Versal ACAP Processing for HL-LHC Calorimeters Signal Reconstruction

27th Conference on Computing in High Energy and Nuclear Physics (CHEP)
24th October, 2024

Francisco Hervas, *Alberto Valero, Luca Fiorini, Hector Gutierrez*

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
Financiado por la Unión Europea
NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
AGENCIA ESTATAL DE INVESTIGACIÓN
IFIC INSTITUT DE FÍSICA CORPUSCULAR
CSIC
VNIVERSITAT DE VALÈNCIA
HL
CHEP 2024

Table of contents

1 Introduction
- LHC calorimeter read-out
- HL-LHC calorimeter signal reconstruction
- Real time processing module

2 Methodology
- Versal ACAP device for algorithms test
- Setup development
- Development flow
- Modified perceptron description
- Modified perceptron RTL

3 Results
- Accuracy comparison
- Multi-core implementation
- Timing comparison

4 Summary
- Summary and future work

1 Introduction

LHC calorimeter read-out

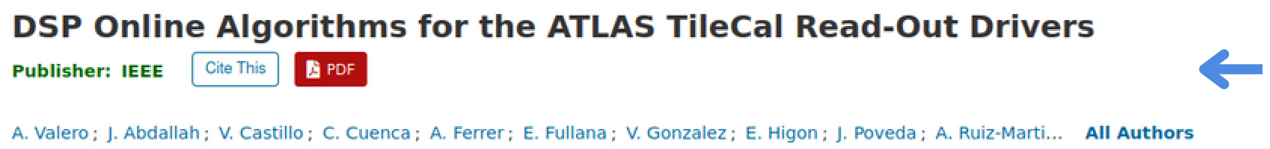
- In the **LHC**, **Bunch Crossings** (BC) happen at **40 MHz** (25 ns)
- The **processing** happens after the **Level-1 Trigger**, at **100 kHz** (10 us)
- Signals are processed online using the **Optimal Filtering** (OF) algorithm
    - The processing is made using **Digital Signal Processors** (DSPs)
    - Therefore, it is **sequential**
    - **Fixed point** arithmetic is used

Optimal Filtering Algorithm

$$A = \sum_{i=1}^{n} a_i (S_i - p)$$

$$\tau = \frac{1}{A} \sum_{i=1}^{n} b_i (S_i - p)$$

$$\chi = \frac{1}{A} \sum_{i=1}^{n} |((S_i - p) - A g_i)|$$

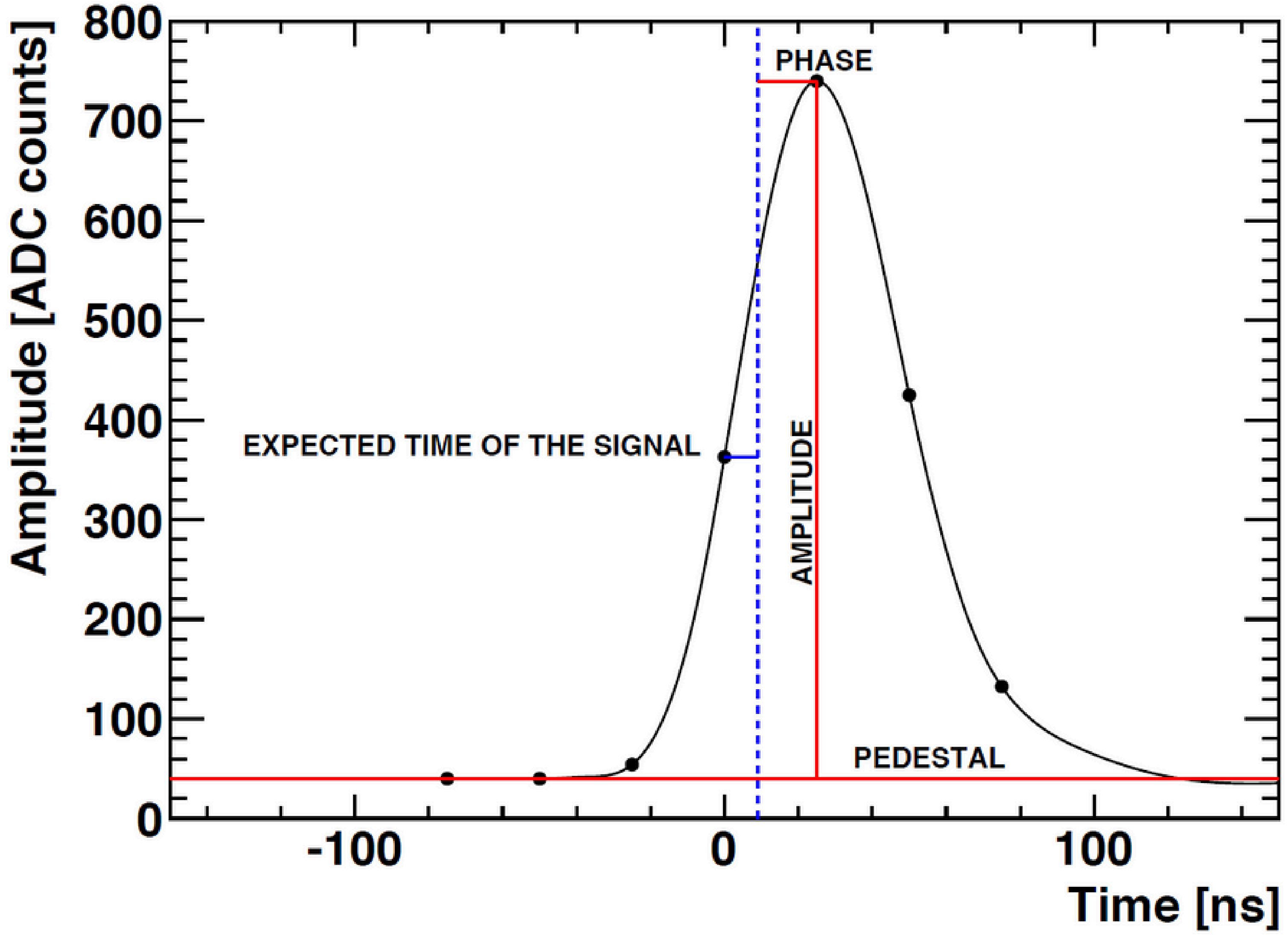


DSP Online Algorithms for the ATLAS TileCal Read-Out Drivers

Publisher: IEEE Cite This PDF

A. Valero; J. Abdallah; V. Castillo; C. Cuenca; A. Ferrer; E. Fullana; V. Gonzalez; E. Higon; J. Poveda; A. Ruiz-Marti... **All Authors**

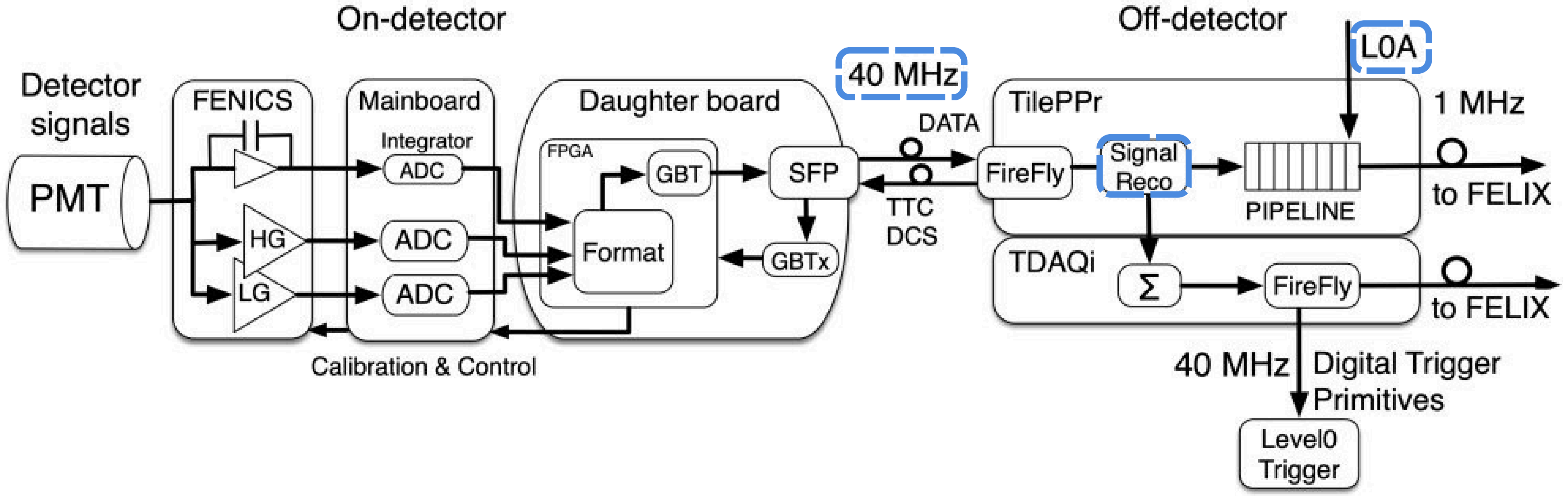
Click on the image for reading more on the topic

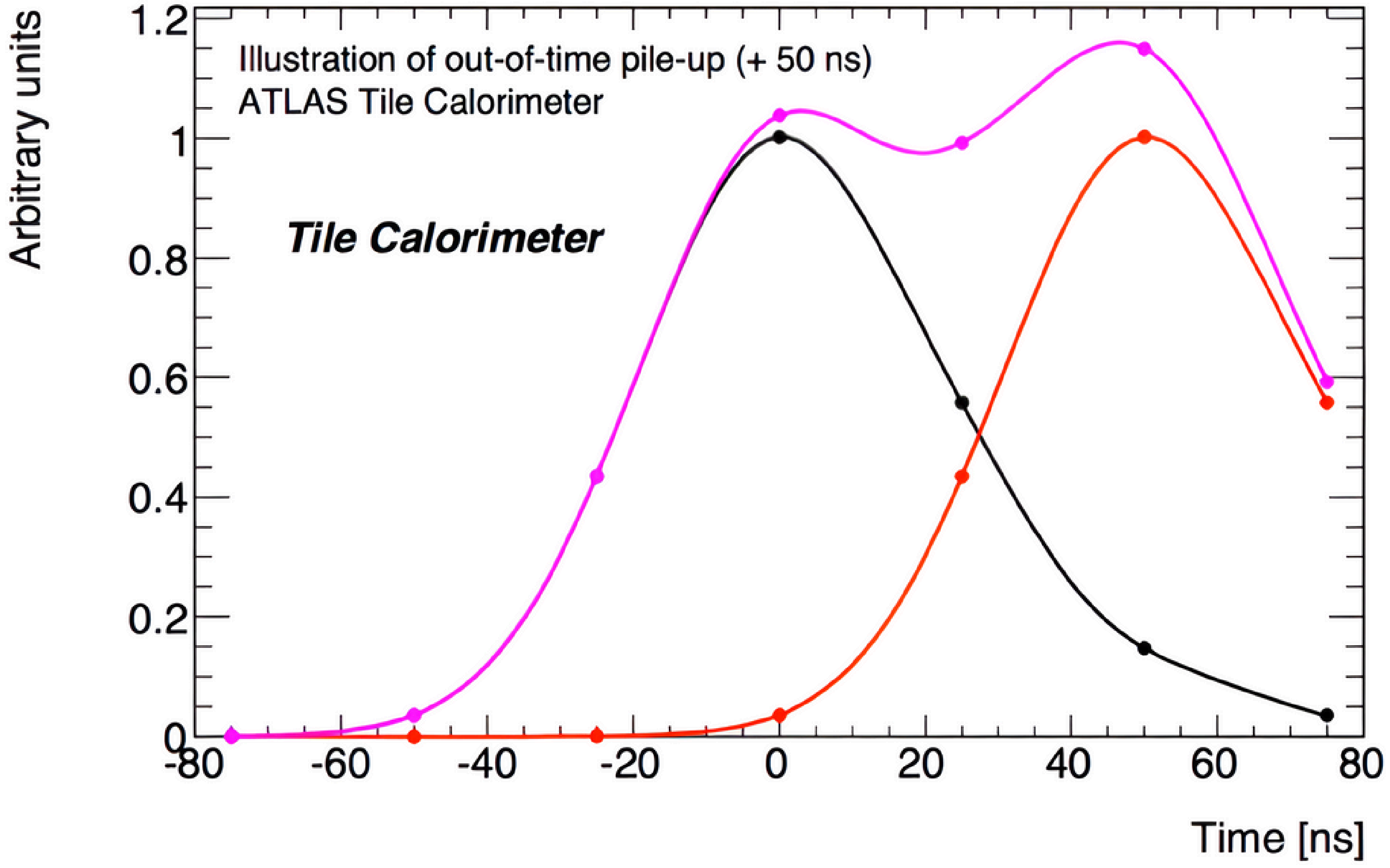


1 Introduction

HL-LHC calorimeter signal reconstruction

- In the **HL-LHC**, signals will be **reconstructed for every BC** at **40 MHz** (25 ns) **before the trigger**
  - Signals need to be processed by **FPGAs** due to their **low and deterministic latency** for signal synchronization
  - Multiple **simultaneous signals** will produce **pile-up**
- There is a need for more sophisticated algorithms for signal reconstruction
  - **Deep learning** algorithms (**Neural Networks**)





1 Introduction

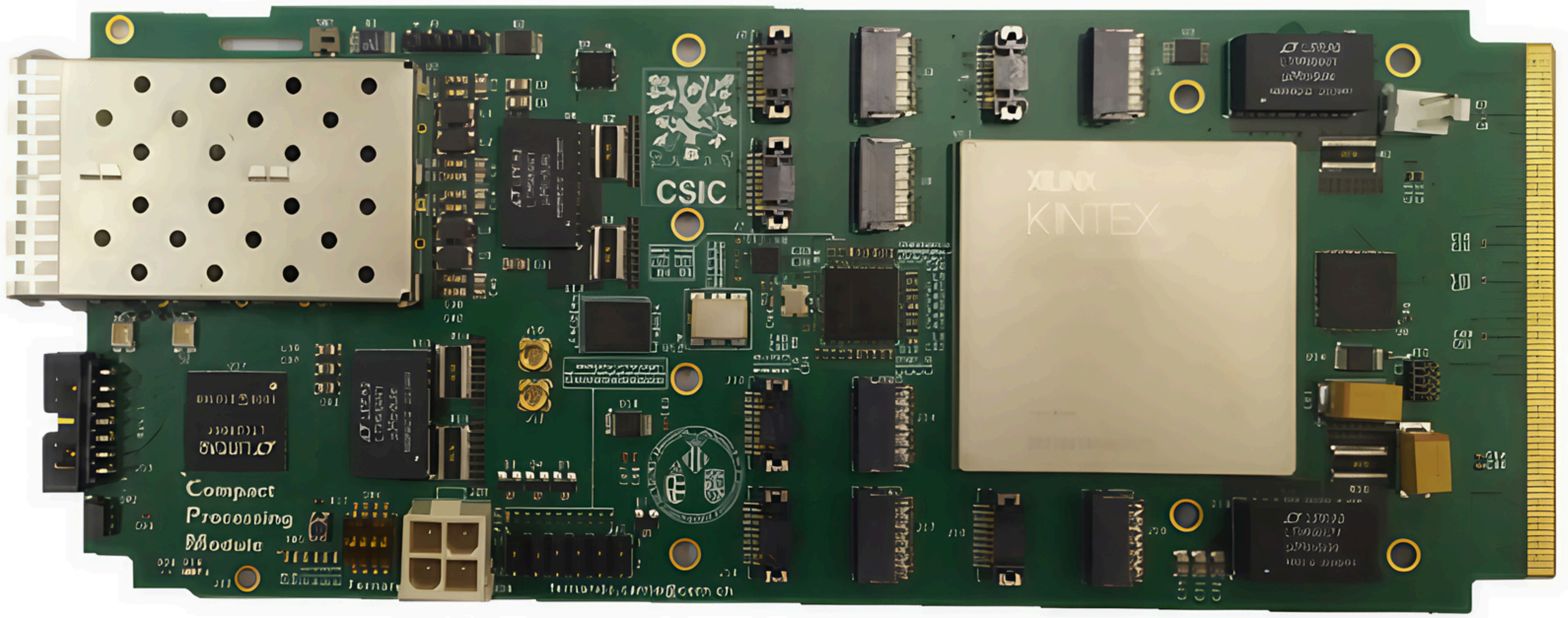
Real time processing

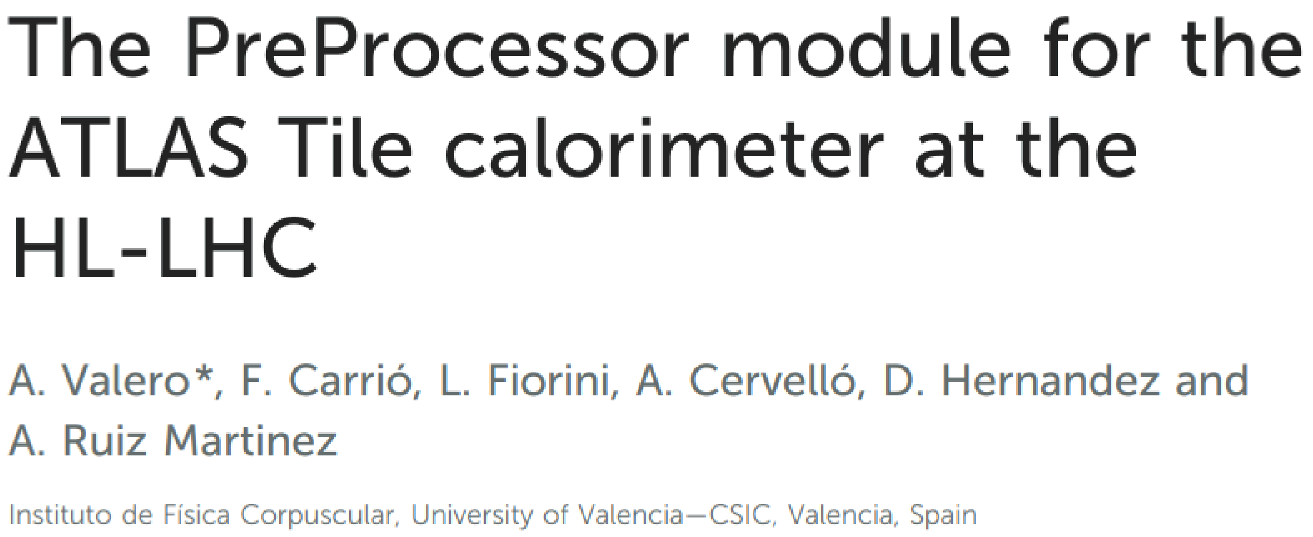
- **Real time applications** fit better with **FPGAs**
- Algorithm replication must fit **area/ occupancy**
- **Cycle accurate in latency** to interconnect modules

- **Real time** requirements:
  - Maximum algorithms in parallel: **162**
  - Maximum sample latency: **200 ns**
  - Sample frequency: **40 MHz**

KU115 Chip resources:

FFs	1326720
LUTs	663360
Block RAMs (Mb)	75.9
DSP Slices	5520




The PreProcessor module for the ATLAS Tile calorimeter at the HL-LHC

A. Valero*, F. Carrió, L. Fiorini, A. Cervelló, D. Hernandez and A. Ruiz Martinez

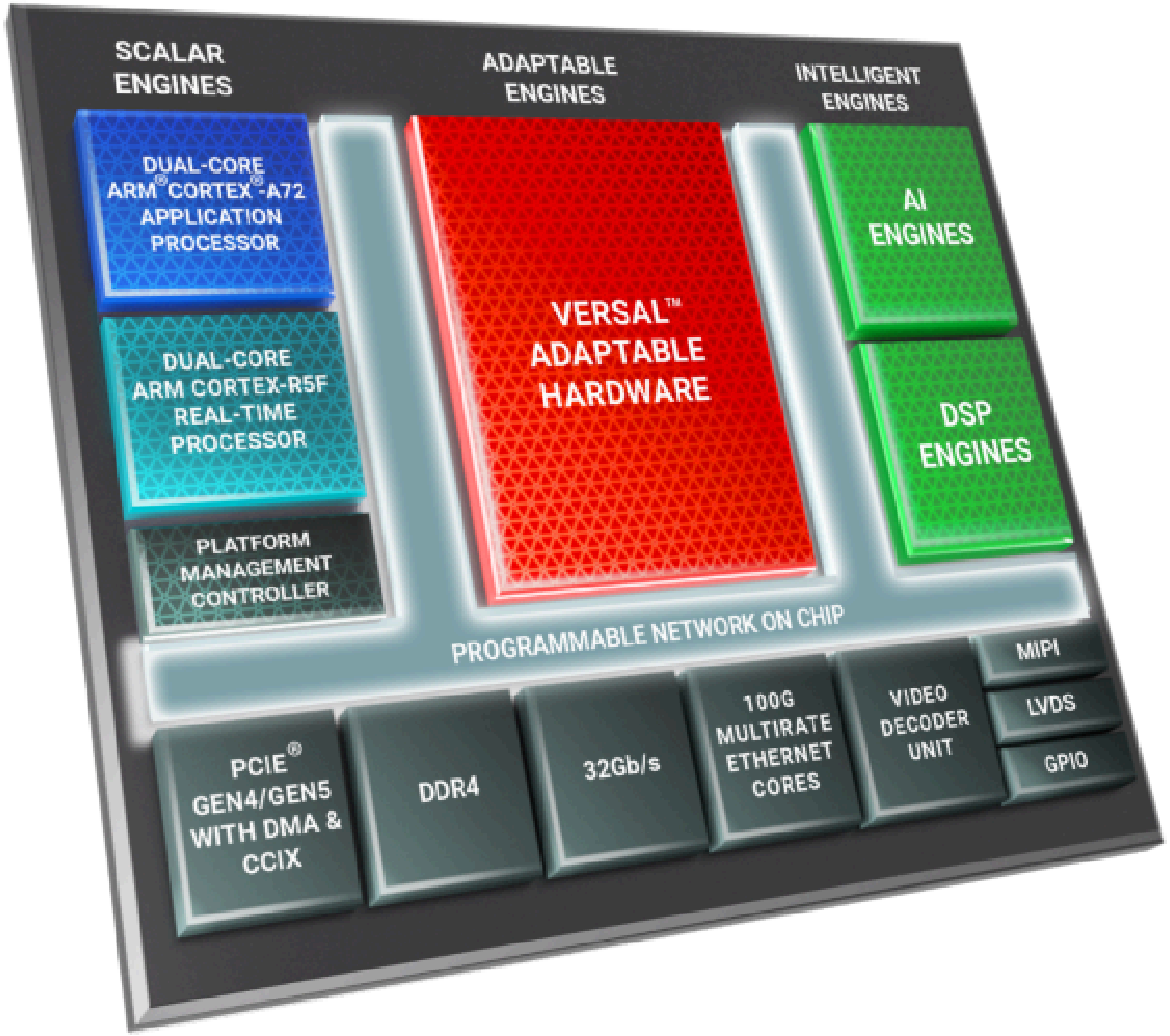
Instituto de Física Corpuscular, University of Valencia—CSIC, Valencia, Spain

Click on the image for reading more on the topic

2 Methodology

Versal ACAP device for algorithms test

- **Versal ACAP** System on Chip (SoC) for **algorithm testing**
    - **PL - FPGA:** Algorithm hardware acceleration and data moving between memories
    - **PS - CPU:** Managing and control of the accelerators
    - **Memory:** Data buffering
    - **Connectivity:** External data reception and data transmission



SCALAR ENGINES
ADAPTABLE ENGINES
INTELLIGENT ENGINES
DUAL-CORE ARM® CORTEX®-A72 APPLICATION PROCESSOR
DUAL-CORE ARM CORTEX-R5F REAL-TIME PROCESSOR
PLATFORM MANAGEMENT CONTROLLER
VERSAL™ ADAPTABLE HARDWARE
AI ENGINES
DSP ENGINES
PROGRAMMABLE NETWORK ON CHIP
PCIE® GEN4/GEN5 WITH DMA & CCIX
DDR4
32Gb/s
100G MULTIRATE ETHERNET CORES
VIDEO DECODER UNIT
MIPI
LVDS
GPIO

VC1902 Chip resources:

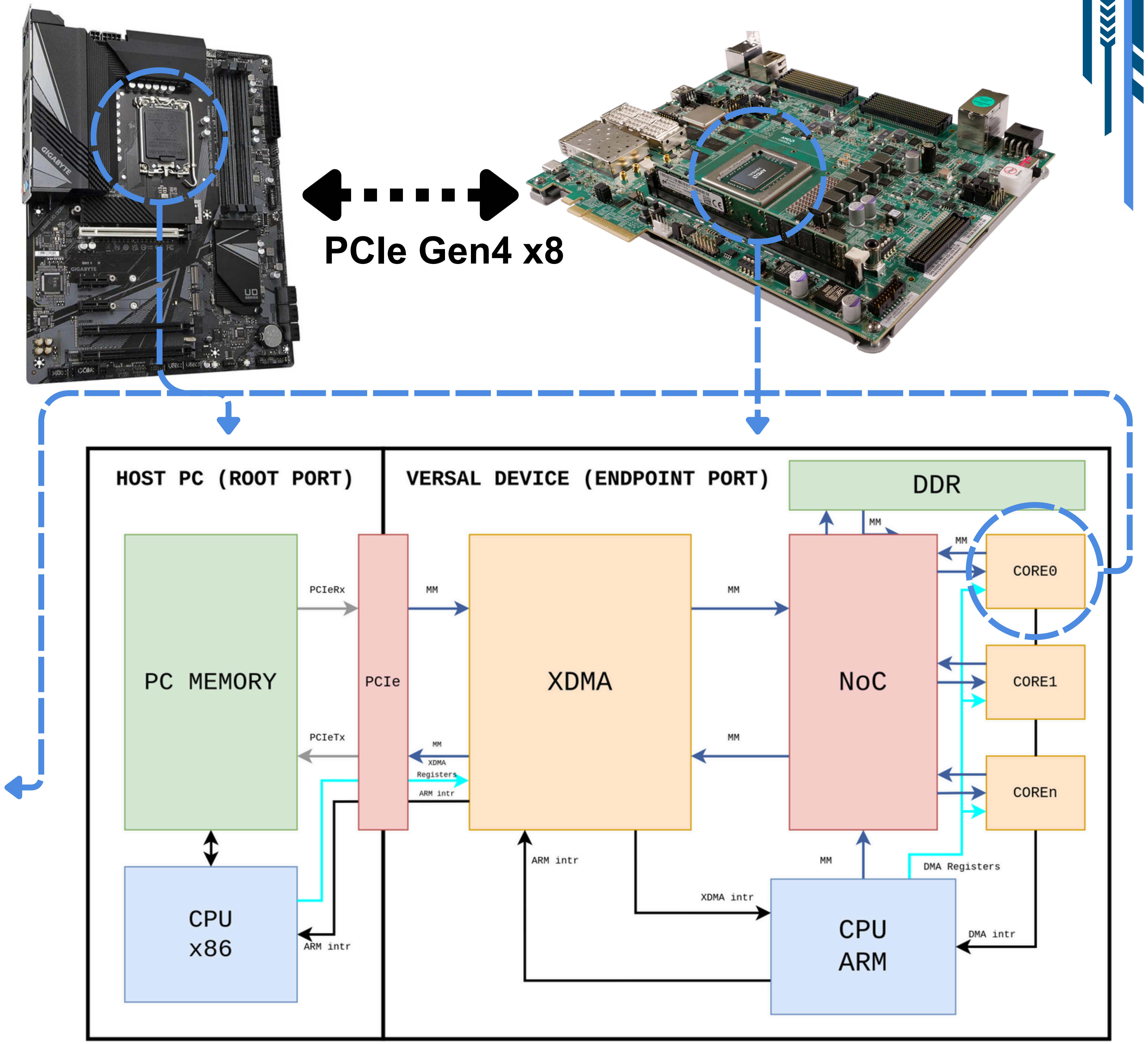
CPU	ARM A72 Dual core
	ARM R5F Dual core
Memory	OCM 256 KB
Connectivity	Ethernet x2
	USB 2.0 x2
	UART x2
	SPI x2
	I2C x2
	CAN-FD x2

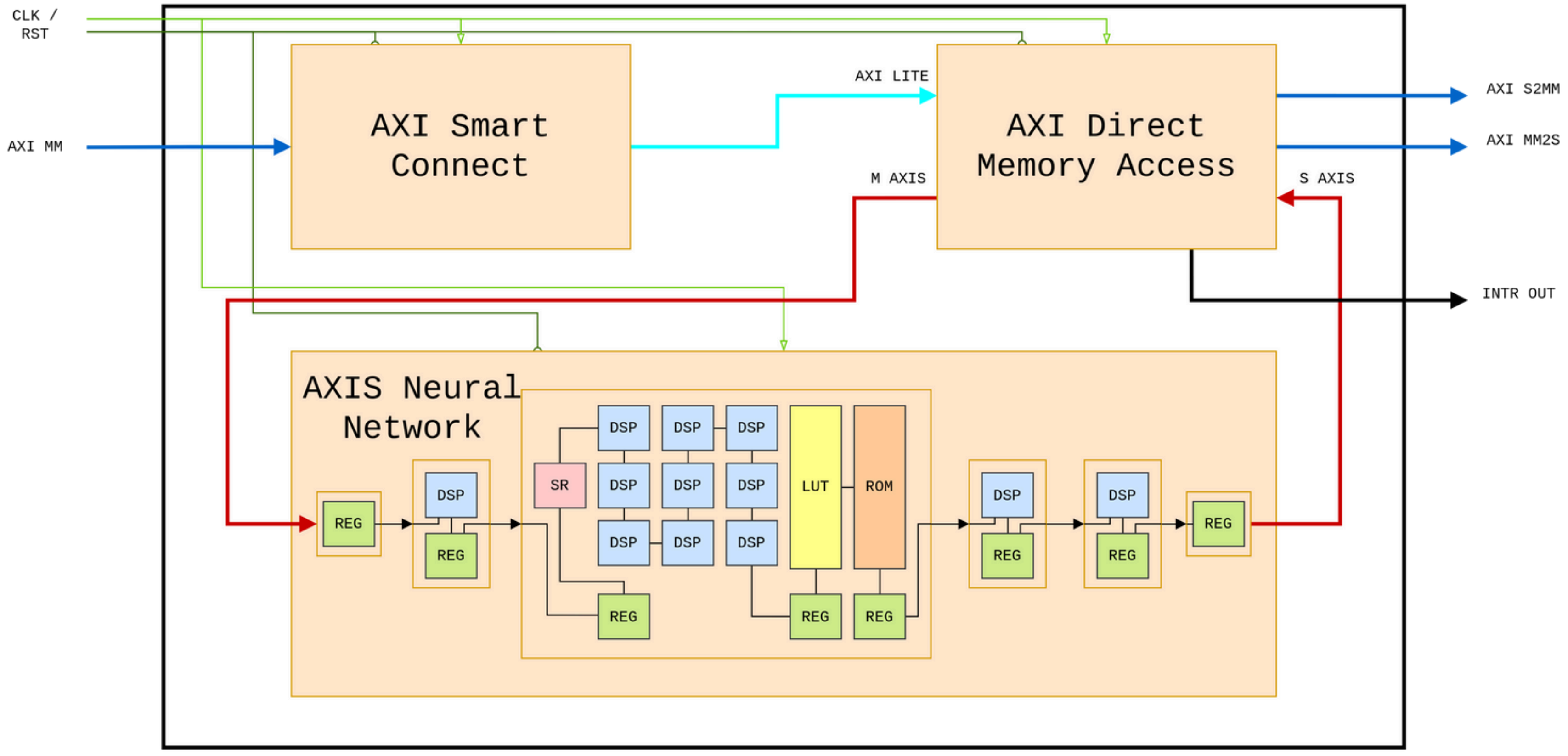
AI Engines	400
DSP Engines	1968
LUTs	899840
NoC Ports	28
DDR MC	4
PCIe – DMA	1

2 Methodology

Setup development

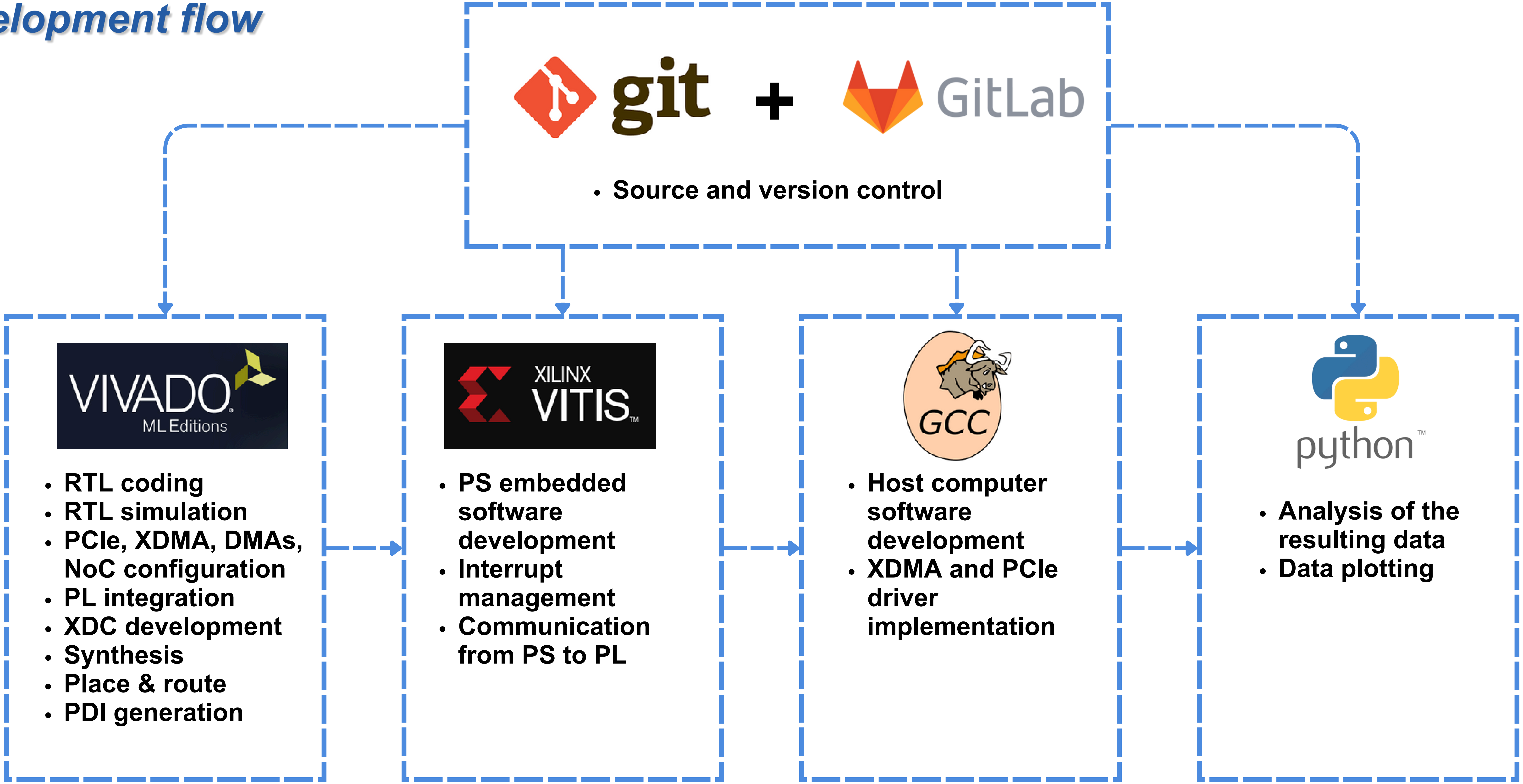
- Driver implementation in host CPU for communication with **XDMA**
- Driver implementation in device CPU for managing **internal DMAs**
- **NoC configuration** for internal communication
- **Interrupt system** development
- **Multiple cores** executing algorithms





2 Methodology

Development flow



- Source and version control

- RTL coding
- RTL simulation
- PCIe, XDMA, DMAs, NoC configuration
- PL integration
- XDC development
- Synthesis
- Place & route
- PDI generation

- PS embedded software development
- Interrupt management
- Communication from PS to PL

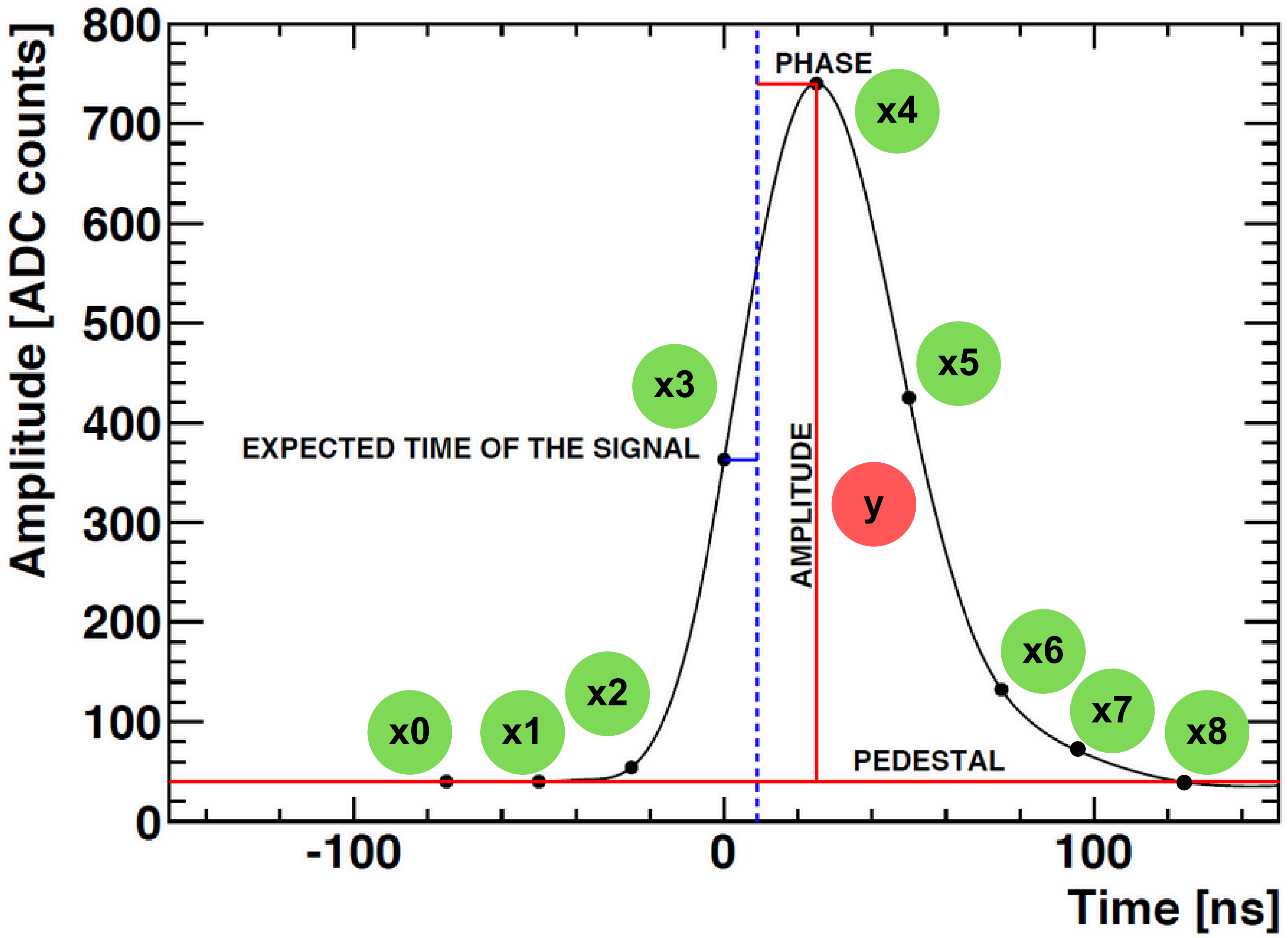
- Host computer software development
- XDMA and PCIe driver implementation

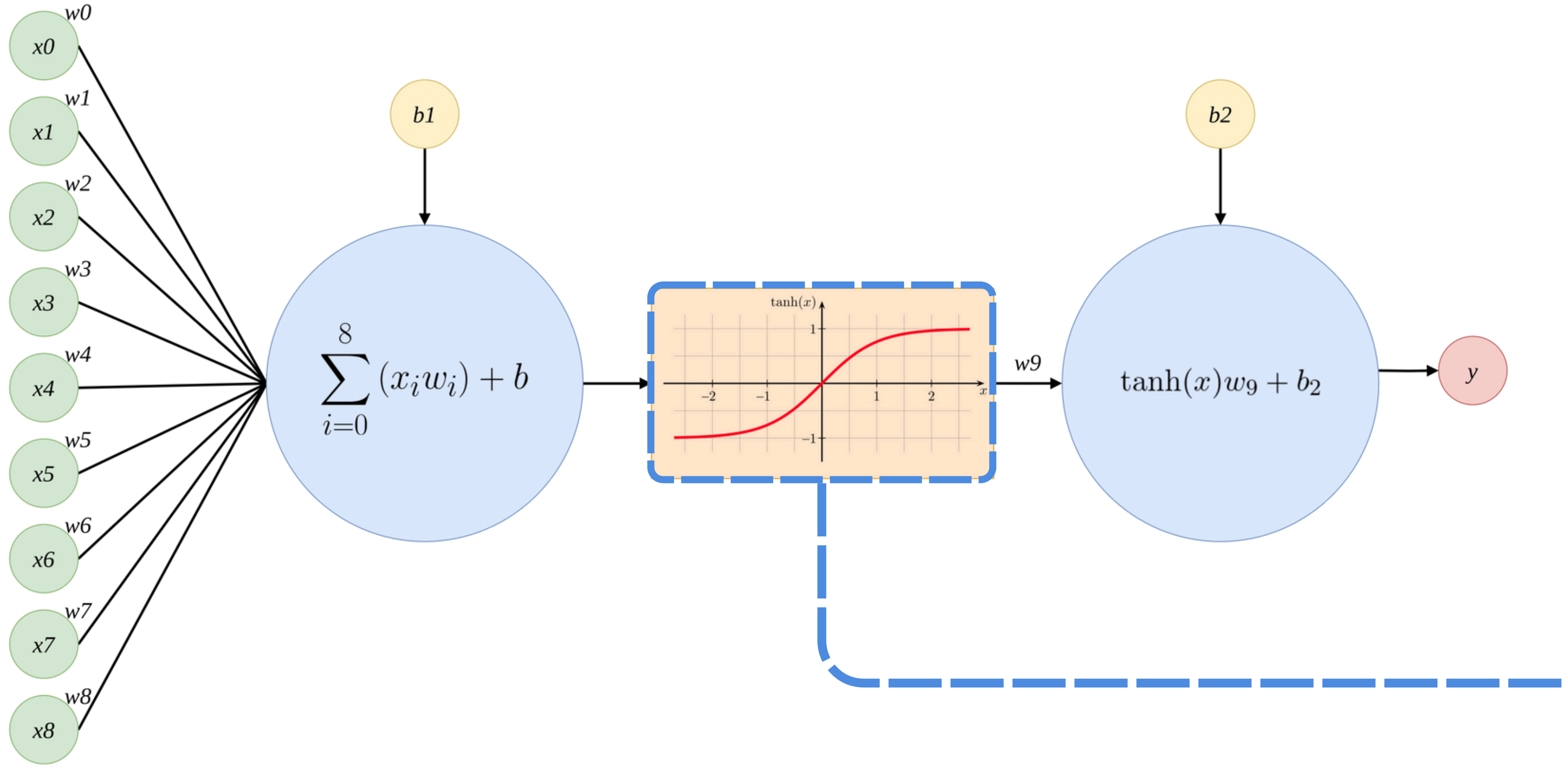
- Analysis of the resulting data
- Data plotting

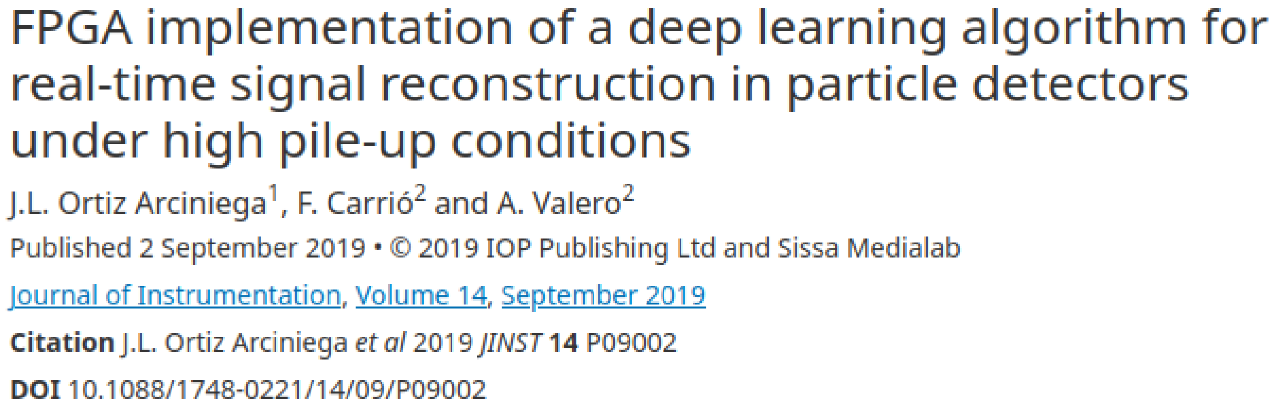
2 Methodology

Modified perceptron description

- Modified version of a **perceptron** with **2 neurons**
- **Hyperbolic tangent** as the activation function
  - Quantized and implemented in a Look Up Table
- Input: **9 BC** sliding window
- Output: **Amplitude** of the **central BC** in each window







FPGA implementation of a deep learning algorithm for real-time signal reconstruction in particle detectors under high pile-up conditions

J.L. Ortiz Arciniega[1], F. Carrió[2] and A. Valero[2]

Published 2 September 2019 • © 2019 IOP Publishing Ltd and Sissa Medialab

Journal of Instrumentation, Volume 14, September 2019

Citation J.L. Ortiz Arciniega *et al* 2019 *JINST* **14** P09002

DOI 10.1088/1748-0221/14/09/P09002

Click on the image for reading more on the topic

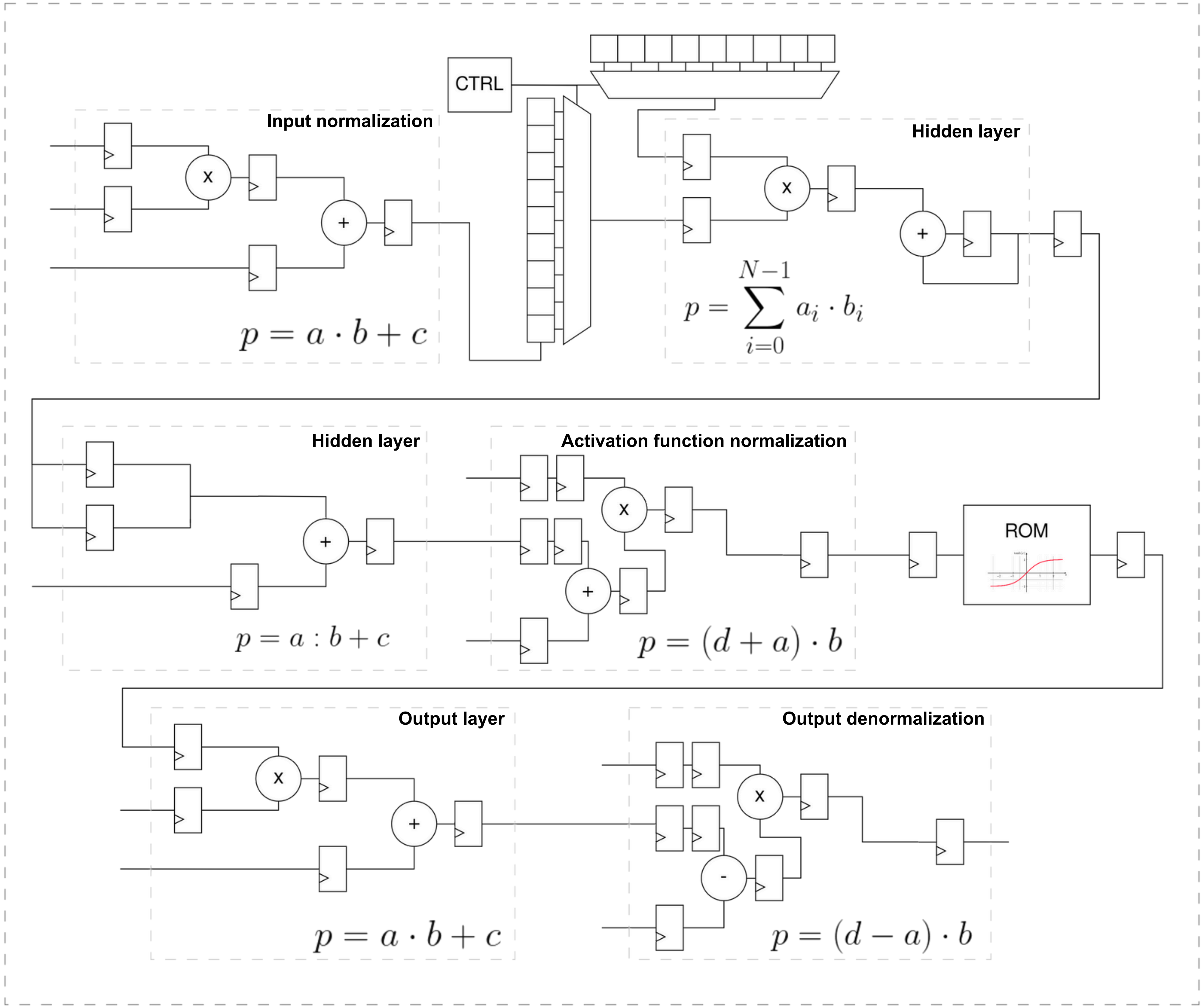
2 Methodology

Modified perceptron RTL

- Written in **VHDL**
- Synthesized and implemented in **Vivado**
- **Fixed point** arithmetic
- Activation function **tanh(x) quantized** over **5000 values**
- **Latency** of **34 clock cycles**

VC1902			
Resource	**Utilization**	**Available**	**Utilization (%)**
LUT	72	899840	0.01%
FF	203	1799680	0.01%
BRAM	4	967	0.41%
DSP58	6	1968	0.30%
BUFG	1	980	0.10%

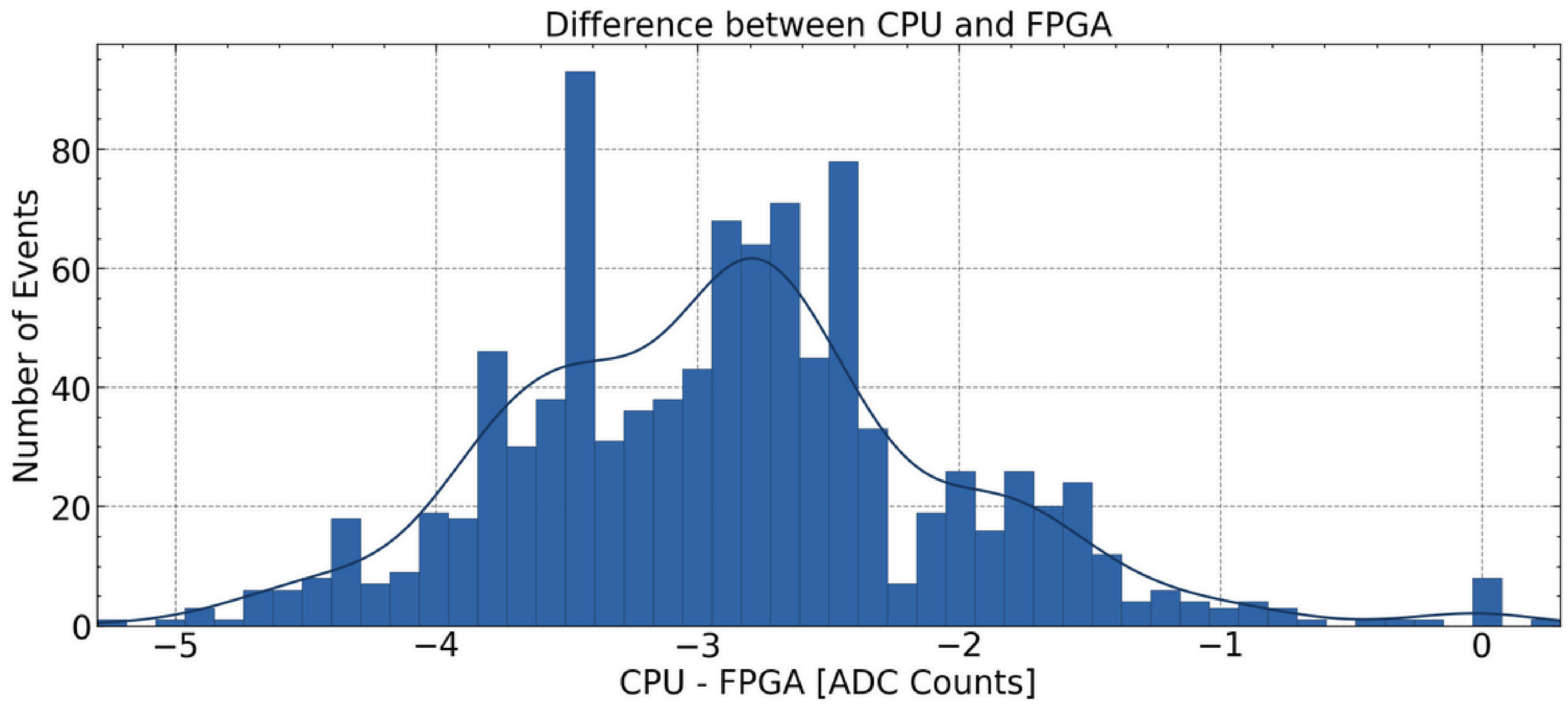
KU115			
Resource	**Utilization**	**Available**	**Utilization (%)**
LUT	65	663360	0.01%
FF	203	1326720	0.02%
BRAM	3.5	2160	0.16%
DSP48E2	6	5520	0.11%
BUFG	1	1248	0.08%

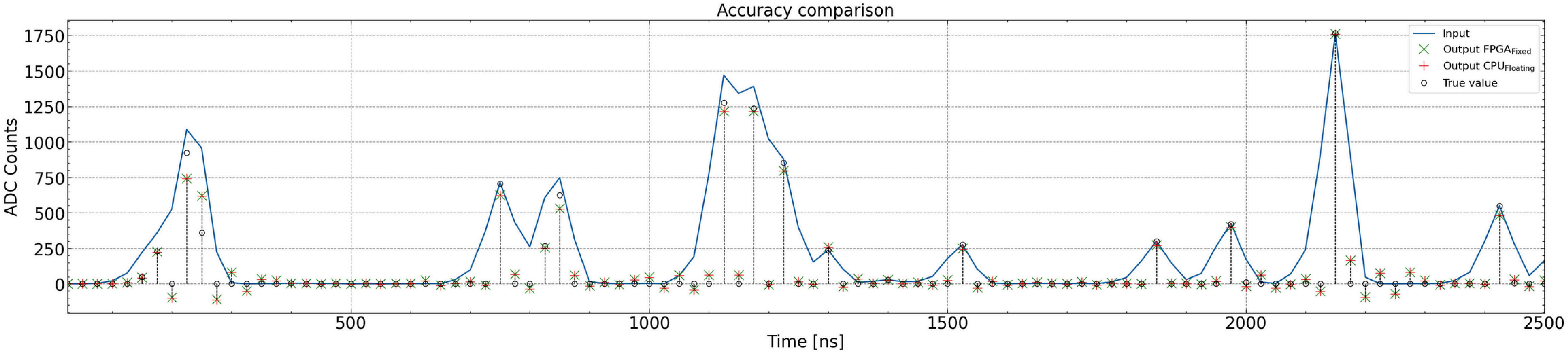


3 Results

Accuracy comparison

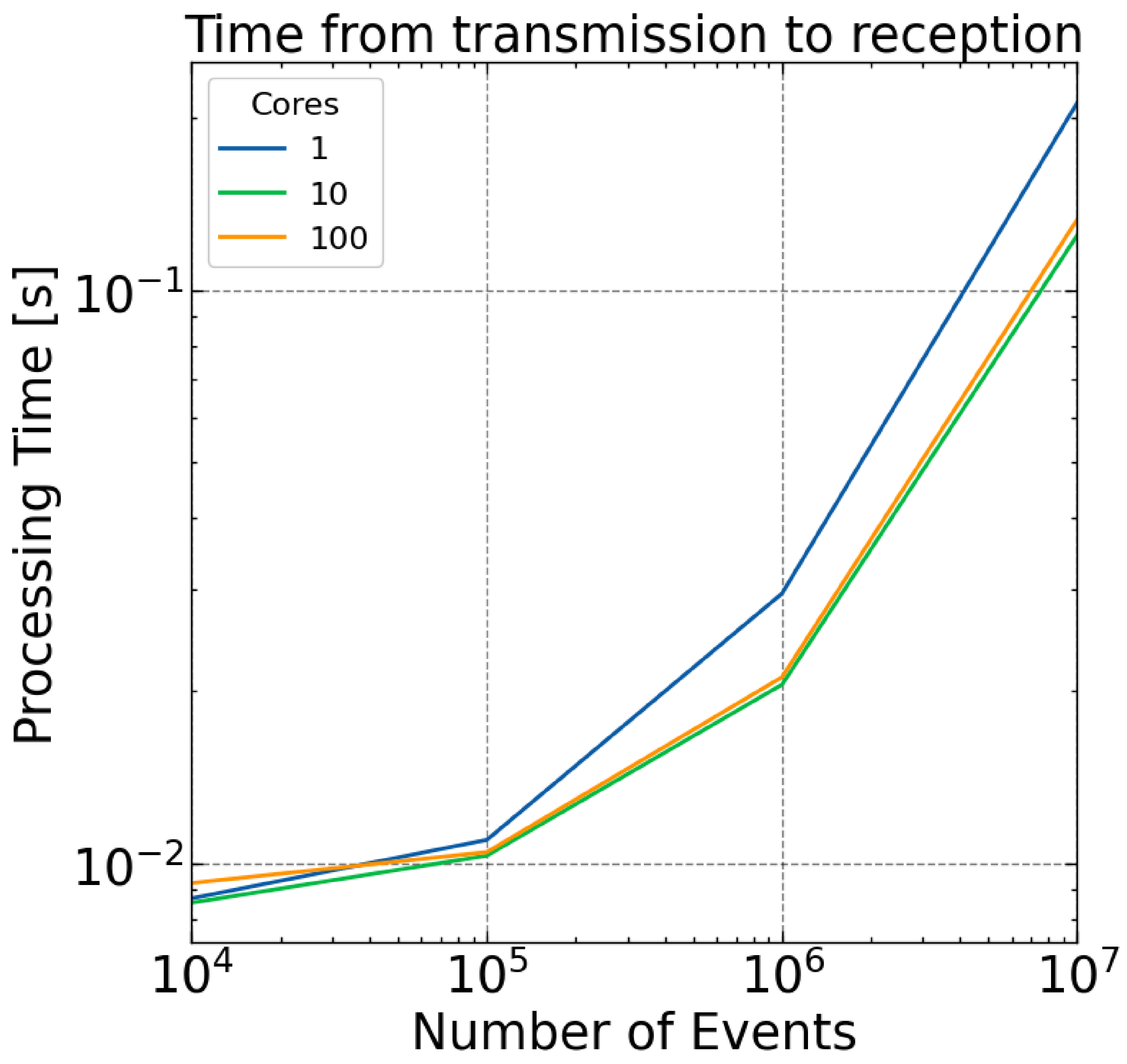
- CPU (**Floating point**) vs. FPGA (**Fixed point**)
- Maximum difference: **5 ADC Counts**
- FPGA amplitude > CPU amplitud due to the **fixed point implementation**





3 Results

Multi-core implementation



The **number of cores** is dependent of:

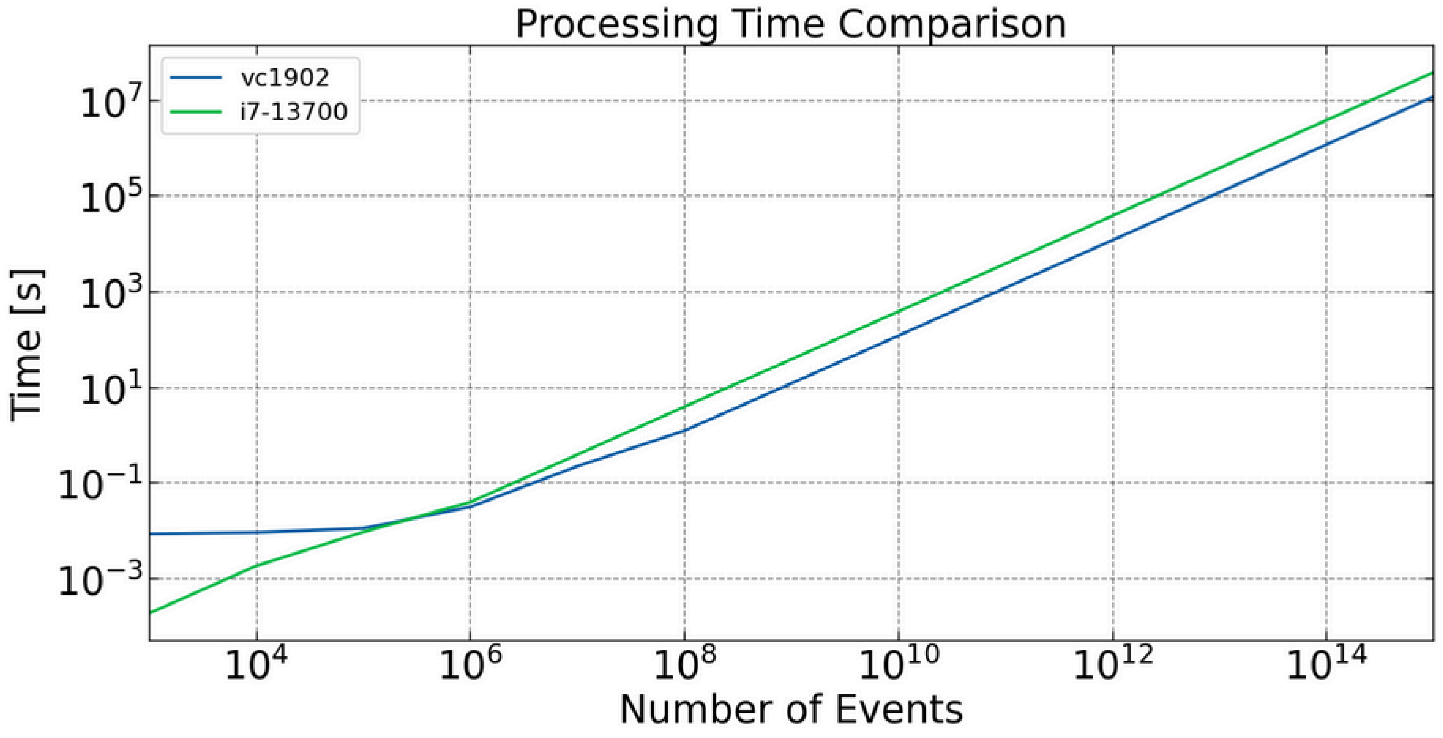
- NoC and DDR **bandwidth**
- NoC **OT** transactions
- PL **resources** used for each core (LUTs, FFs, BRAMs, DSPs, ...)

If **processing bandwidth** is greater than **NoC and DDR bandwidth**, **backpressure** is going to happen

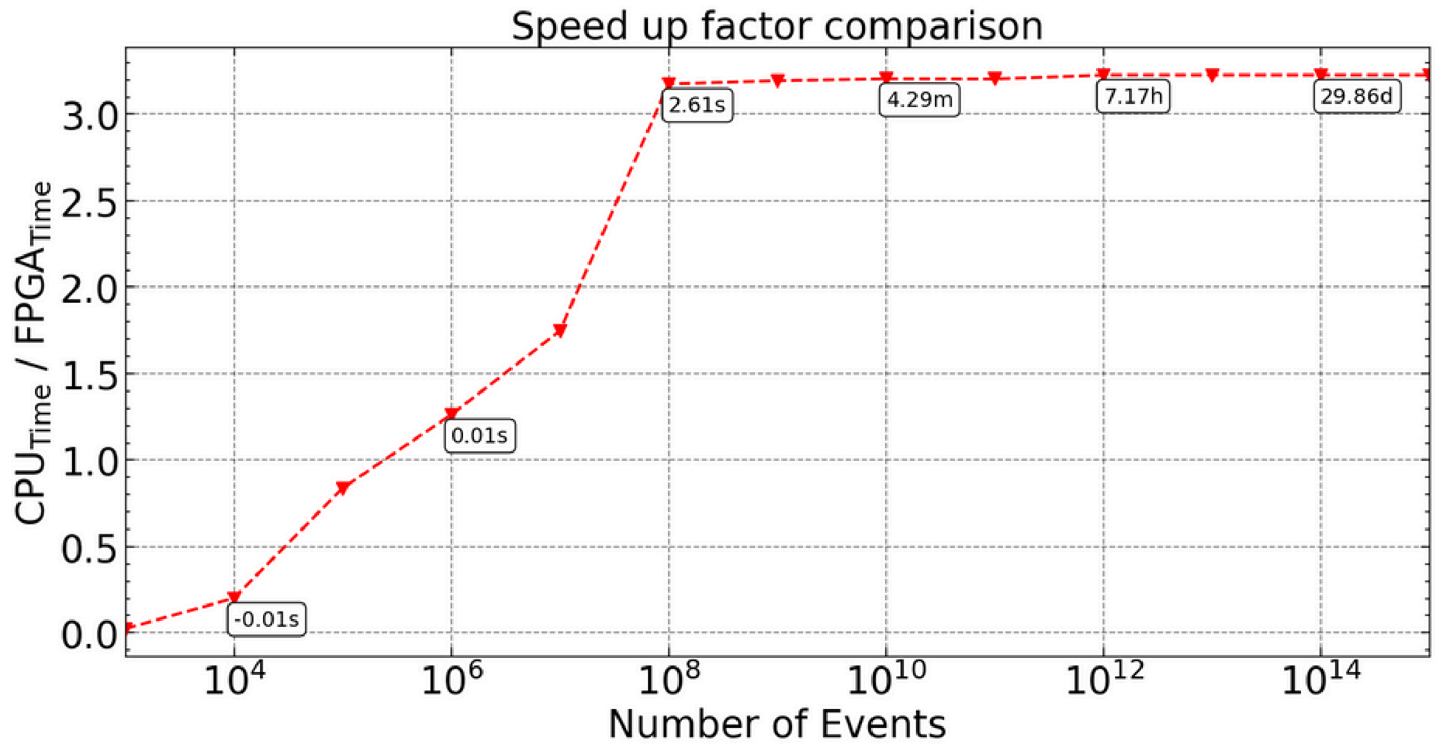
- Source is **ready to send** data, **but** the consumer is **not read to receive**, so **data** is going to be **lost**

3 Results

Timing comparison



- For **less than 10^6 events**, the **CPU** is **better than** the **FPGA** due to the fixed minimum time for transmission and setup
- For **more than 10^6 events**, the **FPGA** has a **better** performance **than** the **CPU**



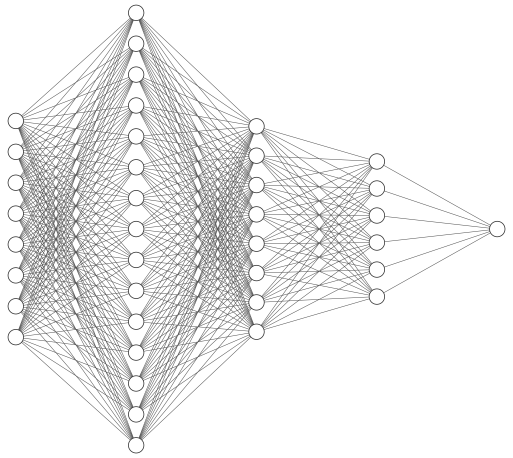
- The **speed up factor** remains stable (**x3.2**) for **more than 10^8 events**
- For **10^{12} events**, the **FPGA is 7.17 hours faster than the CPU**

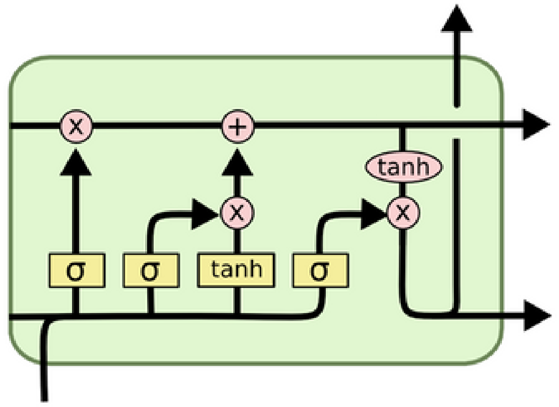
4 Conclusions

Summary and future work

Summary:

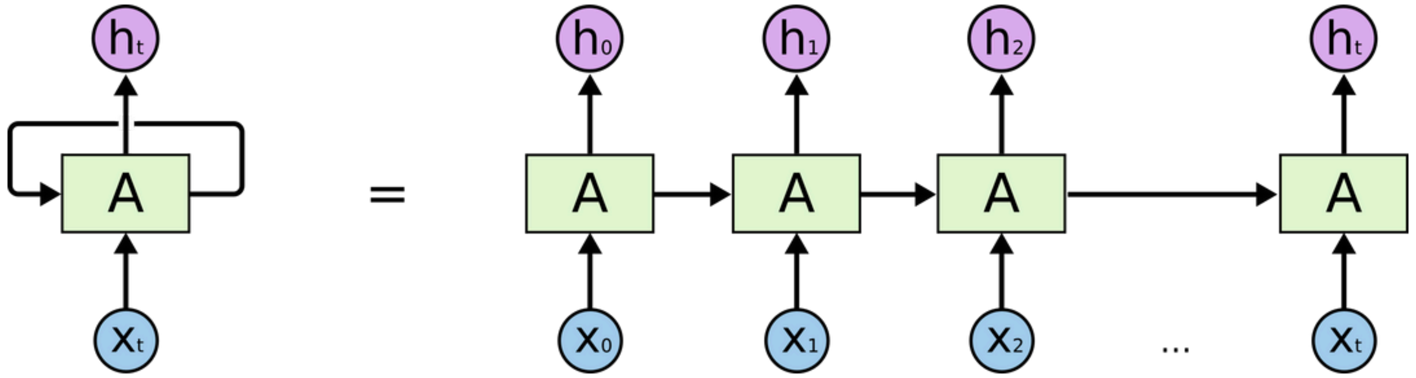
- FPGA implementation of deep learning algorithms improves efficiency over traditional CPU
- Algorithm optimization for real time application is a trade-off between latency/throughput and area usage





Future work:

- Evaluate more complex deep learning algorithms for real time implementation
- Power consumption will be measured and monitored
- AI Engines utilization and optimization
- Optimization in terms of latency and power consumption



Funding

“This work is supported by Ministerio de Ciencia, Innovación y Universidad con fondos Next Generation y del Plan de Recuperation, Transformacionales y Resiliencia (project - TED2021-130852B-100)”

GOBIERNO
DE ESPAÑA

MINISTERIO
DE CIENCIA, INNOVACIÓN
Y UNIVERSIDADES

Financiado por
la Unión Europea
NextGenerationEU

Plan de Recuperación,
Transformación y
Resiliencia

AGENCIA
ESTATAL DE
INVESTIGACIÓN

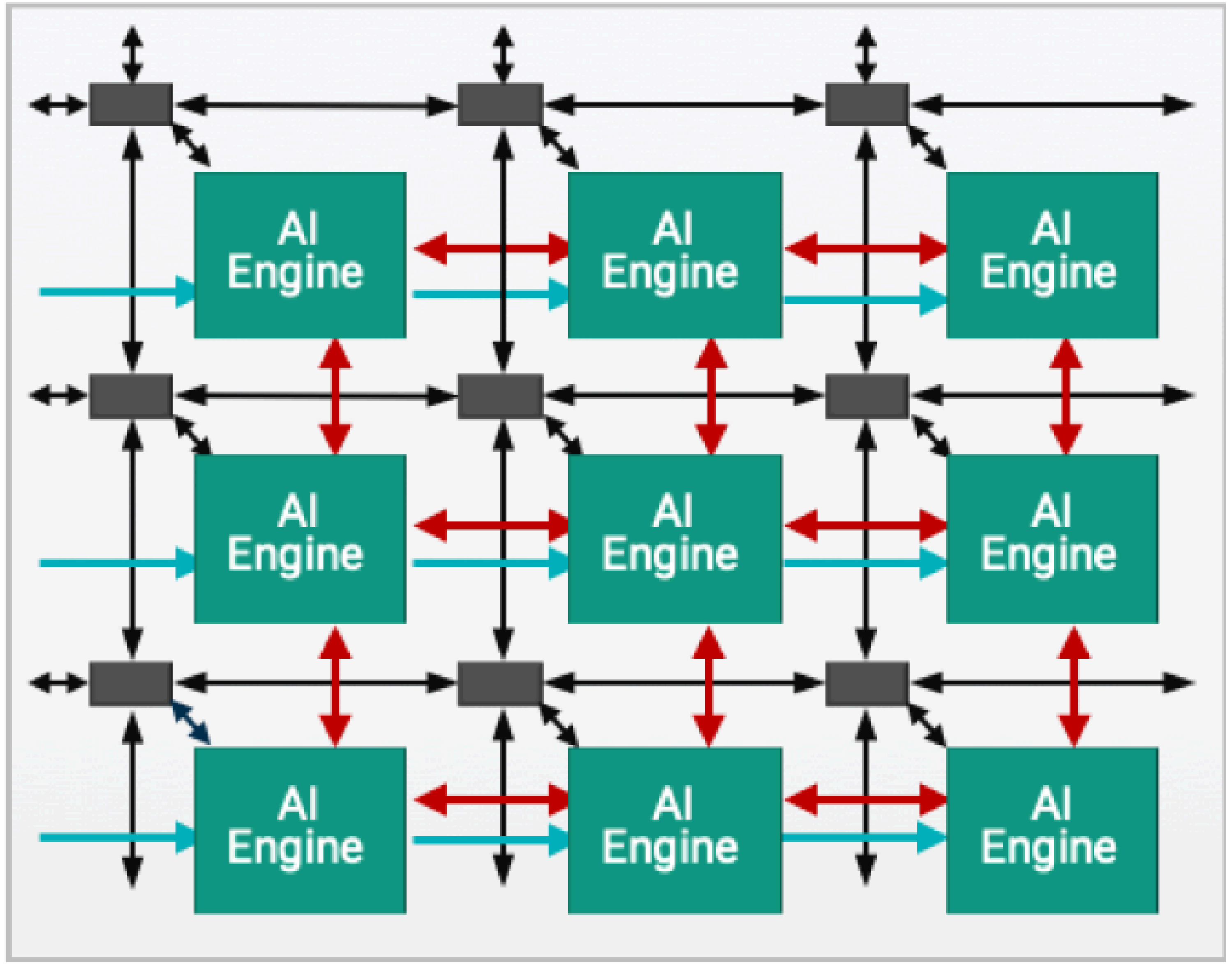
Versal ACAP Processing for HL-LHC Calorimeters Signal Reconstruction

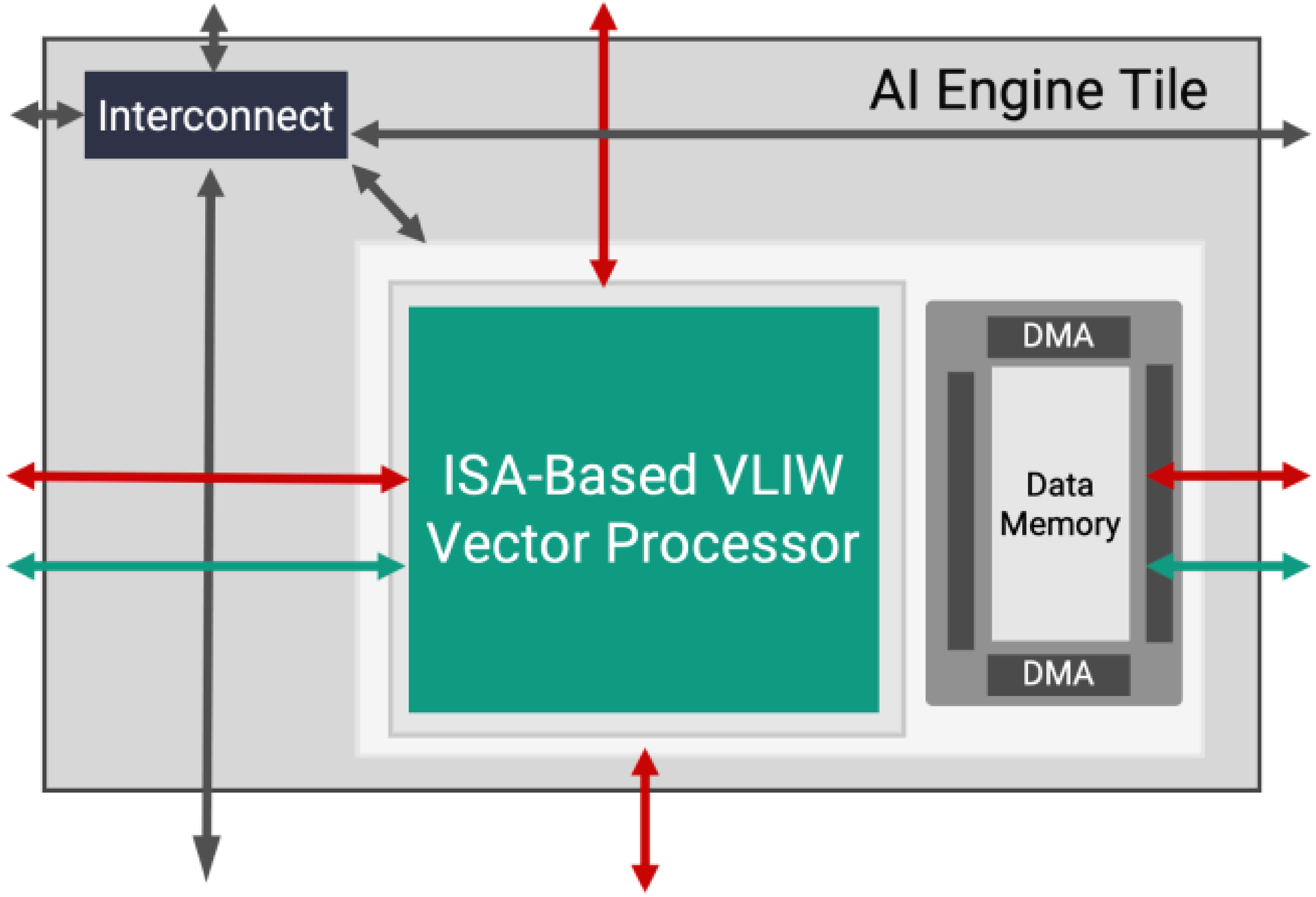
27th Conference on Computing in High Energy and Nuclear Physics (CHEP)
24th October, 2024

Francisco Hervas, *Alberto Valero, Luca Fiorini, Hector Gutierrez*

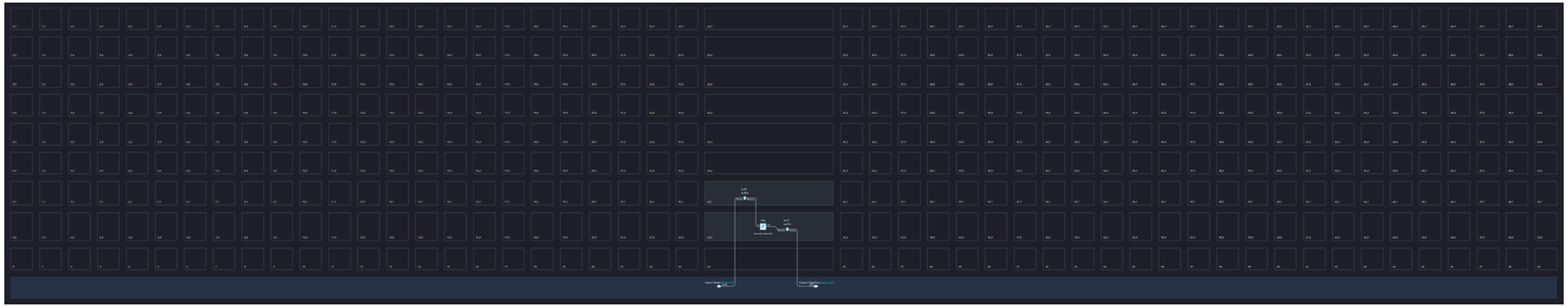
GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
AGENCIA ESTATAL DE INVESTIGACIÓN
IFIC INSTITUT DE FÍSICA CORPUSCULAR
CSIC
VNIVERSITAT DE VALÈNCIA
HL
CHEP 2024

AI Engines

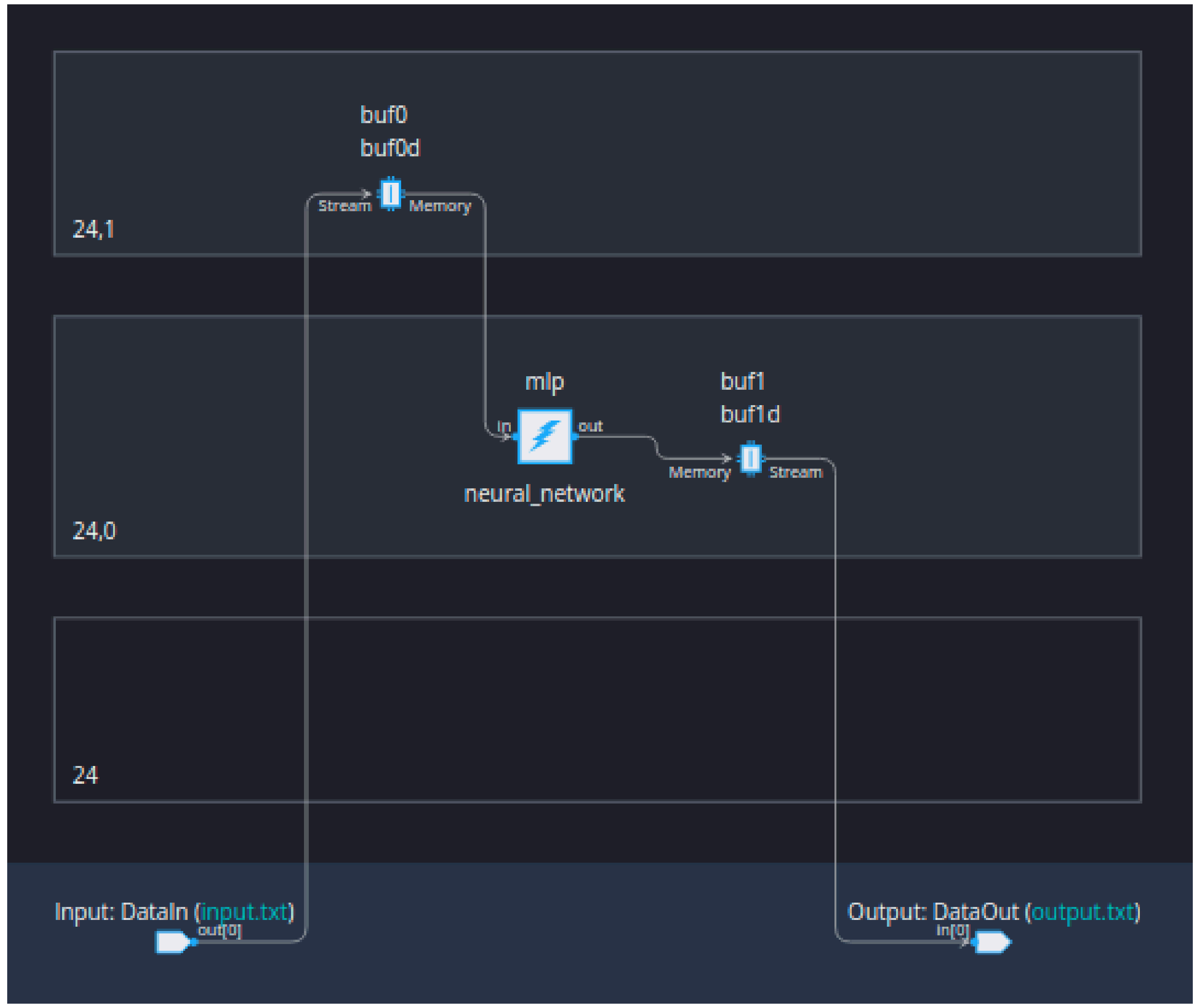

AI Engine
AI Engine
AI Engine
AI Engine
AI Engine
AI Engine
AI Engine
AI Engine
AI Engine


AI Engine Tile
Interconnect
ISA-Based VLIW Vector Processor
DMA
Data Memory
DMA

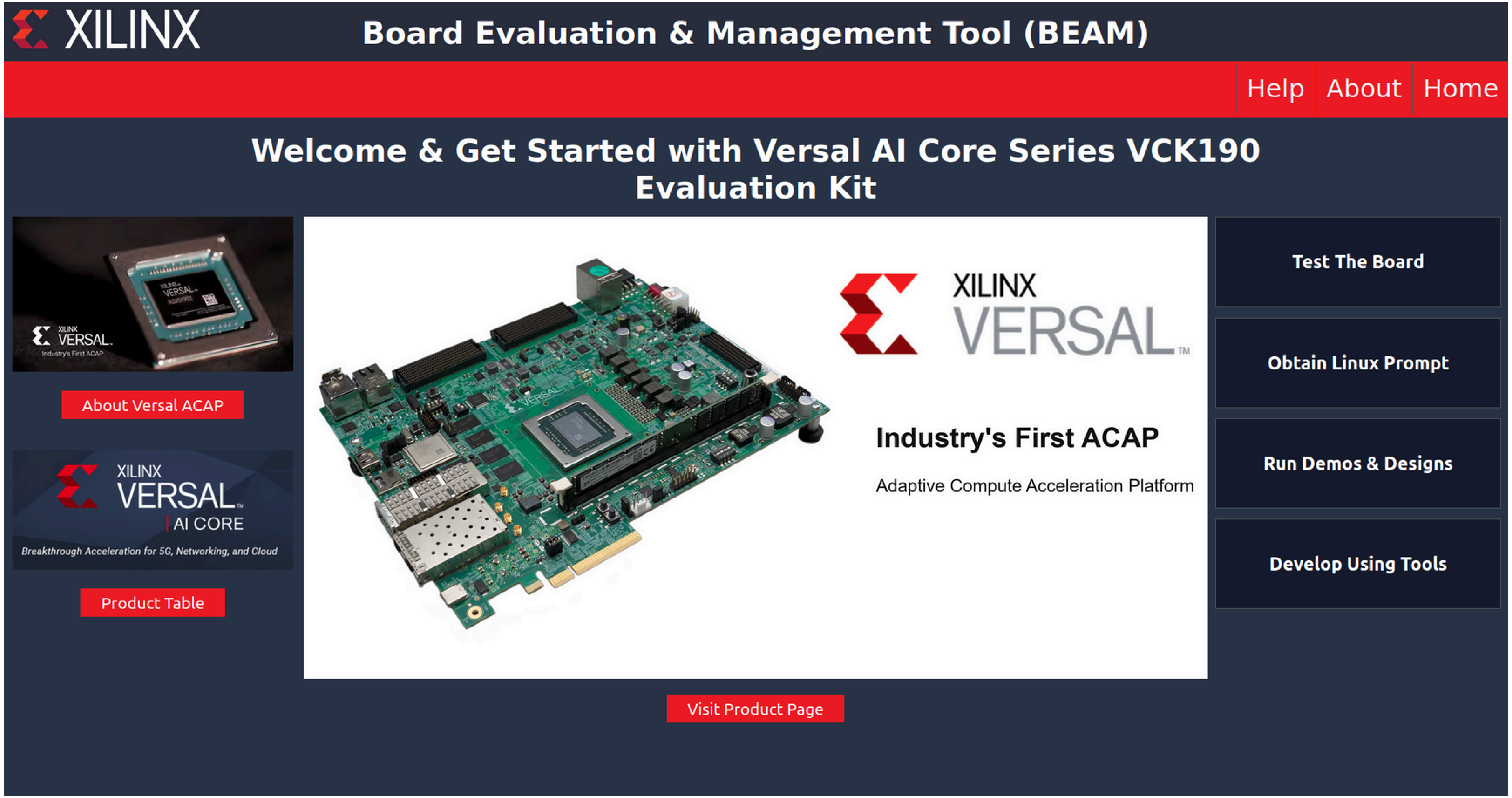
AI Engines



- 400 AI Engine Tiles
- Frequency
  - 1.25 GHz working
  - 312.5 MHz transport
- Latency
  - Input net: 12 cycles
  - Output net: 8 cycles



BEAM System Controller



XILINX Board Evaluation & Management Tool (BEAM)

Help About Home

Welcome & Get Started with Versal AI Core Series VCK190 Evaluation Kit

XILINX VERSAL
Industry's First ACAP

About Versal ACAP

XILINX VERSAL AI CORE
Breakthrough Acceleration for 5G, Networking, and Cloud

Product Table

XILINX VERSAL™
Industry's First ACAP
Adaptive Compute Acceleration Platform

Visit Product Page

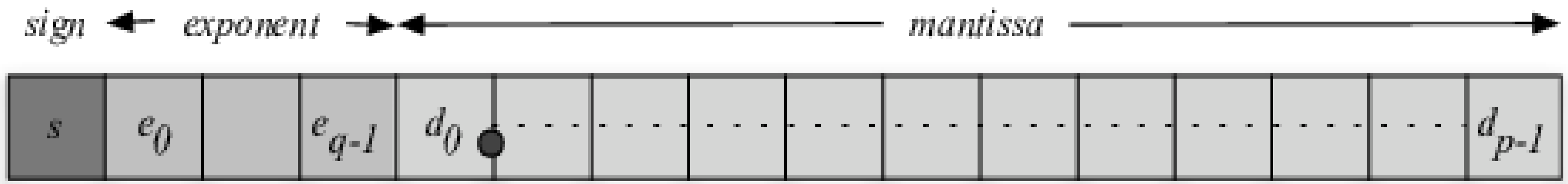
Test The Board

Obtain Linux Prompt

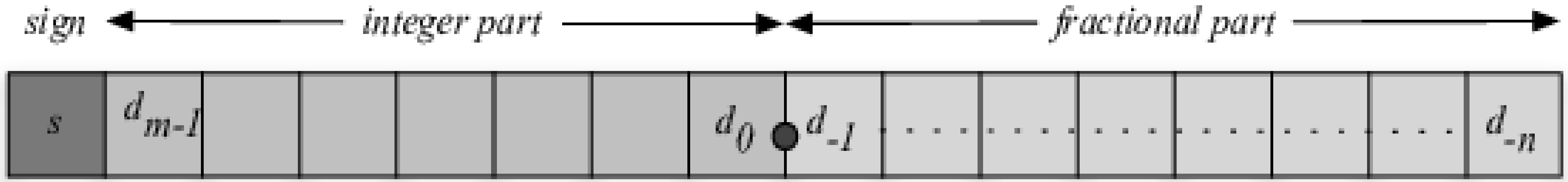
Run Demos & Designs

Develop Using Tools

Fixed point vs. Floating point

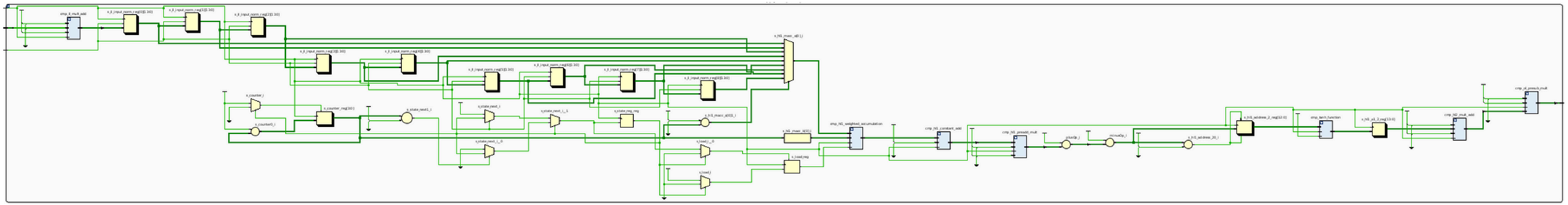


Floating-Point Format



Fixed-Point Format

RTL Design


cmp_il_mult_add
s_il_input_norm_reg[0][13:0]
s_il_input_norm_reg[1][13:0]
s_il_input_norm_reg[2][13:0]
s_il_input_norm_reg[3][13:0]
s_il_input_norm_reg[4][13:0]
s_il_input_norm_reg[5][13:0]
s_il_input_norm_reg[6][13:0]
s_il_input_norm_reg[7][13:0]
s_il_input_norm_reg[8][13:0]
s_h1_macc_a[0]_i
s_counter_i
s_counter0_i
s_counter_reg[3:0]
s_state_next1_i
s_state_next_i
s_state_next_i__1
s_state_next_i__0
s_state_reg_reg
s_h1_macc_a[0]1_i
s_h1_macc_a[0]_i
s_load_i__0
s_load_i
s_load_reg
cmp_h1_weighted_accumulation
cmp_h1_constant_add
cmp_h1_preadd_mult
plusOp_i
minusOp_i
s_h1_address_20_i
s_h1_address_2_reg[12:0]
cmp_tanh_function
s_h1_a1_2_reg[13:0]
cmp_h2_mult_add
cmp_ol_presub_mult