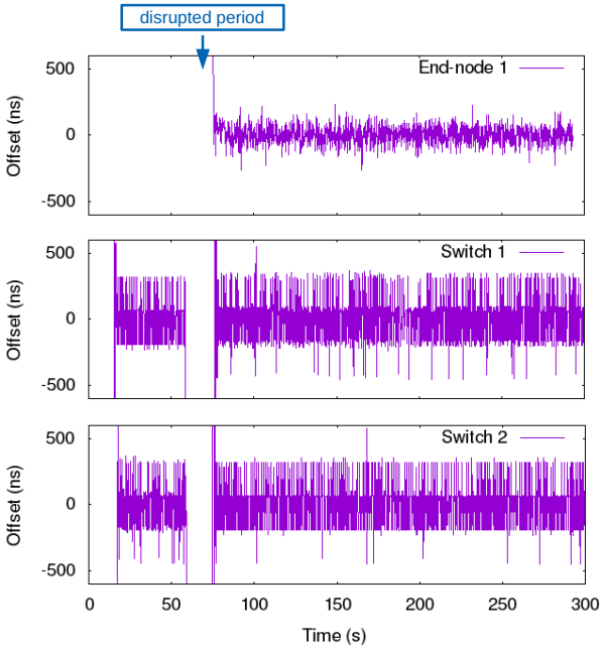


Figure 19 - Time offset in the scenario of changing GM

Based on the current results, we plan to develop further the testbed; one of important development is to integrate wireline TSN into wireless systems such as 5G radio. The integration between TSN and 5G is a demanding requirement in future communication systems, especially in the context of Industry 4.0 and Cyber-Physical System (CPS). An example of the need for that integration is the connection between wireline floor network and the remote manufacturing execution system that can only be reached via wireless links.

4.3. References

[AS] "IEEE draft standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications," IEEE P802.1ASRev/D6.0 December 2017, pp. 1–496, Jan 2018.

[CEA] M. -T. Thi, S. Ben Hadj Said and M. Boc, "SDN-Based Management Solution for Time Synchronization in TSN Networks," 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 361-368.

[Decremps14] S. Decremps, S. Imadali, and M. Boc, "Fast deployment of services in sdn-based networks: The case of proxy mobile ipv6," Procedia Computer Science, vol. 40, pp. 100–107, 2014.

[Labraoui17] M. Labraoui, M. Boc, and A. Fladenmuller, "Self-configuration mechanisms for sdn deployment in wireless mesh networks," in 2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM). IEEE, 2017, pp. 1–4.

[linuxptp] R. Cochran et al., "The linux ptp project."

[Volgyesi17] P. Volgyesi, A. Dubey, T. Krentz, I. Madari, M. Metelko, and G. Karsai, "Time synchronization services for low-cost fog computing applications," in 2017 International Symposium on Rapid System Prototyping (RSP). IEEE, 2017, pp. 57–63.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

5. CONCLUSION

In the document D2.5, we presented the achievements of the Work Package 2, task 2.3 dealing with the prototyping activity of current and future communication systems for Cyber Physical System.

The integration of existing technologies led to the design of the PIARCH board. This is the result of a tight collaboration with teams working in Work Package 6: we received from this group vertical specifications and the resulting board will be used by a secure gateway to enable remote monitoring and maintenance of a connected and possibly autonomous vehicle.

The deliverable D2.3, dealing with novel enablers for future communication systems going beyond 4G could offer, presented in particular a MIMO system based on algorithms using trained neural network (training is done offline). We propose to implement this algorithm on a real target. This allows to face issues not seen during the mathematical study: determine the computation requirements to run the algorithm under real time constraint, evaluate the impact of the quantization and focus more precisely on memory management.

To achieve this, we presented a new way of development: whereas usually the implementation of the algorithm involves several teams with their own expertise, by using this process, DSP engineers can manage at a very early stage of development the various implementation issues (processing capacity, memory, quantization) and integrate / modify the algorithm to meet the requirements in terms of performances and silicon as well. The time to market, which is also a key point in a commercial project, should dramatically decrease.

This new process is articulated around simulation platforms: the developer keeps the same code at all stages of development and with different compilation options can refine the simulation until a simulation on a FPGA platform. By using commercial offers of remote FPGA in the cloud, we can run long simulations on realistic target with limited costs.

Regarding the networking constraint, which is also a key point in the design of a low latency communication system, we presented first results of the implementation of the IEEE 802.1AS norm which deals with time synchronization network issue. Preliminary results show that the synchronization offset between the grand master and a device is on the scale of hundred nanoseconds, which fulfil the requirements and is promising for further simulations.

This document is a first release describing the process of the validation. Results of implementation will be completed and presented in the final version at the end of the project.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276