Project number: 826276

CPS4EU

Cyber Physical Systems for Europe

D2.6 Simulation tools and experimental platforms v2

Reviewer (name – company): P. Gougeon (Valeo), E. Hamelin (CEA)

Dissemination level: Public

Version	Date	Author (name – company)	Comments
V1.0	2022/08/12	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)	
		Eric Dujardin (Thales TRT)	
		David Choukroun (Sequans)	

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

EXECUTIVE SUMMARY

Work Package 2 (WP2) of the project CPS4EU addresses the communication aspects of Cyber Physical Systems (CPS). We study how to guarantee high-performance and secured communications for CPS, based on the technologies such as 4G, 5G, MIMO, IEEE 802.1 Time-Sensitive Networking.

This deliverable is the second report of Task 2.3 in WP2. The solutions and results presented in D2.5 are further discussed in this D2.6.

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. *Objective*

The objective of this document is to present the validation and evaluation results of WP2. This WP focuses on the communication modules for next generations of Cyber Physical Systems (CPS). The solutions that we identified in WP2 are based on well-known technologies: Wi-Fi, Ethernet, high-performance LTE (cat 4), low-power LTE (Cat M1, Cat NB1), 5G, MIMO, IEEE 802.1 Time-Sensitive Networking (TSN). The work presented in D2.1 have highlighted the need of a pre-integrated platform. We provide the validation of the PiArch platform with the board and the preliminary test and validation.

A 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 technologies in 5G, especially in the aspects of high bandwidth and low latency. Therefore, this document will also present evaluation results of MIMI encoder and TSN. These technologies are important in 5G to guarantee high bandwidth and low latency, which are required by future CPS.

On 5G MIMO, this document focuses on the simulation process of some selected innovative solutions on a real DSP. This simulation is done on different platforms and finally we will run the simulation using a DSP mapped on remote FPGAs. On TSN, we describe the wired and wireless TSN testbeds and their evaluation results of IEEE 802.1AS Time Synchronization, IEEE 802.1Qbv Scheduling, and IEEE 802.1Qav Queuing Enhancement.

1.2. *Definition, acronyms, and abbreviations*

Acronym / abbreviation	Description
AI	Aritificial Intelligence
AGU	Address Generator Unit
ALU	Arithmetic and Logic Unit
API	Application Programming Interface
AWS	Amazon Web Service
CPS	Cyber-Physical System
CPU	Central Processor Unit
DMA	Direct Memory Access
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GCC	GNU Compiler Colection
GM	Grand Master
LTE	Long Term Evolution
LTE – M	Long Term Evolution – Machine Type Communication
MIMO	Multiple Inputs Multiples Outputs
PReLU	Parametric Rectified Linear Unit
PCI	Peripheral Component Interconnect
PCIe	PCI express
PiArch	Pre-Integrated Architectures
ReLU	Rectified Linear Unit
RTL	Register Transfer Level
SIMD	Single Instruction Multiple Data
SSH	Secure Shell
TCM	Tightly Coupled Memory
TLM	Transfer Level Model
TSN	Time Sensitive Networking
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. SIMULATION OF A MIMO DECODER ON A DSP MODEL

2.1. *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

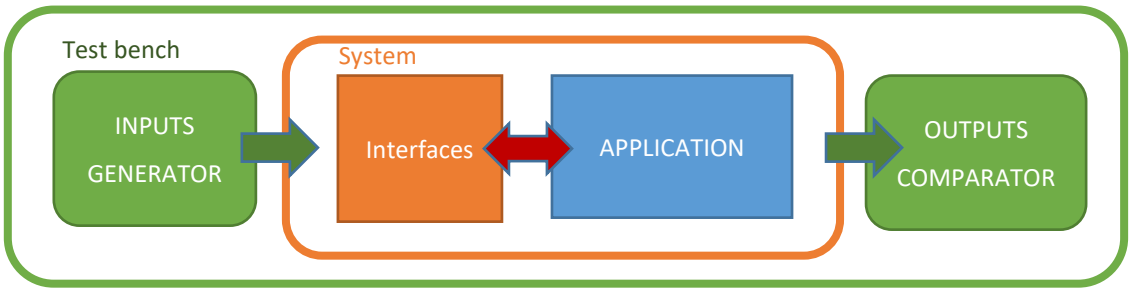


Figure 1: Whole neural network system simulation

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 implementations: 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 use is limited and cannot manage complex scenarios; it can be replaced then by a true algorithm generating data on the fly.

2.2. *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 2. 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

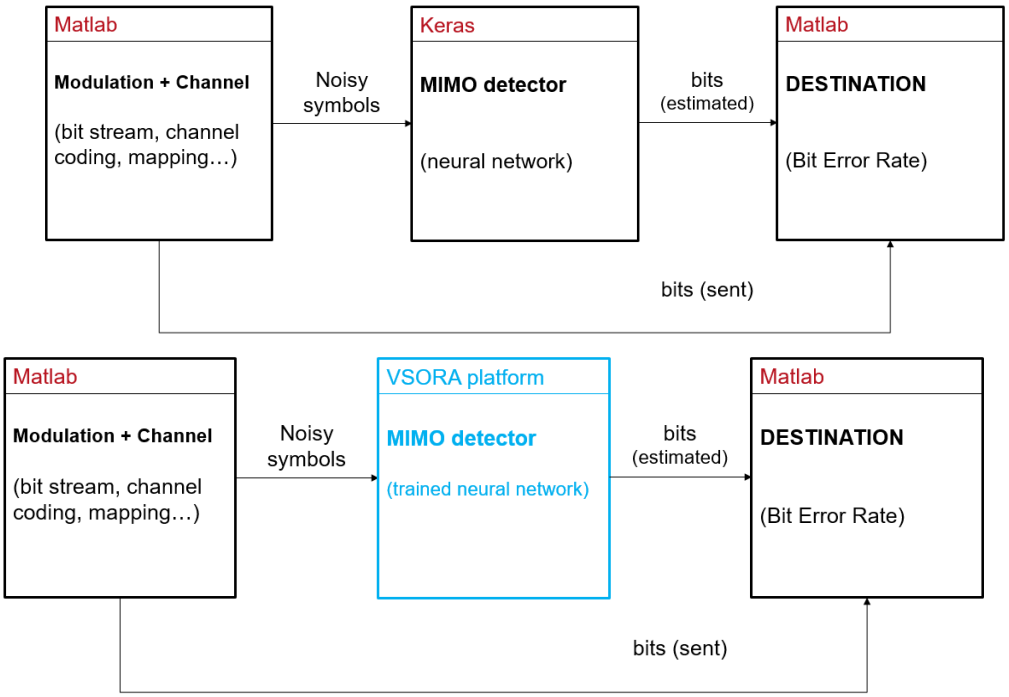


Figure 2: Illustration of the test bench

2.3. *VSORA library (Add-on VS D2.5)*

The project CPS4EU allowed VSORA to continuously improve the DSP library (VSLIB) and to enrich it. We can mention especially the following main achievements:

- Adding the new container “vector”: this container allows to handle single dimension matrices (n x 1) with n is in range 1 to 2^32. This container completes the matrix (2d container (n x m) with n and m in range 1 to 2^16) and matrix3d (3d container (n x m x d) with n and m in range 1 to 2^10 and d in range 1 to 2^12).
- Sort function: this function has been added to the instruction set with a close interaction with hardware acceleration. Pure software sort function is very time consuming. With this implementation combining hard and soft and based on the radix method, we gain efficiency and we are close to the theory regarding the number of cycles consumed.
- Dedicated functions for AI processing: activation functions have been added as unary functions (first stage in the processing pipeline) (ReLU, ReLU6, PReLU, sigmoid, tanh …). It is also possible to load in a RAM the values of a user-defined function for specific applications.

The library is released with the simulation platforms. An exhaustiv documentation (finalized during the project CPS4EU) is attached to each release. This documentation is created using the open source project Sphinx (Sphinx is a documentation generation tool: originally created for Python documentation, Sphinx is now used widely among the open source world).

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

Figure 3: Main page of VSLIB's documentation

This documentation is html pages to facilitate navigation (a significant effort has been made to create hyperlinks between pages).

All functions supported by the DSP are listed. Each one has a description in a dedicated page containing:

- A clear definition with parameters explanation.
- A list of the constraints of use with the related unitary tests to validate these constraints.

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

- An example and the corresponding outputs. This code is included in the regression test and regularly checked.

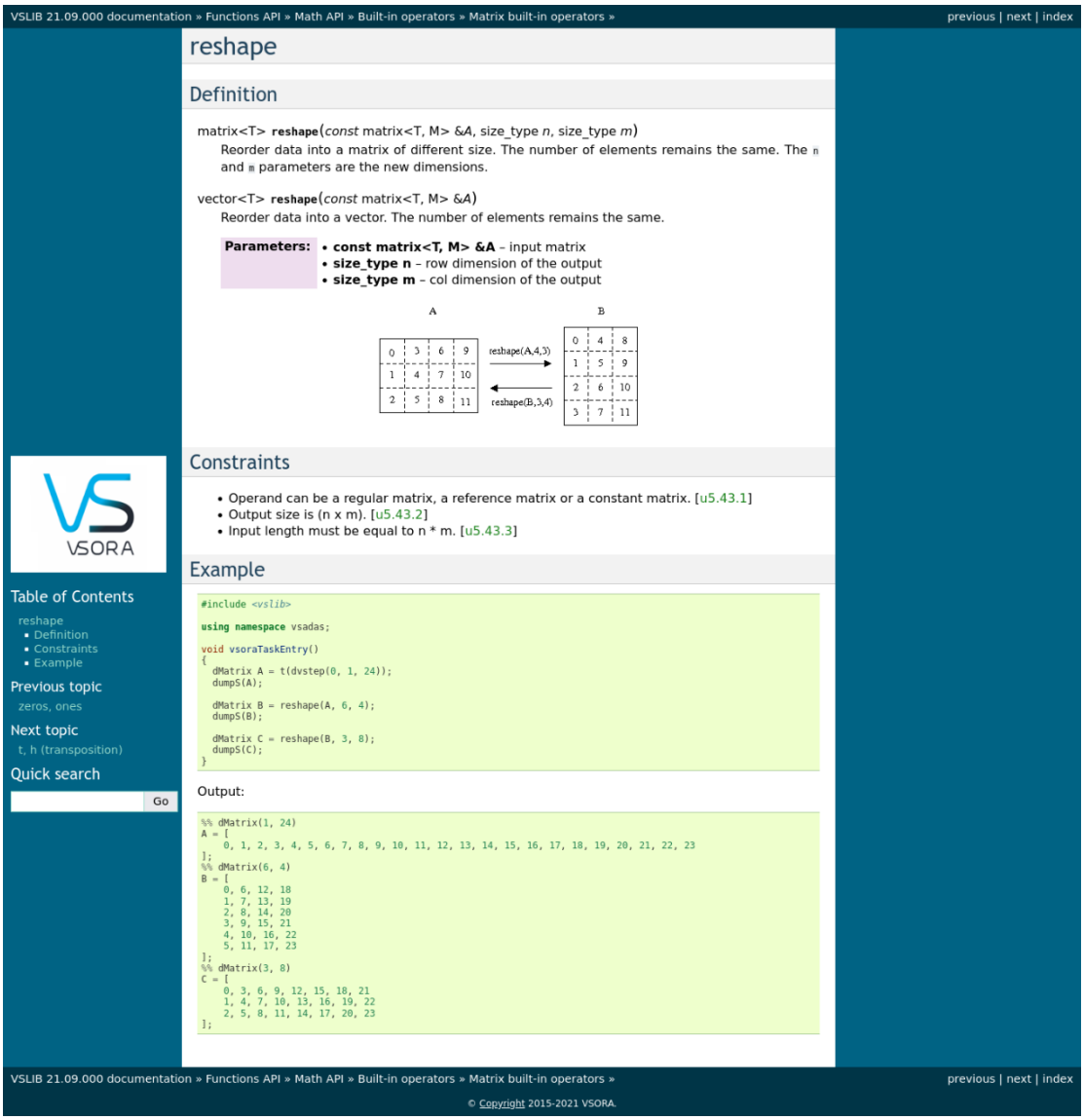


Figure 4: Example of function reshape documentation

A porting guide is also included to help the end user to run vslib in a custom embedded system: the library has been already ported by VSORA to an ARM host processor and we expose in this section the main steps to use another host processor.

2.4. *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.

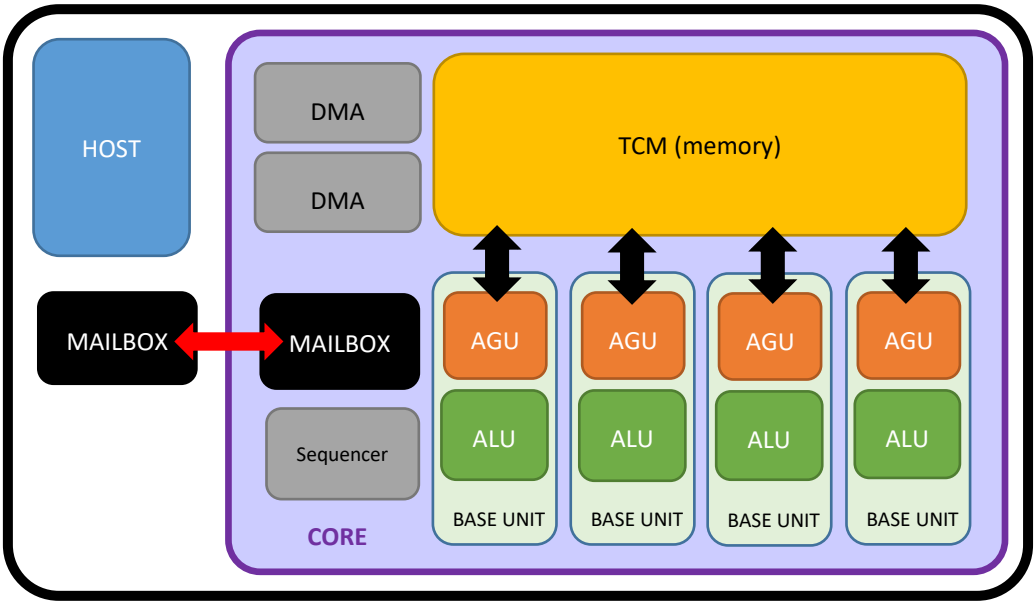


Figure 5: VSORA's DSP architecture

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 2.5) 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

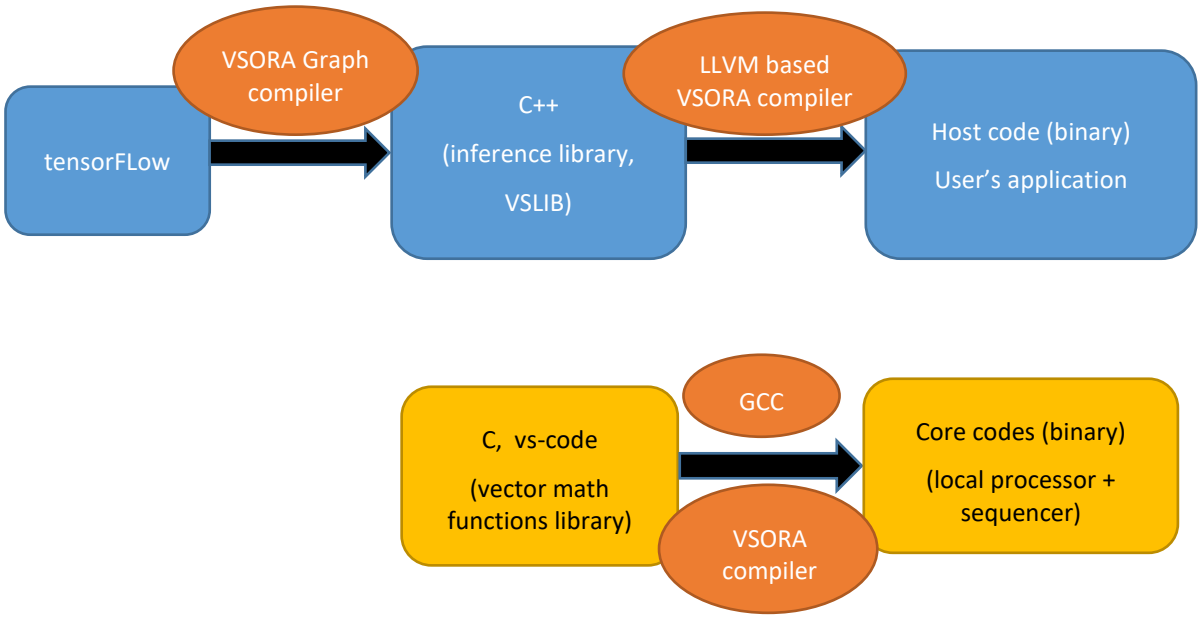


Figure 6: VSORA's DSP Compilation flow

2.5. *Simulation platforms*

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

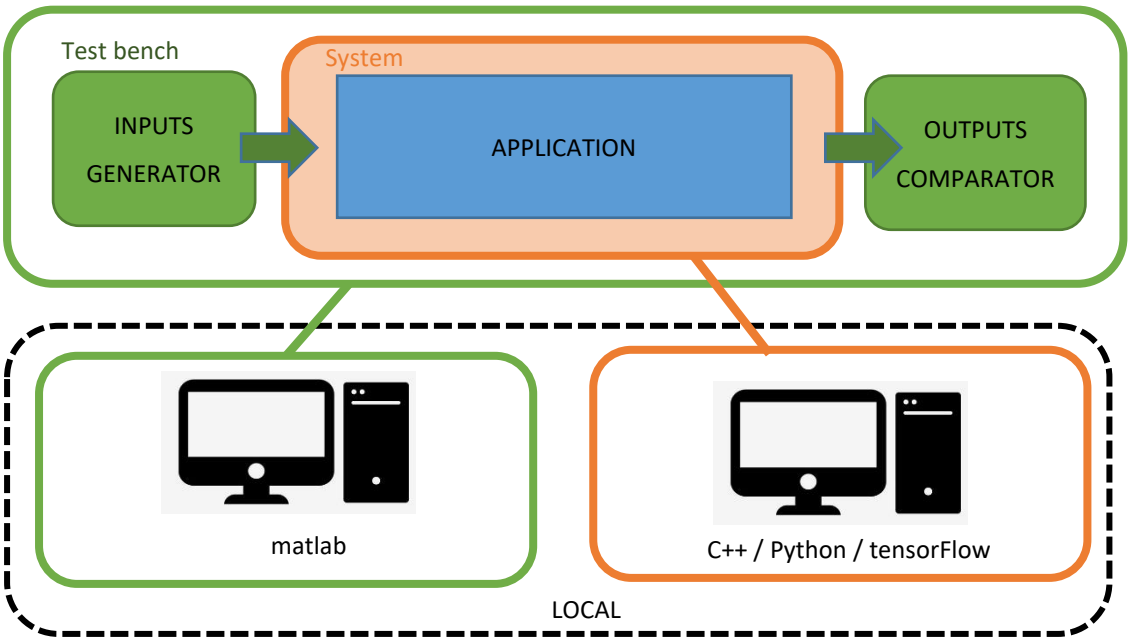


Figure 7: Native simulation

- **Native platform**

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.

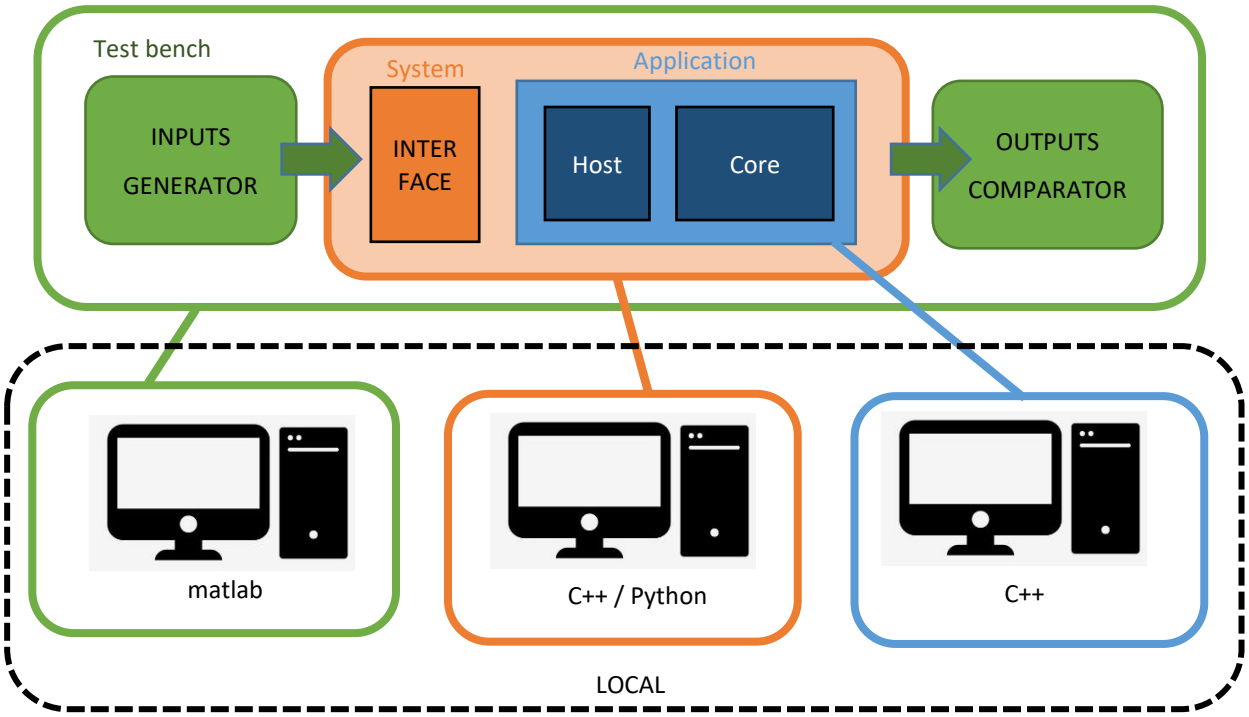


Figure 8: High level simulation

- **High Level Platform**

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**

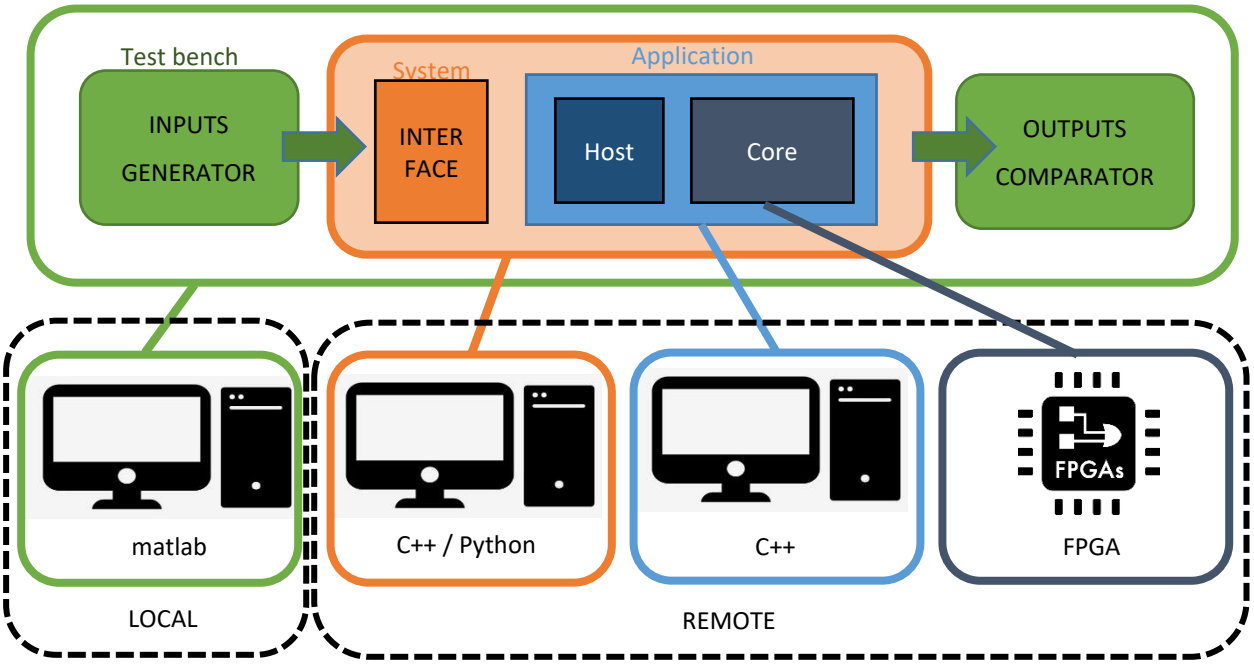


Figure 9: FPGA simulation

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).

2.6. *Remote FPGA infrastructure*

Description

VSORA has chosen to use the cloud computing platform from Amazon, called Amazon Web Services (AWS) F1. 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.

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

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

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

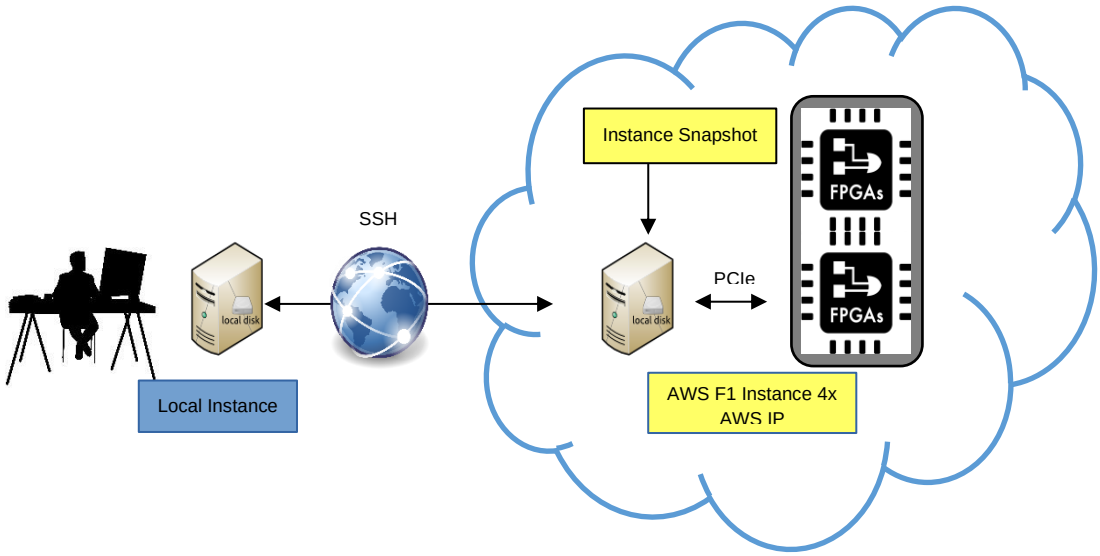


Figure 10: Remote FPGA infrastructure

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 5: VSORA's DSP architecture). A simulation executed on the FPGA is up to 100 times faster than on high level platform.

2.7. *Experimental results (Add on VS D2.5)*

Implementation of a Neural Network on VSORA DSP

Let's present the different parameters that must be considered when implementing a neural network on a DSP (especially VSORA DSP).

- DSP: it executes the algorithms of the neural network. It is characterized by **internal resources**: mainly the number of arithmetic units (especially number of MACs) and the size of the internal Memory (TCM: Tightly Coupled Memory). These parameters have an impact on the silicon size (and thus on the solution price) as well as on the power consumption.
- Inputs: data to be handled depends on the application: it can be images, text, sound, or data stream in the case of the MIMO decoder. We define the notion of batch: a **batch** represents the number of inputs that the DSP is processing in parallel. batch1: a single input is handled by the DSP; batch N: N inputs are handled in parallel by the processor. We define also the notion of **slicing**: in some cases, the input cannot be processed as a single entity. For example a High Definition image could require more memory than the TCM can support. In this case we cut (slice) the input on smaller parts which can be processed. Slicing can also be applied on Weights: this has an impact on the external stream parameter. Note that batch size and slicing are not exclusive: we can slice an input in M parts and group them in a batch of N elements.
- Outputs: this represents the data processed by the DSP. Given a frequency for the DSP, we can define the **throughput** which is the number of data handled per time unit. Depending on the type of input data, we speak about the number of images per second, number of bits per second, etc. We define also the **latency** which is another parameter characterizing the outputs and different from the throughput. It represents the delay needed by the DSP to process an input.
- External Memory: in most of the cases, the internal memory cannot store all the data needed for the processing. This data is typically the weights of the network. The amount of data to be stored externally is generally not an issue since DDR is a massive storage with a large capacity, but the **external stream**

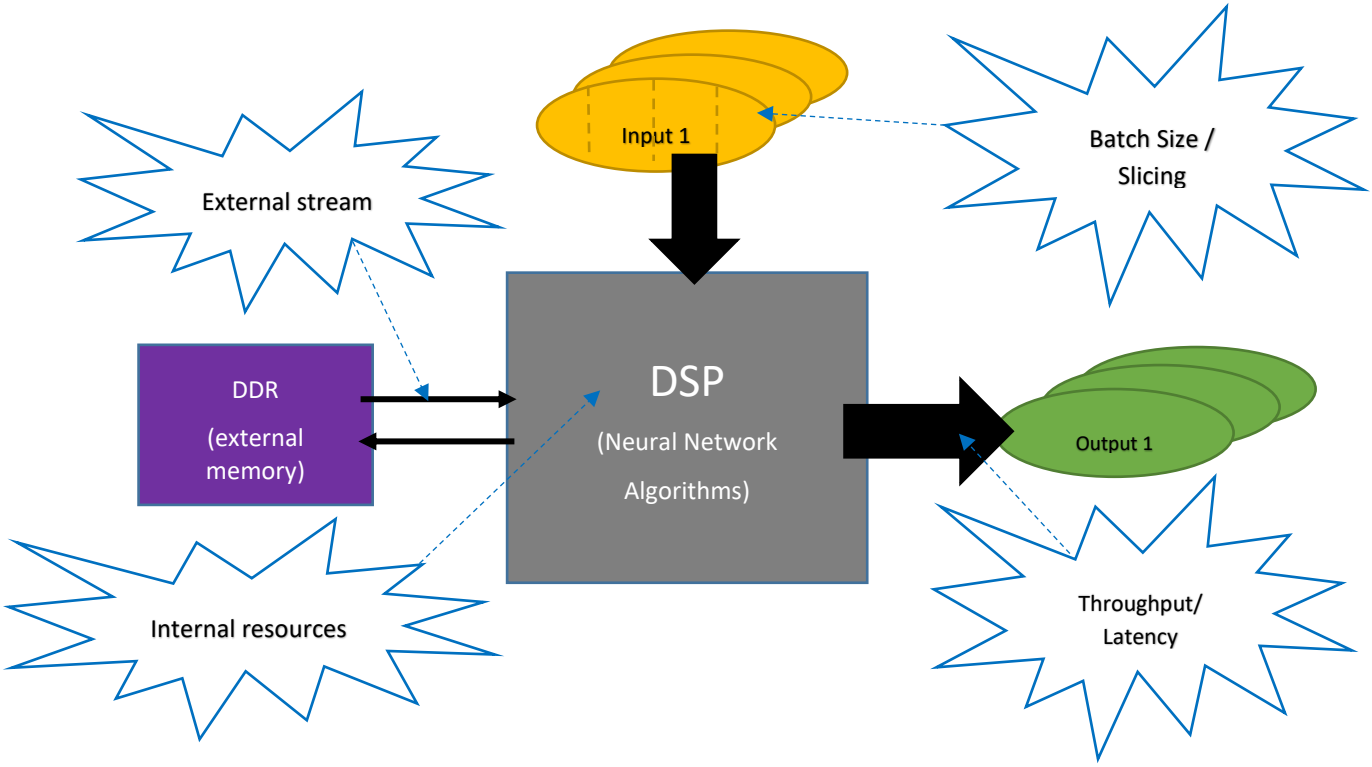


Figure 11: Parameters of implementation of Neural Network on DSP

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

of this data between the DDR and the DSP is limited by the system. This parameter has an impact on the power consumption: data transfer between an external RAM and the DSP requires more energy than keeping them internally (the consequence on the solution's price is marginal).

- **Throughput vs latency vs batch size**

Now, let's study the relationship between throughput, latency and batch size. Consider first the time necessary for processing of elements. For a given system (number of MACs, frequency), we can determine the minimum time T_{min} required for the processing of one element. Since all resources are not used 100% of the time, an efficiency factor α (α between 0 and 1) should be applied (α is also the rate of use of internal resources). Thus, the effective time necessary for processing (N is the batch size) is:

$$T_N = \frac{N}{\alpha_N}.\ T_{min}, \qquad 0 < \alpha_N \leq 1$$

This value is exactly the definition of the latency of the system.

We observe with simulation that:

$$\alpha_N \geq \alpha_1.$$

Indeed, it is always more difficult to allocate all resources to process a single input than to process N inputs. The DSP handles blocks of data on a regular basis. When the number of processing units increases or the size of data get smaller, it is more common to have idle units in the system and then to have a lower efficiency.

Let's define (T_1, α_1) and (T_N, α_N) the parameters time of processing and efficiency factor for a system with batch 1 and batch N respectively.

The throughput TP is defined by

$$TP_N = \frac{N}{T_N} = \alpha_N.T_{min}$$

We conclude that:

$$TP_N > TP_1$$

By increasing the batch size, we improve generally the throughput of the system. The gain is the factor $\frac{\alpha_N}{\alpha_1}$. For small network (few processing) or small vectors input data, and with a lot of processing units, α varies greatly. However, in most common neural networks, we observe that the DSP developed by VSORA has a high level of efficiency, even for a batch size equal to 1 (ie $\alpha_N\ close\ to\ \alpha_1$): increasing the batch size does not improve the throughput in a significant way.

The drawback of increasing the batch size is that the latency is increasing as well. The increase of the latency between a batch size 1 system and a batch size N system is $N.\frac{\alpha_1}{\alpha_N}$.

Batch size increase is reserved for very small networks or when the latency is either not a key point or remains within an acceptable range.

- **Internal memory vs external memory**

The external memory stores generally the weights of the network. The developer has 2 choices:

- Load weights at the start-up of the processing and keep them in the TCM.
- Load weights on the fly.

The first solution requires to have enough space in the internal memory. In this case, the overhead due to the load at start-up is negligible and we gain on the power consumption (DDR interface is not / little used)

The second solution allows to have a smaller TCM, and then a smaller silicon footprint which is directly linked to the price of the solution. However, the software architecture to support weights load / unload is more complex and we risk of having idle processing units if the load process is too slow.

- **Internal resources vs latency**

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

A solution to reduce the latency is to increase the number of processing units. This solution increases also the silicon area of the DSP (linked to the price) and is not linear. Depending on the dimensions of the processed matrices, the probability to get idle computing units is higher and thus the efficiency of the processing is lower. In this case, the solution is to increase the batch size with the drawback explained above.

We see that the mapping of a neural network on the DSP is a multi-dimensional problem. Many factors have to be taken into account (throughput, latency, batch size, data slicing, silicone area, power consumption). There is not a unique solution, and the choice has to be made by the developer. The hardware configuration can vary only in the case of an IP definition, otherwise the resources (computing unit, external memory bandwidth, internal memory size) are fixed. The context of use also imposes its constraints: for example, the latency is a key point in real time applications.

The choice is often not easy: the development flow proposed by VSORA and especially the high level platform allows to evaluate different solutions and helps the customer to make the best decision.

Validation of the library and HW on various neural networks:

VSORA implemented various public neural networks to validate the library and the hardware. The algorithms cover multiple types of neural network: FC (Fully Connected), CNN (Convolutional Neural Network, RNN (Recurrent Neural Network), Transformer Model, Mixed Mode (DSP & AI)

AlexNet	CafeNet	DeepSpeech
GoogleNet v3	SqueezeNet	Bidirectionnal LSTM
Inception v3	Vgg16	DLRM
MobileNet v1	Vgg19	SyntaxNet
MobileNet v2	Yolo v1	KITTI PointPillars
ResNet50 v1	Yolo v3	SRResNet (CNN + others)
ResNet50 v1.5	Yolo v4	SRGan (CNN + others)
ResNet50 v2		

We can also mention the neural network proposed by CEA for the MIMO algorithm which is of type “Fully Connected”.

High level simulation results

We ran this neural networks on various DSP configuration (different number of MACs) with the high level model and we measured the performances obtained. We are generally satisfied with the results and we give as an example results for the ResNet50 v1.5.

- Frequency of DSP: 2Ghz
- Batch Size : 1

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

- Algorithm complexity: 4.09 GMACs
- Image size : 224 x 224 (RGB)

Core Size (MACs)	Nb of cycles	Nb of images	Latency (ms)	Efficiency
256	1.91e+07	105	9.56	84%

We do not disclose results for other configurations or networks since this information is confidential, for more details please contact VSORA directly.

FPGA simulation results

The place and root of the DSP database is done with the tool “vivado”. The tool generates reports of the FPGA synthesis. A synthesis lasts approximatively 4 to 6 hours. As the IP evolves regularly, some issues arise that require reworking the RTL model in order to achieve good performances.

The constraints of the FPGA (number of LUTs) allow to map a DSP IP with a maximum of 2 base units (16 ALUs). The TCM size is also limited (7.4 MWords for a quantization of 10 bits).

Once the FPGA is synthetized, we load the binary file on Amazon’s datacenter and we get an AGFI (Amazon Global FPGA Image): this is an identifier which will reference it.

In the same time, we create an AMI on amazon’s server. An AMI (Amazon Machine Image) is a supported and managed image in the AWS ecosystem (amazon’s container format equivalent to kubernetes or docker with small differences since it is linked to a specific hardware), that provides the information required to launch an instance. VSORA default system is Ubuntu (18.04 and 20.04) which is in the list of supported machines / OS by Amazon.

3. CLB Logic

Site Type	Parent	Child	Non-Assigned	Used	Fixed	Available	Util%
CLB LUTs	358713	0	0	358713	0	895200	40.07
LUT as Logic	356293	0	0	356293	0	895200	39.80
LUT as Memory	2420	0	0	2420	0	450720	0.54
LUT as Distributed RAM	2408	0	0	2408	0		
LUT as Shift Register	12	0	0	12	0		
CLB Registers	182726	0	0	182726	0	1790400	10.21
Register as Flip Flop	182674	0	0	182674	0	1790400	10.20
Register as Latch	52	0	0	52	0	1790400	<0.01
CARRY8	8218	0	0	8218	0	111900	7.34
F7 Muxes	17403	0	0	17403	0	447600	3.89
F8 Muxes	3934	0	0	3934	0	223800	1.76
F9 Muxes	0	0	0	0	0	111900	0.00

Figure 12: Screen shot of the place and root report

We launch the FPGA simulation on the neural network ResNet50 v1.5. Here are the different stages of the simulation:

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

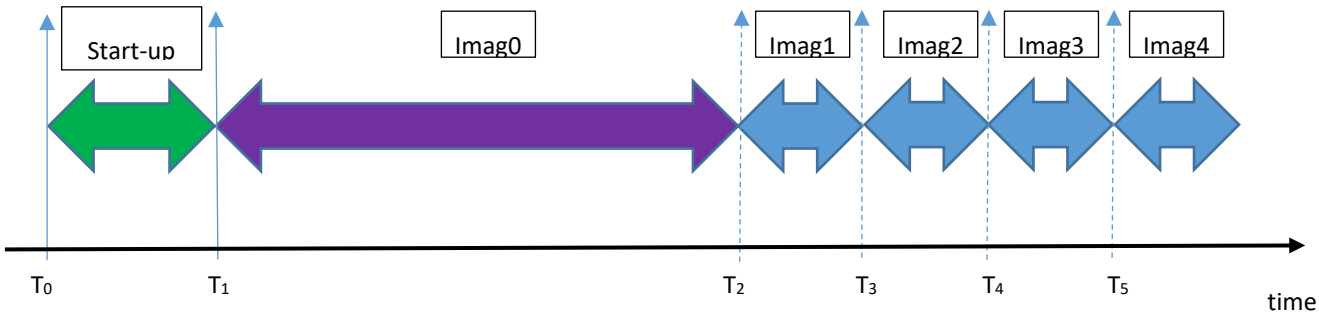


Figure 13: Scheduling of AI processing on FPGA

TIME STAMP	DESCRIPTION
T_0	First, we must reserve a physical slot in the server by executing a VSORA script "create_instance.py" (this script calls the Amazon API). This operation checks the availability of the desired instance (this has never been an issue) and boots the system on the server side. It takes approximatively 10 seconds.
T_1	At this time, we launch the application and we process the first data (image). We observe a long time before the system returns the result. We do not know exactly what is going on (this comes from the internal management by amazon web service). We analyze it as a delay to fill the internal cache of the machine: indeed, in the case of an AI-DSP IP, the AMI weight is approx. 30 GBytes (a lot of libraries are necessary to run the application) and the delay to process the first image is 2 min. When running the FPGA in another context (classic DSP, no neural network), the AMI weight is approx. 4 GBytes and we do not observe an extra time for the first processing.
T_2 , T_3 , …	Once a processing is ended, the application launches a new processing on the next data. The time of processing is constant (approx. 1 second).

The following table compares the different processing delays for 1 image on the development platforms (measured in a steady state)

PLATFORM	PROCESSING DELAY	REMARKS
NATIVE	8 sec	Delay is acceptable but this model run unquantified data and we cannot estimate the DSP load (memory, number of cycles of processing)
HIGH LEVEL	9 min	Delay is long, exact result for quantization study, approximation for DSP load (number of cycles of processing, 10% error margin)
RTL (VERILOG)	6 hours	Delay is prohibitive. Exact DSP load. This model allows DSP simulations with other configurations (number of base units)
REMOTE FPGA	1 sec	Delay very short, even with the overhead of cache load. Exact DSP load (cycles + memory). The drawback is that the configuration of the DSP IP is fixed (2 base units)

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

Conclusion

The remote FPGA platform is useful in a certain context of use: for an algorithm study, for example a neural network operating the image classification, it is necessary to run it over hundreds or even thousands images to get a reliable result. This platform also gives confidence in the RTL model of the chip since it is the base material of the synthesis. However, for a complete validation in a given configuration (e.g. 64 base units), the RTL simulation or the use of emulation platform (like the Palladium platform proposed by Cadence) is still necessary (but with another range of price).

On the other hand, this platform cannot be used to study the DSP architecture definition: the IP configuration in the FPGA is limited (number of base units is maximum 2), and it is necessary to place and root another database each time the quantization changes. In this case it is better to use the high level platform which gives realistic results.

Simulation of a MIMO decoder

The object of the study is to explore a new way to resolve the decoding of a MIMO (Multiple Input Multiple Output) channel using the AI processing. This is a preliminary work to the next generation of communication stantard (6G).

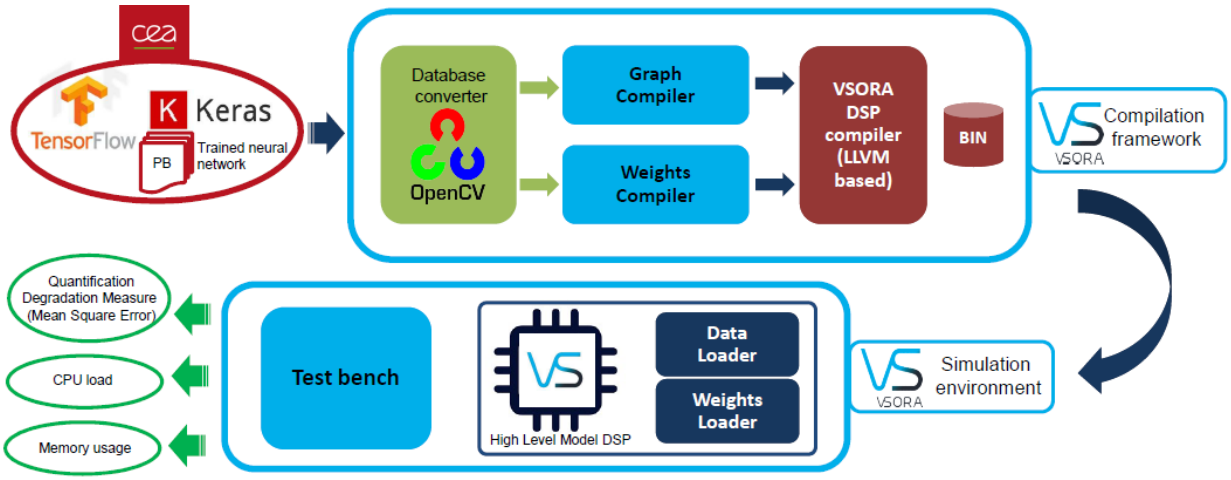


Figure 14: Processing flow of the MIMO decoder

In this study, the CEA has developped a neural network using the standard framework TensorFlow. The target was to act as commercial relationshipt between CEA and VSORA who provided the DSP (hardware + libraries + methodology) where this neural network would be mapped.

In particular, we put to the test our release process, we improved drammatically our documentation in order to the “customer” to be autonomous as far as possible. We also stressed our support methodology using the mantishub platform (web interface to log bugs). Of course VSORA supported the CEA in the implementation of the network.

The neural network was chosen and already trained before the joint work. The purpose of the collaboration was to map this processing on a real target, and to use the simulation plaforms to study the influence of the quantization (Post Training Quantization) on the decoding performances and the influence of the capacity processing on the throughput.

The collaboration was successful, we managed to map very quickly the neural network on the DSP using VSORA’s libraries and methodology.

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

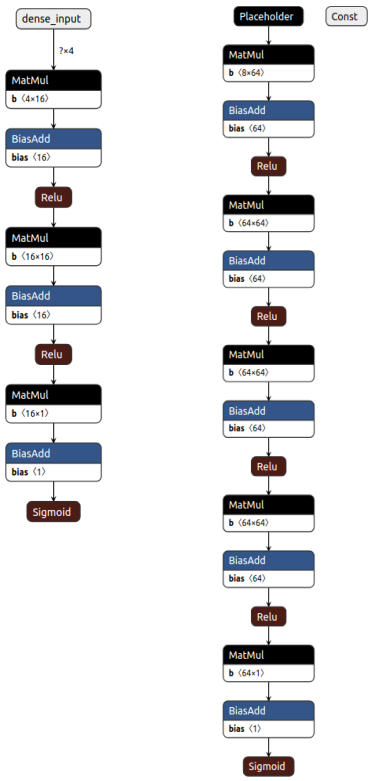


Figure 15: Neural network user for MIMO4x4 (left) and MIMO8x8 (right)

The neural network developped by the CEA for the MIMO decoder is of type fully connected.

It has very few layers. A layer is a processing applied on input and whose output will feed the next layer. Basically, in this neural network, we have 9 layers for MIMO 4x4 and 15 layers for MIMO 8x8. VSORA's graph compiler has the ability to merge these layers into to address all procesing stages of the DSP (typically, activation function can always be merged) and that's why finally, will not have 9 (resp. 15) steps of processing.

A better way to charachterize the complexity of the neural network is to determine the number of MACs. A MAC (multiplication / accumulation) is the base operation used in most of the layers like MatMul or conolution. For both networks we have:

- MIMO 4x4: 33578 MACs
- MIMO 8x8: 1285635 MACs

These numbers are very small: in comparison, the ResNet50 v1.5 has a complexity of 4.09 GMACs (= 4.09×10^9 MACs) (in other words , the neural network for MIMO4x4 is more than 100.000 x less complex than the ResNet50 v1.5).

Moreover, the input data used for the network is a vector of size (4x1) for MIMO 4x4 or (8x1) for MIMO 8x8 (4 and 8 elements respectively): we gather inputs of each antenna in a container. In comparison with ResNet50 v1.5, input is an image of size (224 x 224 x 3) (the dimension 3 comes from the RGB parameter) = 150.528 elements.

With this example, we understand well that these 2 types of neural networks (MIMO and ResNet50 v1.5) cannot follow the same management of the processing.

For the MIMO neural network, it is necessary to create batch processing: theoretically, the batch size should be adapted to the processing capacity of the DSP. In the study conducted by the CEA, we chose to keep the same batch size (32000: input data are matrices of (4x32000) or (8x32000)) for all configurations of DSP: by varying a single parameter in simulation (here the processing capacity of the DSP or the quantization) it is easier to compare the results. This was also possible because the delay introduced by the batch processing was acceptable for all configurations (even for small DSPs, where the latency was the highest).

Regarding the weights of these neural networks, the training process gives the following results:

- MIMO 4x4 weights = 369 words
- MIMO 8x8 weights = 13121 words

These are very small data vectors. We decided to store the weights into the TCM (load from DDR at startup): in a real project we would have choose this solution because the impact on the silicon area is not significant and it reduces dramatically the software complexity of the on the fly load management.

As presented, the purpose of the study was to analyze the influence of the quantization on the results and also to determine the throughput obtained according to the processing capacity. These simulations do not fit on the remote FPGA platform (the DSP configuration is fixed as explained previously in the document). That's why all simulations were ran using the high level platform: the pattern lengths allow to have reasonable simulation time. The results of this study were the subject of the joint publication (CEA/VSORA) of a scientific paper presented at the VTC Fall 2022 (Vehicular Technology Conference) and are summarized in the report D2.4 for the CPS4EU project.

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

2.8. *Overall Conclusion*

The CPS4EU project allowed to develop an advanced DSP for the AI processing. This DSP is proposed as an IP and it is highly configurable: we can choose the number of processing units, the quantization of the data, the internal memory size, the interface with an external memory.

The DSP has been developed having continuously in mind the goal to propose a new development flow: traditionally teams developing the algorithms are different from the teams porting the embedded code on the real target. Now, with VSORA's development flow, the signal processing engineers can evaluate very quickly the impact of their algorithm choices on the DSP load (number of cycles to execute this processing) and on the memory usage (internally and externally). This methodology improves dramatically the time to market of a new product, and helps in the definition of an efficient architecture for a specific application.

This new methodology is based on various development platforms and its philosophy is similar to the PiArch concept declined in the CSP4EU project. One of these platforms uses remote FPGA using the Amazon Web Service framework (AWS). Amazon was the pioneer in this service and if now other actors also propose similar offers, we still do not see European companies on this market (competitors are mainly Chinese). The main usage of this platform is intended for the long simulations necessary to validate complex algorithms especially using the AI features (like the image classification process) because it considerably reduces the simulation times.

The CSP4EU project offered also the opportunity to create an exhaustive documentation and a complete database of regression tests which improves the reliability of the DSP. This was validated by the collaboration between VSORA and the CEA by implementing on a real target a new algorithm for the MIMO channel decoding which will be used in future communication standards.

Although we did not have the opportunity to use the remote FPGA platform during this collaboration (the scope of the study didn't fit the purpose of this platform), this collaboration was still successful: we achieved to map the neural network proposed by the CEA on VSORA's DSP. Moreover, we run simulations for the study of the quantization impact on the decoding performances and the study of the throughput obtained with different processing capacities.

Finally, the collaboration between the CEA and VSORA allowed publishing a scientific paper presented at the VTC Fall 2022 and exposing the common work executed during the CPS4EU project.

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

3. TIME-SENSITIVE NETWORKING

3.1. *TSN Evaluation Platforms*

We survey existing solutions for the simulation and analysis of TSN protocols. Simulation and analysis are a family of techniques dedicated to the temporal evaluation of a complex technical solution ahead of its implementation. This can be extremely beneficial when they use a small fraction of the implementation cost, as they often display, and possibly explain shortcomings of the technical solutions, or even complete inadequacy of such solutions. The early discovery of these issues reduces the risks of wrongheaded choices. Using those solutions along the development process also provides early performance characterization and helps detect implementation errors.

We also present some tools that are already available for developing TSN systems. As the required TSN-compliant electronic components are now available, the associated development boards and lab testing equipment, both hardware and software, are also available. We list the ones that we are aware of, along with our evaluation.

We first describe simulation, then analysis techniques and solutions, and finally the development solutions.

3.1.1. Network Simulation

Software simulation is a large domain of activity, dedicated to modelling the dynamics of physical systems. A model defines variables, which evolution describes that of the system. We focus naturally on the simulation of computer networks. There are mainly two paradigms of software simulation.

- Discrete-Event Simulation (DES) is appropriate when variables change between specific values (e.g. integers) at given points of time. The change of a variable targets an event, which affects other variables after a given delay.
- Continuous simulation is appropriate for systems whose variables vary continuously (e.g. through real numbers). This simulation is often described by differential equations connecting the modelling variables.

When considering the physical layer of a network, one has to use continuous simulation to validate the modulation techniques. For example, the electromagnetic compatibility of a link with its environment is modeled by equations in which variables change continuously.

When considering above layers however (in which TSN is mostly defined), Discrete-Event Simulation is more appropriate. Frames move though communication lines in a predictable amount of time, depending only on the physical data rate. Within switches, they wait in queues, which are very well simulated through discrete events. Therefore, we only consider this model of simulation.

A number of software frameworks support DES. However, we only consider those, which already implement the mechanisms of TSN protocols. Indeed, they are so complex that we do not consider re-implementing the standards within an open DES framework. DES-based network simulation frameworks provide built-in representations of network elements, ready for simulation.

Omnest

Omnest is a commercial DES framework from OpenSim ltd (Hungary), targeting network simulation. It also appears as Omnet++, an open-source version that is dedicated to the education and academic users. It comes with a set of libraries including INET, which models a large number of Ethernet and IP protocols. It has a mixed graphic and textual user interface, as simulations typically require some coding in the dedicated modeling language, or C++. It also offers some graphing capabilities.

On top of Omnest and INET, Nesting is an open-source library which implements a small, but significant subset of TSN. The credit-based shaper (802.1av) and the time-based scheduler (802.1bv) are indeed supported.

Core4INET is another library on top of INET, which also supports the credit-based shaper and the time-based scheduler. In addition, it also supports filtering and policing, defined by 802.1ci.

Recently in 2022, INET has integrated all of TSN, a big step ahead. This makes it a strong candidate for TSN simulation, especially since the academic community will likely adopt it.

While we have not used that last piece, we know Omnest as a mature and dependable tool. It is looks more as a software development tool, than a pure graphical modeling tool. A simulation model is a project, made of several

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

files, using different grammars. This makes it more appropriate for detailed network engineering, than for system engineering.

Pegase

Pegase is a commercial real-time network simulation framework from Real-Time at Work (France). It notably includes a complete support of TSN protocols: time synchronization (802.1AS-2020), credit-based shaper (802.1av), time-based scheduler (802.1bv), frame preemption (802.1Qbu), asynchronous time shaper (802.1Qcr), frame replication and elimination (802.1CB), policing and filtering (802.1Qci) (and possibly others).

It also includes bridging capabilities with other real-time protocols like ARINC, CAN, Flexray. The user interface is 100% graphic and includes several graphing abilities, including chronograms. Live capture files may be imported and analyzed in the same way as simulation results. It also has helpers to simplify the input of complex configurations or suggest an alternate topology, up to the assisted design of transmission schedules.

Being written in Java, Pegase runs on all major OSes. As a commercial tool, its licences are limited in time, in a subscription model. We have found the tool to offer a very large set of functions. We also have found the support to be responsive and able to provide quick solutions and fixes within the product. We have also found the UI to be quite rigid, not very intuitive, and subject to crashes (with no data loss in most cases) and minor display bugs. Execution is reasonably fast, except in some rare cases of very long computation delays.

VisualSim

VisualSim is a commercial simulation framework by Mirabilis Design (USA). It is based on the well-known “Ptolemy II” open-source simulation framework, adding a large number of extensions and optimizations. It supports both DES and continuous simulation, and has a number of optional libraries targeting various markets. In particular, a library supports TSN simulation.

The level of support is said to be very complete, with the complete range of official TSN standards and results display solutions, including time charts. Network simulation is based on DES in the framework. This framework is written in Java, making it portable.

As VisualSim simulates a whole range of domains, it is possible to associate the simulation of networks with that of processors and OSes. It is even possible to run real application binaries on the simulation of the computing nodes exchanging over the simulated TSN network.

We have not used this tool already, only Ptolemy II that we know as a valuable simulation tool.

TCN Analytics

TCN analytics is a commercial network simulation framework by Time Critical Networks (Sweden). Also based on DES, it supports only the main TSN standards. It runs on Windows (Linux possible on demand) and emphasizes GUI simplicity and easiness, e.g. an automated graphical layout, merge of several switches into one, and various results display charts.

TCN Time Analysis has an advanced user interface, which allows easily modelling and handling the network topology and the scheduling of the packet flows. It provides quality results and allows combining different TSN protocols and building heterogeneous network with other protocols (CAN, LIN).

3.1.2. Network analysis

Timing performance analysis is a family of techniques that compute a time limit to the execution of given processes. It is still a topic of research, notably for TSN, as currently no approach supports the whole of TSN. We give here the main approaches before listing the tools.

The Network Calculus (NC) is one of the first developed timing performance analysis approaches for networks (e.g. for AFDX). This approach is based on mathematical fundamentals similar to the algebra (min+) and (max+). The Network Calculus models any element of the network (flow, node, etc.) through curves or functions. The Network Calculus based analysis aims at computing the worst-case end-to-end delay of each flow. Recent publications have shown its application to the main components of TSN, e.g. (L. Zhao, 2018).

The Compositional Performance Analysis (CPA) approach is based on Real-Time Calculus. It is applicable to both processor and network architectures. The compositional analysis constructs a worst-case scenario and then determines an upper bound on the delay, locally in each node crossed by the frame. The upper bound of the

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

end-to-end delay is obtained by summing the different local delays. CPA has been used to analyse a subset of TSN, in (D. Thiele, 2015).

Bordoloi et al. developed in (U. D. Bordoloi, 2014) the first approach to end-to-end delay analysis specific to AVB and its CBS credit scheduling policy. It is defined for the analysis of a single node and was extended to the analysis of the delay across several switches using the principle of the holistic method. These works have been extended to the main component of the TSN network in (M. Ashjaei, 2017).
Other approaches are the Holistic Analysis (limited to simple and small architectures), the Trajectory method (found not to be correct in some cases), and the Eligible Intervals method, supporting only a small subset of TSN on a single switch.

We have found very few tools that support the time analysis of TSN networks, and these tools are less mature than the simulation tools.

Pegase

Aside its simulation features that we presented above, Pegase also provides a worst-case latency analysis function. Integrated within the software and based on the same input models, it provides, for a given network path, a latency time that is guaranteed not to be exceeded.

Pegase uses the Network Calculus approach for computing worst-case end-to-end latencies in a given network. Once the network model is built, analysis is done in one click.

Using the tool, we have found a number of bugs and shortcomings. Real-Time at Work has been providing several corrections, i.e. things are improving with time. For now, we have found that

- In some very simple cases, the analysis is not able to find the real worst case and provides a much higher value instead
- clock drifts are not taken into account by the analyzer.
- The tool forces the synchronization between VLs sent by different nodes.
- No Gantt chart describing the worst-case as the algorithm is not based on the identification of a specific worst-case

Other than that, the same remarks as above apply for Pegase.

TIMAEUS

TIMAEUS is a new network analysis tools based on the Network Calculus approach. It is based on an internal meta-model that is specialized for each network technology. As for now it supports the main TSN components, namely the “bounded low latency” standards IEEE 802.1Qbv (Time-Aware Scheduling) and IEE 802.1Qav standard (Forwarding and Queueing Enhancements) and their combination. It also provides Gantt chart describing the identified worst-case.

We have found the tool to be quite easy to use and intuitive. We have found in some cases bugs in the consistency of the Gantt chart compared to the analysis results.

TIMAEUS also provides a configuration module to automatically generate Gate Control Lists (GDLs) for the use of the IEEE 802.1Qbv (Time-Aware Scheduling).

3.1.1. Development Tools

We list here some equipment that we have found of interest regarding the development of TSN systems. One needs TSN Ethernet switches, TSN Ethernet computing nodes, and TSN measurement tools. Partial compatibility is possible, as non-TSN Ethernet equipment’s can be integrated; for example TSN switches can adapt standard Ethernet flows to TSN. For performance, complete compatibility is better

PC-based development

A PC needs a TSN-compatible Ethernet card. Intel Ethernet controllers I210, I211 and I225 are widely available, cheap and support the main TSN features. Those controllers may also be used on other computers through a PCIe bus; this is notably the case for Raspberry Pi CM4 modules. The I225 provides a 2.5 Gb/s speed, but it has a bad reputation regarding stability (e.g. for the wake-on-LAN feature). Standard Linux configuration utilities support these controllers. We have used them to build working prototypes, but we have found that some key

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

features do not work when setting up a complete TSN configuration. For example, the two main schedulers cannot be combined, and either of them cannot be combined with a master clock. .

As an aside, Raspberry CM4 modules now offer native hardware time synchronization, a key element of TSN.

NXP LS1028ARDB development kit

This development kit is based on the NXP LS1028 microprocessor, which offers several interesting features. It is a powerful dual-core ARM A72 with a set of interfaces, notably one TSN Ethernet controller, and a 6-ports TSN Ethernet switch. There are also 2 TSN connections between the processor and the switch.

This system supports a large set of TSN features. The availability of both a complete TSN switch and computing node in a single package makes the whole very useful. NXP provides a mature Linux-based SDK (now based on Yocto) and notably includes “TSNTool”, a utility for TSN configuration made simpler than standard Linux commands.

IMX-8M+ Development Kits

The iMX-8M+ is the latest addition to the iMX8 family from NXP. It notably includes a neural processing unit, efficient image processors (for capture, compression and display), and two gigabit Ethernet interfaces with TSN support (one with large protocol support).

Several development kits and computer modules are available from vendors. We have notably used one from Phytec. A Yocto-based development software is provided, based on the development software from NXP. However “TSNTool” is not compatible with this microprocessor, as its Ethernet controllers are not the same as on the LS1028.

These systems are relatively cheap and offer a very serious set of features for the requirements of current applications. They are also easier to source than Raspberry boards.

Profishark Ethernet tap

This family of Ethernet taps is available from Profitap (Germany). It is a middle-cost set of tools emphasizing time precision for Ethernet capture with various speeds. Some can be time synchronized with GPS or PPS. They work in connection with a PC host, used to record the captured traffic. Windows is preferred but Linux can be used too, in this case the installation is more complex. Once installed, Wireshark can be used directly to capture traffic. As TSN traffic is standard Ethernet, these boxes can perfectly capture TSN Ethernet traffic. In particular, traffic preemption is said to be decoded. The following limitations apply:

- These tools do not understand PTP, i.e. they cannot stamp frames based on one of the system’s PTP clocks
- They know nothing about the traffic scheduling of the system, therefore the correctness of the traffic w.r.t. this scheduling has to be worked out with extra analysis tools based on the recorded PCAPNG files.

TSN Box

This instrumentation tool from TSN Systems (Germany) can be used mainly in two ways: as an active Ethernet tap, and as a traffic generator. It provides 8 Ethernet ports to tap 4 connections at a time. 4 of them are automotive 100Base-T1 ports, two are 1000Base-T ports, and the last can be either a 1000Base-T or a 1000Base-T1 port. “BaseT1” ports work on a single twisted copper cable along the 802.3bp/bw standards, reducing cable cost, weight, and connector complexity.

We have mostly used the box as a tap. In an automotive context, it was able to detect easily a pre-standard Broadr-Reach connection. It is able to synchronize with the PTP clock of the devices under test. Boxes can also provide the PTP clock, and be chained to synchronize more accurately together using PPS links.

Specific features are very useful, such as the ability to replay captured PCAPNG traffic, possibly with dynamic resynchronization. Boxes can also generate traffic along the main TSN standards.

The boxes provide a Web interface for configuration and can be used as taps from Wireshark, running on Windows or Linux. The most efficient use it to use them from the TSN Tool software that is also sold by TSN Systems, which provide a more synthetic view on the traffic.

Other Tools

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

The following are other tools that we have not tested:

- The Relyum RELY-TSN-PCIe 4-ports TSN Ethernet card connects on a PCIe bus in PCs or other computers. It is based on a completely different solution than Intel's (FPGA running an IP from SoC-e) and can be used as either multiple Ethernet ports or a switch. It supports a large set of TSN features. This solution seems to be also developed by other vendors, and the SoC-e IP is recommended by Xilinx.
- We are also aware of TSN IP from Fraunhoffer institute, also supporting many TSN features.
- The Microchip EVB-LAN9668 switch is based on the LAN9668 component from Microchip. While the component supports a 2.5 Gb/s on some ports, only 1 Gb/s is available on the switch. Many TSN features are supported.
- Other significant component manufacturers such as T.I., Renesas, Marvell, Broadcom have also announced or produced either switch or interface controller components.
- Hilscher netANALYSER is a passive Ethernet capture tool, focusing on precise time measurement on up to 4 inputs. It is limited to 100 Mb/s connections.
- Keysight provides a large catalog of network instrumentation tools, mostly acquired from Ixia Networks. One notably uses them to qualify TSN-labelled equipment such as switches.

3.1.2. Conclusions

Network simulation and analysis are complementary approaches regarding the temporal evaluation. Simulation provides insights on the internal functioning of the system, and shows how it behaves in most cases. Analysis gives hard worst-case guarantees, often required for safety-critical systems. Simulation provides realistic run-time behavior but it is optimistic in nature, as all possible run-time scenarios cannot be run, except for very simple systems. Analysis provides a time limit for all possible scenarios. It is pessimistic in nature, as a run-time scenario (except, again for very simple cases) does not necessarily reach the time limit. Finally, one generally finds simulation less CPU-hungry than analysis.

We have found quite mature solutions for developing TSN systems. NXP has been providing working solutions with both switch and node components. Others (Microchip, SoC-e, Fraunhoffer) support working solutions. Finally, PCs themselves can be made, quite cheaply, part of a TSN network. Measurement tools should be taken into account, to understand the inner behaviour of the network.

3.2. *IEEE 802.1AS Time Synchronization*

3.2.1. IEEE 802.1AS and Precision Time Protocol

Precision Time Protocol (PTP) was defined in IEEE 1588 (IEEE, 2020), and was extended in IEEE 802.1AS standard (Std, 2020). Another protocol that was developed before PTP is Network Time Protocol (NTP). However, NTP is mainly used for synchronizing computers with time servers on the internet and Wide Area Network (WAN). As a result, although having similar principle with PTP, NTP generally achieves lower precision, in the order of millisecond. In contrast, PTP has sharper time resolutions, for example, PTP with hardware-timestamping provides nanosecond precision. PTP is used by IEEE 802.1AS to target highly precise network-wide synchronization in TSN. For example, the standard specifies that end-to-end precision is below 1 microsecond over 7-hops network.

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

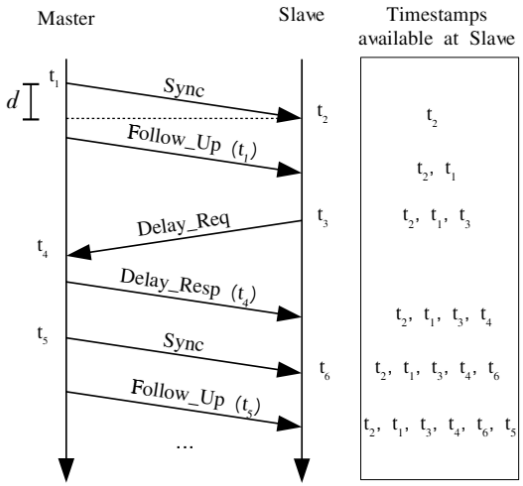


Figure 16: PTP protocol

Figure 16 depicts a network of two End Instances, one acting as master, and one acting as slave. The figure shows how the slaves collect timestamps for estimating the clock offset between their clocks and the GM's clock. The GM takes transmission timestamp and disperses *Sync* messages in order to advertise its clock. Note that with two-step PTP mechanism, the GM sends also *Follow_Up* message to convey the transmission timestamp of that *Sync* message. In one-step mechanism, the timestamp is included directly inside the *Sync* message. Similarly, the slaves send *Delay_Req*, and receive *Delay_Resp* to collect other timestamps.

With the collected timestamps, the slave estimates the peer delay d by averaging the total delay of two-way transmissions, assuming that the link is symmetric. The estimation of d is given as

$$d = \frac{(t_4 - t_3) + (t_2 - t_1)}{2},$$

where t_1, t_2, t_3 and t_4 are transmission timestamps and reception timestamps of *Sync* and *Delay_Req* message, respectively. The time offset δ is calculated as

$$\delta = t_6 - t_5 - d,$$

with t_5 and t_6 are the transmission and reception timestamp of the next *Sync* message, respectively.

3.2.2. IEEE 802.1AS over Wireless Networks

Since the results of time synchronization over wired Ethernet TSN have been showed in the previous version D2.5, this D2.6 focuses only on the synchronization over wireless networks, i.e., Wi-Fi and 5G.

In wireless links, the estimation of delay d in previous equation is possibly inaccurate due to link asymmetry and clock drift, i.e., clocks advancing at different rates. In real systems, peer delay variation might highly impact the precision of the synchronization. To deal with this asymmetry, a PTP implementation can utilize some mechanisms (e.g., linear regression, PID controller) to prevent slaves' clock from high fluctuation. We show that IEEE 802.1AS encounters several challenges when it is used in wireless networks, mainly due to high delay, high jitter and low accurate timestamping.

Firstly, the precision of PTP relies strongly on delay estimation but delay in wireless networks is less deterministic than in wired networks. On one hand, various factors of wireless links, e.g., noise, interference, shadowing and multipath, can cause retransmission, making delay increase and become less deterministic. On the other hand, unlike Ethernet, wireless links are not full duplex and the medium is shared among multiple devices. Due to this non-full duplex, the uplink and downlink are asymmetric, then the peer delay d is not equal to the average as in previous equation. Moreover, due to the medium sharing, wireless networks need mechanisms for multiple access. For instance, IEEE 802.11 needs collision detection and avoidance mechanism, e.g., Carrier Sense Multiple Access (CSMA). This mechanism potentially introduces more delay to the overall delay of a wireless link. In this case, the overall delay becomes higher and less deterministic.

Secondly, PTP also relies heavily on the accuracy of the transmission and reception timestamps, since accurate timestamps will help the previous equation to estimate the peer delay d accurately. As a result, several Ethernet cards implement hardware-timestamping which has an accuracy lower than a nanosecond. With this capacity,

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

the timestamping accuracy is limited mainly by the quantization of the timestamps only, rather than the hardware itself (A. Mahmood, 2014).

3.2.3. Experiment Setup

Figure 17 depicts the setup of the Wi-Fi experiment. We run experiments for one-hop scenario first, then two-hop scenario. In one-hop experiments, we only measure the synchronization between the Access Point and the Wi-Fi Station. In two-hop experiments, we measure the synchronization from the Wired Station to the Access Point, passing the Wi-Fi Station.

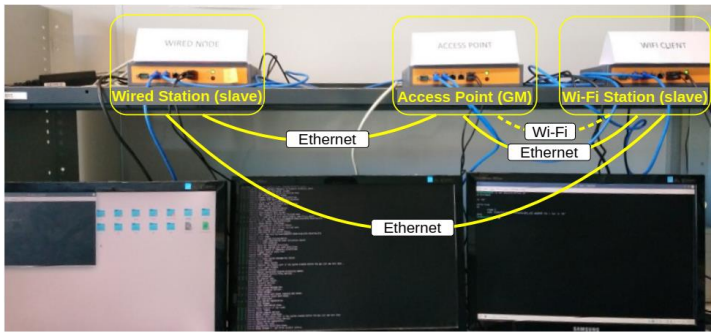


Figure 17: Setup of Wi-Fi experiment

For the 5G setup, we have one 5G User Equipment (UE), one 5G base station (BS), two Wired stations, and two TSN bridges/switches (Figure 18). These 5G hardwares were described in our previous work (R. Gerzaguet, 2017) as a 5G-like testbed. The UE and BS, each of them is built from a radio frequency (RF) transceiver and a custom digital board. The RF transceiver is based on the AD9361 Agile Transceiver from Analog Device. The custom digital board is built on the Zynq-045 Xilinx FPGA with a dual Cortex-A9 ARM processor.

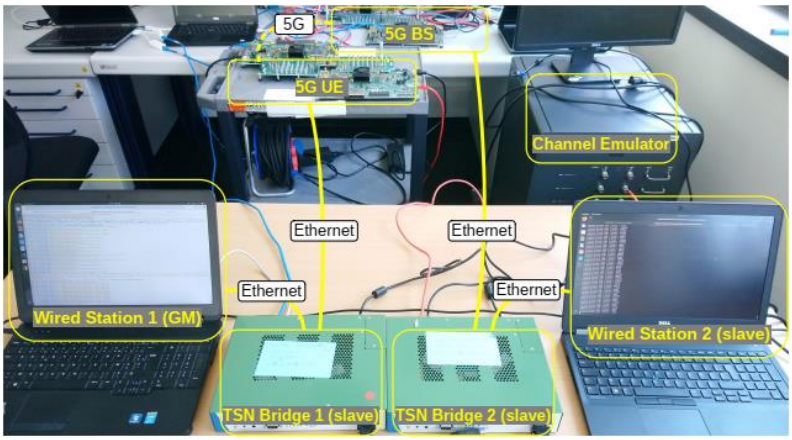


Figure 18: Setup of 5G experiment

3.2.4. Experiment Results

In the one-hop experiments, time synchronization is run over only Wi-Fi link. We run the experiment for 10 minutes and the results are showed in Figure 19. The synchronization takes around 10 to 20 seconds to startup, so this period shows no result in the figure. The offset between the Access Point and the Wi-Fi Station is in the order of hundreds millisecond, compared to the Ethernet hardware-timestamping in previous version of this document, the accuracy of synchronization over Wi-Fi is significantly lower. This inaccuracy can be caused by less precise timestamps and higher jitter. In case of Wi-Fi, there are even more factors that cause the inaccuracy, e.g., the asymmetry of wireless links, the latency introduced by media access or retransmission. Note that TSN for future industrial communications typically requires an accuracy in the order of microsecond.

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

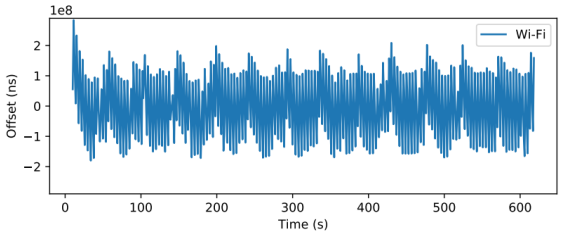
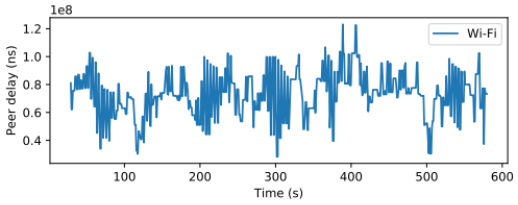


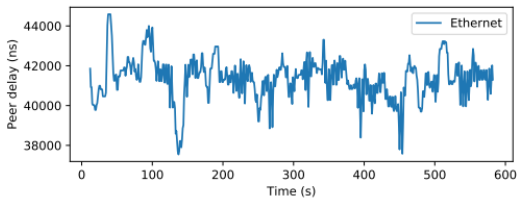
Figure 19: Offset of one-hop synchronization over Wi-Fi

In the two-hop experiments, we evaluate the synchronization from the Access Point to the Wired Station (passing the Wi-Fi Station). This synchronization passes across both wireless and wired links. The Ethernet link from the Access Point to the Wi-Fi Station is not used. The PTP messages are exchanged across both wired and wireless link.

Figure 20 illustrates the peer delay of Wi-Fi and Ethernet in two-hop setup. Note that we use peer-to-peer instead of end-to-end PTP delay mechanism, so that the figure can show separately the delays of Wi-Fi and Ethernet links. We can see that the Wi-Fi link will be the main cause of the degradation in the overall performance because the delay and jitter of this Wi-Fi link are much higher than those of the Ethernet link. The jitter of Wi-Fi link is 1ms, compared to 1µs of Ethernet link. Jitter is an important factor in wireless TSN due to the following. On one hand, it is impacted significantly by the unreliability of wireless link. On the other hand, this jitter impacts negatively the accuracy of PTP, as PTP assumes that the link is symmetric and the peer delay is equal to the average of uplink and downlink delay.



(a) Peer delay between Wi-Fi Station and Access Point



(b) Peer delay between Wi-Fi Station and Wired Station

Figure 20: Peer delay of two-hop synchronization over Wi-Fi

For 5G, we evaluate the synchronization from the GM to the TSN Bridge 2. This synchronization crosses the 5G mobile link. Figure 21 shows that the synchronization over 5G has higher performance than over Wi-Fi. The main reason is that the delay and its variation in 5G (Figure 22b) are lower than in Wi-Fi. An important cause of this difference is the multiple access mechanism of Wi-Fi and 5G. While Wi-Fi uses CSMA for multiple access, 5G multiplexes users in time and frequency without collision in connected mode.

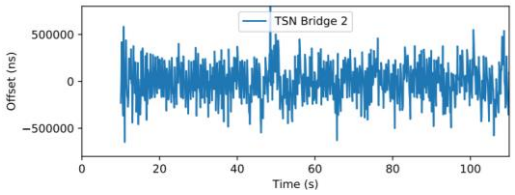
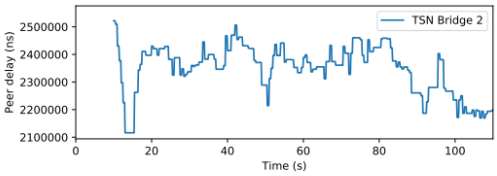


Figure 21: Offset of one-hop synchronization over 5G

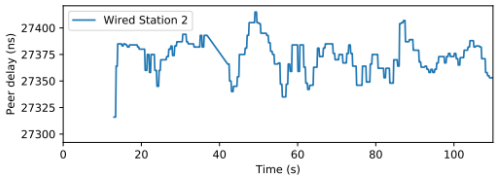
Similar to the Wi-Fi two-hop setup, to observe the synchronization across both wired and wireless links, we collect the results of both TSN Bridge 2 and Wired Station 2 in a same experiment. Peer delays of the two devices are showed in Figure 22. Similar to the Wi-Fi experiment, the peer delay of Wired Station 2 is significantly lower

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

than TSN Bridge 2 because the link between Wired Station 2 and TSN Bridge 2 has Ethernet hardware-timestamping.



(a) Peer delay of TSN Bridge 2



(b) Peer delay of Wired Station 2

Figure 22: Peer delay of two-hop synchronization over 5G

3.3. *IEEE 802.1Qbv Scheduling and IEEE 802.1Qav Queuing Enhancements*

Since we have showed the results of 802.1AS on wired Ethernet TSN in D2.5, therefore this D2.6 provides updated results on two other standards, namely IEEE 802.1Qbv and IEEE 802.1Qav. For simplicity, hereafter these standards might be called Qbv and Qav. We validate the effects of Qbv and Qav schedules inside the TSN bridges and TSN endpoints, then evaluate the performance of these schedules. These scheduling mechanisms are important to guarantee the QoS of communications in CPS. For example, the network flows from a sensor to a robot in a smart factory should be highly deterministic with low latency. Qbv and Qav provide scheduling and queueing mechanisms to address these requirements. Note that for the scheduling of 802.1Qbv to operate correctly, the TSN bridges must keep synchronized with each others using 802.1AS.

3.3.1. IEEE 802.1Qbv

IEEE 802.1Qbv is a TSN standard that targets TSN bridges to perform QoS scheduling. When there are multiple network flows with different QoS requirements, the TSN bridges have to protect the QoS of the high-priority flows against others. The endpoints and bridges that support IEEE 802.1Qbv have minimum two transmission queues per port. In general, each queue can correspond to a traffic class. At a time, the bridge opens some queues for transmission, and other queues have to wait. In this way, the bridges can provide lower latency to critical flows by taking more time to serve these flows. Figure 23 (IEEE, 2015) illustrates an example of Pre-computed Gate Control List (GCL) in Qbv scheduling. The bridge opens and closes its gates based on a cyclic gate control list. Once the end of this list is reached, it is repeated.

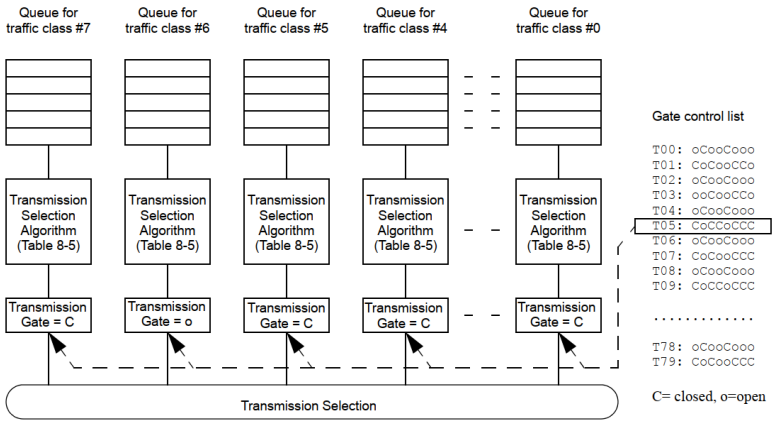


Figure 23: Example of GCL

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

3.3.2. IEEE 802.1Qav

In addition to IEEE 802.1Qbv, the standard IEEE 802.1Qav targets the scheduling on both TSN endpoints and bridges. These nodes have a minimum of two transmission queues (corresponding to two traffic classes) per ports, up to 8 Stream Reservation (SR) classes. The endpoint that acts as a *Talker* sets the Priority Code Point (PCP) field according to the priority of the traffic. The main motivation of 802.1Qav is to deal with the fairness among bursty traffic and non-bursty traffic. When a *Talker* is busy sending a bursty flow, other flows might have to wait for long time. The main principle of 802.1Qav is to maintain a credit value for each flow; when a flow is served, its credit is decreased, otherwise its credit increases. A flow must be served when its credit reaches an upper bound value call *hiCredit*. Figure 24 (IEEE, 2007) depicts an example of Qav with 3 flows and the evolution of the class A's credit. When the credit reaches *hiCredit*, the flow is served, when the credit reaches *loCredit*, the serving is stopped.

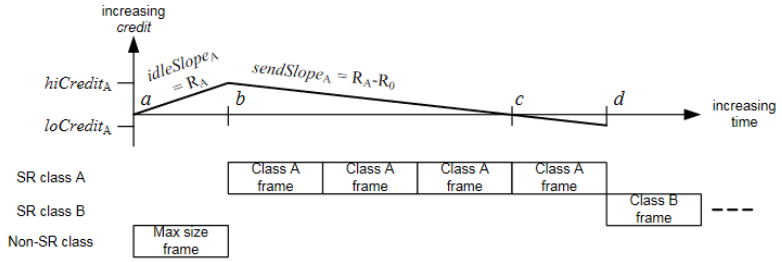


Figure 24: Example of Qav with 3 flows

3.3.3. Experiment Setup

We setup an Ethernet TSN testbed with a *Talker* transmitting traffics to a *Listener*, passing two TSN bridges, as in Figure 25. These devices are connected to each other via Ethernet 1Gb/s. In the scenario of Qbv, the *Talker* sends one QoS flow and one best-effort flow to the *Listener.* The best-effort flow is sent at the rate of 168 Mb/s. The Qbv aims at protecting the performance of QoS flow from best-effort flow. To demonstrate a simple scenario, we define the Qbv schedules as follows. The switches open their gates 80% of the time to serve QoS flow, and 20% to serve best-effort flow. During the experiment, the switches keep synchronized their clocks with each other using IEEE802.1AS.

Figure 25: Ethernet TSN testbed

In the scenario of Qav, the *Talker* sends one audio and one bursty video flow to the *Listener*. The video, in average, has a burst every 0.1 second, and the rate of the burst is 255500 packets/s with each packet contains 1000 bytes. The objective of Qav is to protect the delay of audio flow from the bursts of video flow. The Qav parameters are set as in following table.

Parameter	**Value**
idleSlope	460872
sendSlope	-539128
hiCredit	710
loCredit	-831

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

3.3.4. Experiment Results

802.1Qbv

In Figure 26 and Figure 27, we show the delay of the QoS flow when Qbv is deactivated and activated, respectively. We can see that when Qbv is deactivated, the obtained delay and jitter are high because of the impact from the best-effort flow. Then when we activate Qbv, the delay of QoS flow becomes lower. This result shows that Qbv is able to protect the delay of QoS flow from best-effort flows.

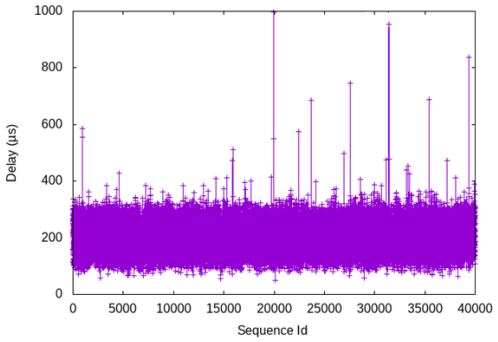


Figure 26: Delay when Qbv is deactivated

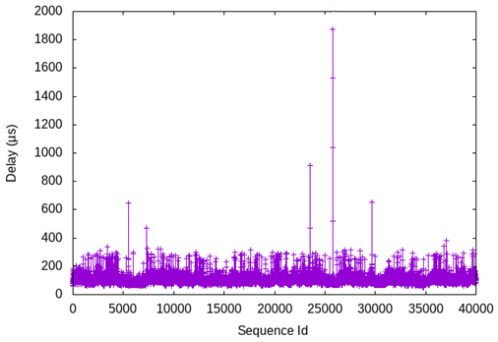


Figure 27: Delay when Qbv are activated

802.1Qav

Similar to the experiments of Qbv, we run experiments with Qav deactivated first and then with Qav activated to see the effect of Qav scheduling. Figure 28 and Figure 29 show the delay of audio and video, respectively, when the Qav is deactivated. For the case of Qbv activated, the results are provided in Figure 30 (audio) and Figure 31 (video). We can see that Qav significantly reduces the delay of audio, while increasing slightly the delay of video. Moreover, the curves of both flows fluctuate less when Qav is activated, meaning that the flows also improve their determinism. This result complies with the main objective of Qav, which spreads out the bursts, in order to improve the fairness among flows, without trying to enhance the quality of the bursty flows.

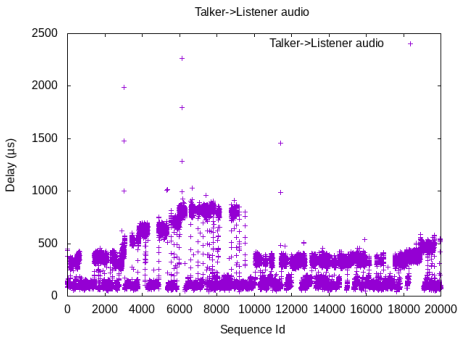


Figure 28: Delay of audio flow when Qav is deactivated

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

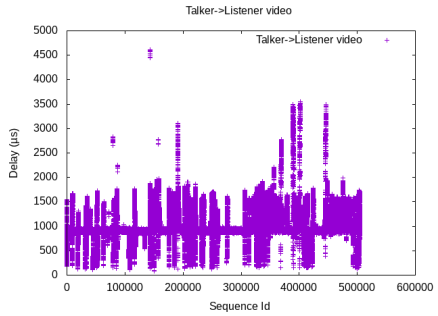


Figure 29: Delay of video flow when Qav is deactivated

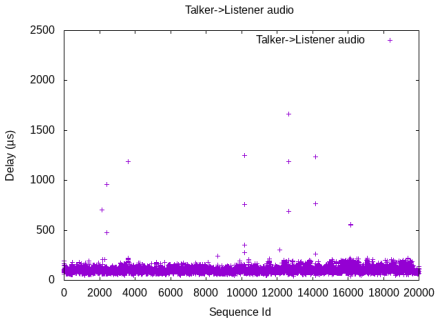


Figure 30: Delay of audio flow when Qav is activated

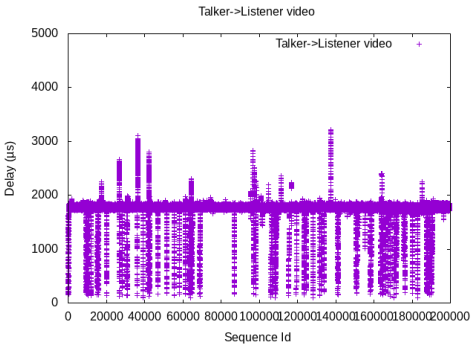


Figure 31: Delay of video flow when Qav is activated

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

4. CONCLUSION

In this document D2.6, we have presented an updated version of D2.5 on the validation and evaluation results of connectivity platforms for CPS. The integration of existing technologies led to the design of the PiArch board. This is also the result of a collaboration with partners working in Work Package 6.

This deliverable also investigates novel enablers for future communication systems beyond 4G, such as a 5G MIMO, IEEE TSN. On 5G MIMO, we 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. This process is built 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.

On the low latency aspect of 5G, we presented wired and wireless TSN testbeds as well as their experiment results. We showed that the time synchronization offset between the network nodes is low on Ethernet and 5G, but remains high on Wi-Fi. To deal with this problem on Wi-Fi, one promising solution is to integrate FTM into 802.1AS. However, we have tested and verified that currently full integration cannot be realized on commercial off-the-shelf (COTS) devices due to the lack of hardware support. For example, the hardware must allow writing *VendorSpecific* field and reading the timestamps inside FTM messages. We also presented evaluation results of 802.1Qbv and 802.1Qav, which show how TSN can support low latency in CPS. The latency and determinism of best-effort and QoS flows (e.g., flows from the sensors to the robots in a smart factory) can be improved using these two standards.

5. REFERENCES

A. Mahmood, R. E. a. T. S., 2014. Impact of hard-and software timestamping on clock synchronization performance over ieee 802.11. *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014),* p. pp. 1–8.

IEEE, 2007. *Forwarding and Queuing Enhancements for Time-Sensitive Streams,* s.l.: s.n.

IEEE, 2015. *802.1Qbv–Enhancements for Scheduled Traffic,* s.l.: IEEE Std.

IEEE, 2020. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2019.*

R. Gerzaguet, S. B. P. R. J. E. X. P. D. D. J.-B. D. B. M. M. P. D. M. a. D. K., 2017. 5G Multi-Service Field Trials with BF-OFDM. *2017 IEEE Globecom Workshops (GC Wkshps),* p. pp. 1–5.

Std, I., 2020. IEEE standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications. *IEEE Std 802.1AS-2020,* p. pp. 1–421.

Yun, J. H. K. W. S. a. D. O. K., 2017. *Method and apparatus for providing in-vehicle network time synchronization using redundant grandmaster,* s.l.: U.S. Patent.

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