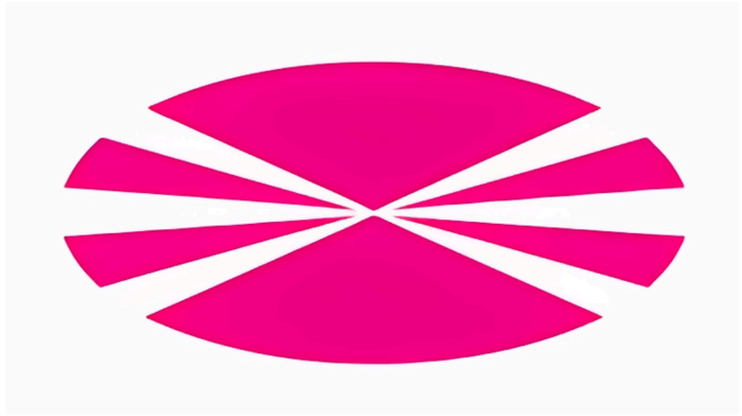
Versal ACAP processing for ATLAS-TileCal signal reconstruction



2nd Computing Challenges Workshop (COMCHA), A Coruña
October 2nd - 4th, 2024

Francisco Hervas, Luca Fiorini, Alberto Valero,
Héctor Gutiérrez, Francesco Curcio

HIGH-LOW
TED2021-130852B-100

INDEX

1 Introduction - LHC TileCal Read-out

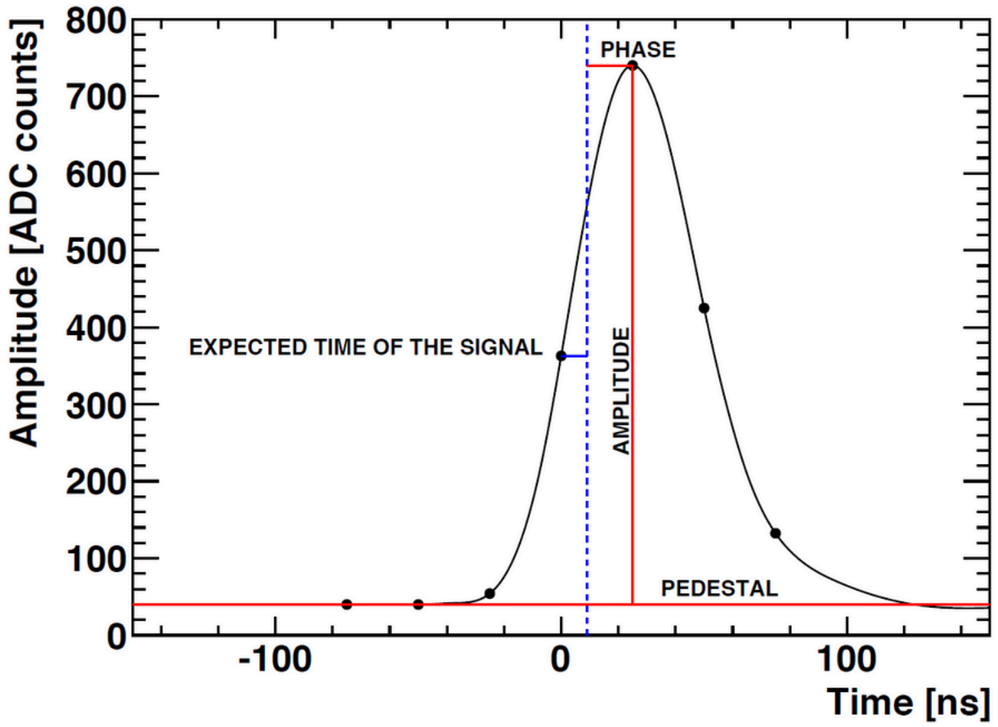
- In the **LHC**, Bunch Crossings (**BC**) happen at **40 MHz (25 ns)**
- The processing happens after the **Level-1 Trigger, at 100 kHz (10 us)**
- Signals are processed online using the **Optimal Filtering (OF) algorithm**
  - The processing is made using **Digital Signal Processors (DSPs)**
  - Therefore,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

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**

DOI: 10.1109/RTC.2007.4382840



1 Introduction - HL-LHC TileCal 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**)

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]

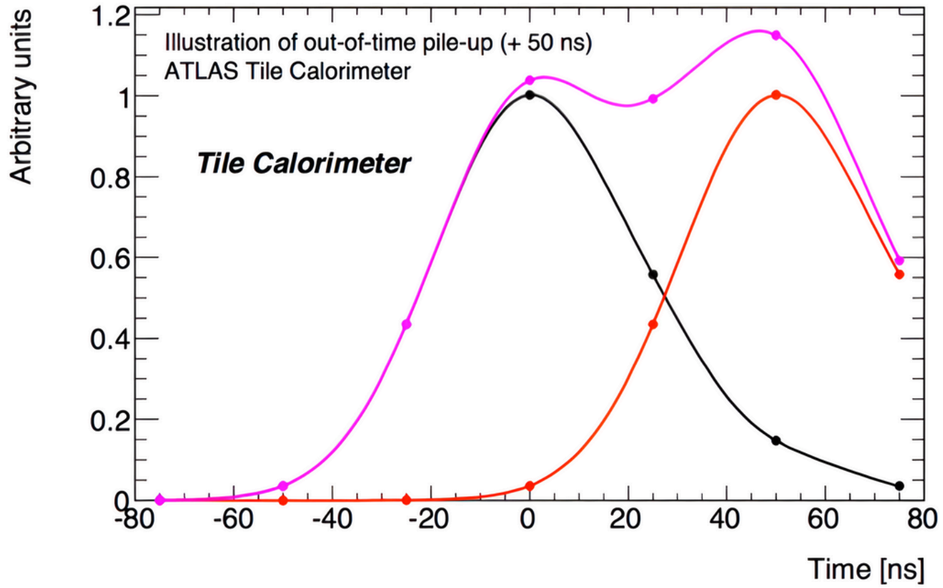
Published 2 September 2019 • © 2019 IOP Publishing Ltd and Sissa Medialab

Journal of Instrumentation, Volume 14, September 2019

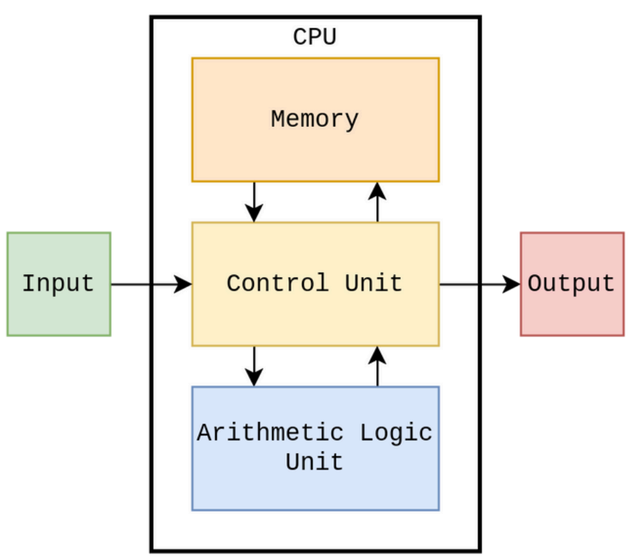
Citation J.L. Ortiz Arciniega *et al* 2019 *JINST* **14** P09002

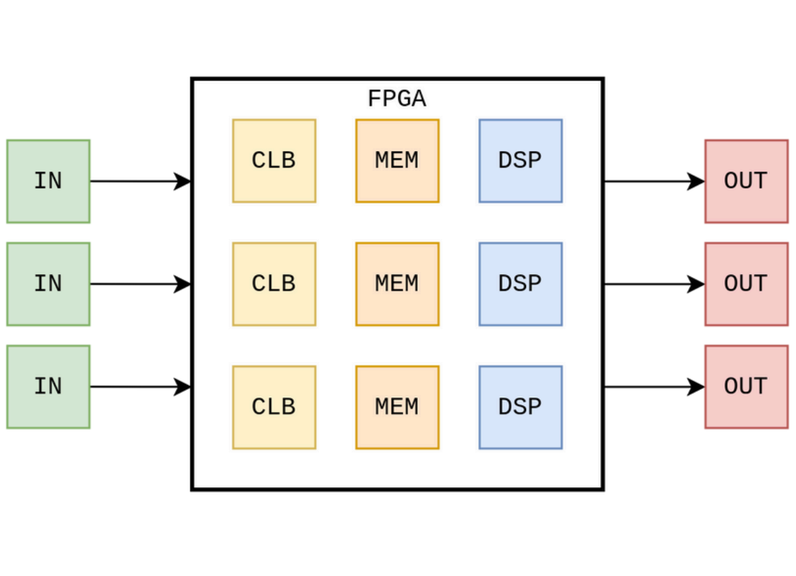
DOI 10.1088/1748-0221/14/09/P09002

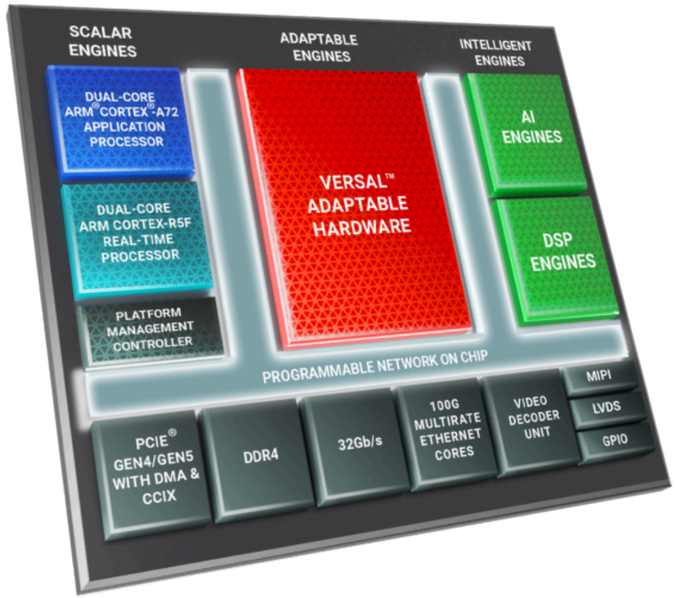
DOI: 10.1088/1748-0221/14/09/P09002



1 Introduction - Device Comparison







CPU (DSP):

- Sequential
- Fixed circuits
- Programming language

FPGA:

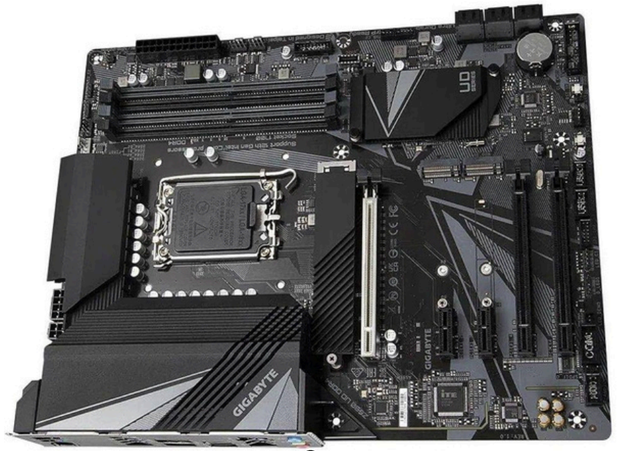
- Parallel and concurrent
- Configurable circuits
- Hardware description language (HDL)

SoC:

- Sequential + Parallel
- Fixed + Configurable
- Programming + HDL

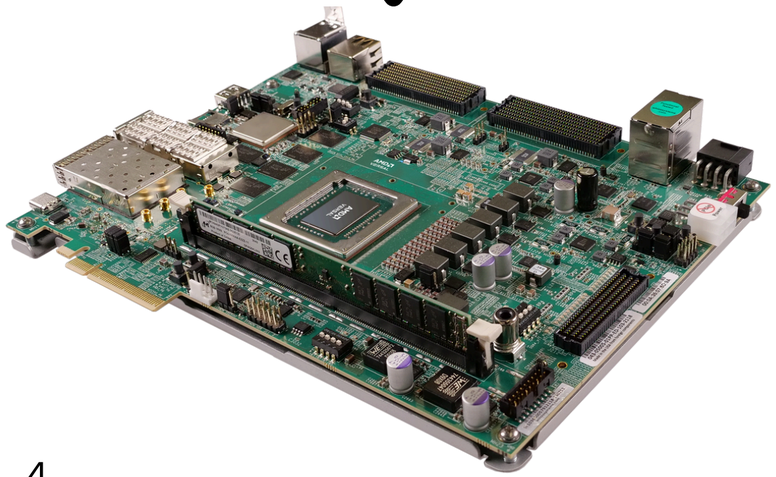
1 Introduction - Setup

- Setup
  - Computer
    - Mother board: Gigabyte Technology Co., Ltd Z690 UD DDR4
    - CPU: 13th Gen Intel Core i7-13700 x 24
    - GPU: NVIDIA GeForce RTX 3050
    - Memory: 64 GB
    - Disk: 2 TB
  - Evaluation board
    - VCK190, VC1902
    - DDR4 (8 GB) and LPDD4 (8 GB)
    - PCIe Gen4 x8
    - JTAG and QSPI
    - MicroSD
    - SYSMON
    - UART, CAN, SFP28 and QSFP28
  - System on Chip
    - x400 AI Engines, x1968 DSP slices, x1968 Logic cells, x899840 LUTs
    - APU A72, RPU R5F
    - x4 Memory controllers
    - x770 I/O pins



+

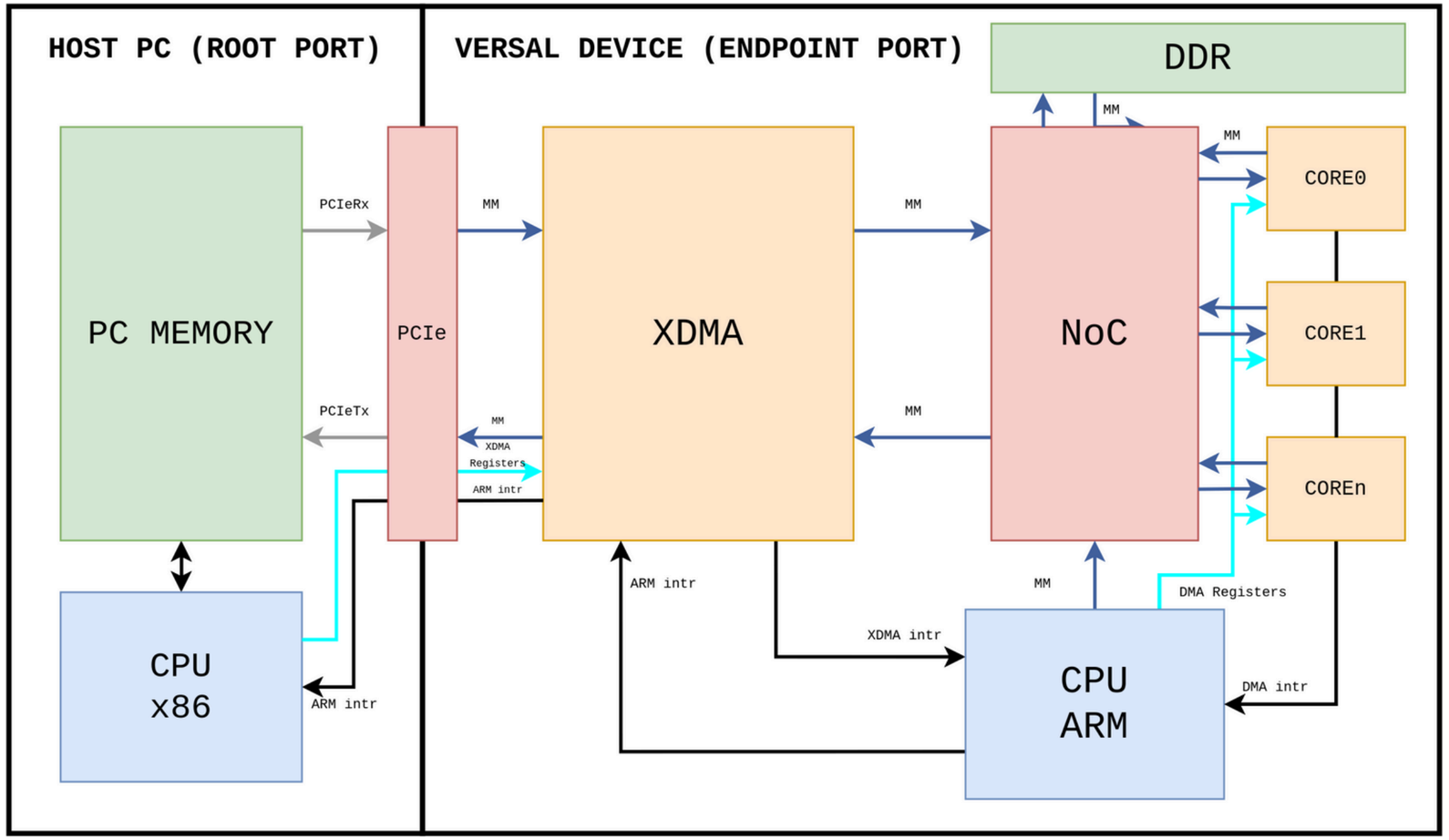
PCIe Gen4 x8



+

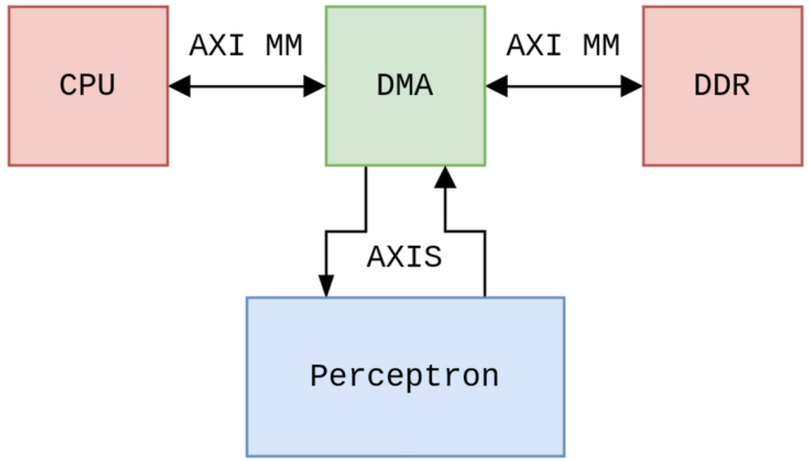
2 Methods - Complete System Implementation

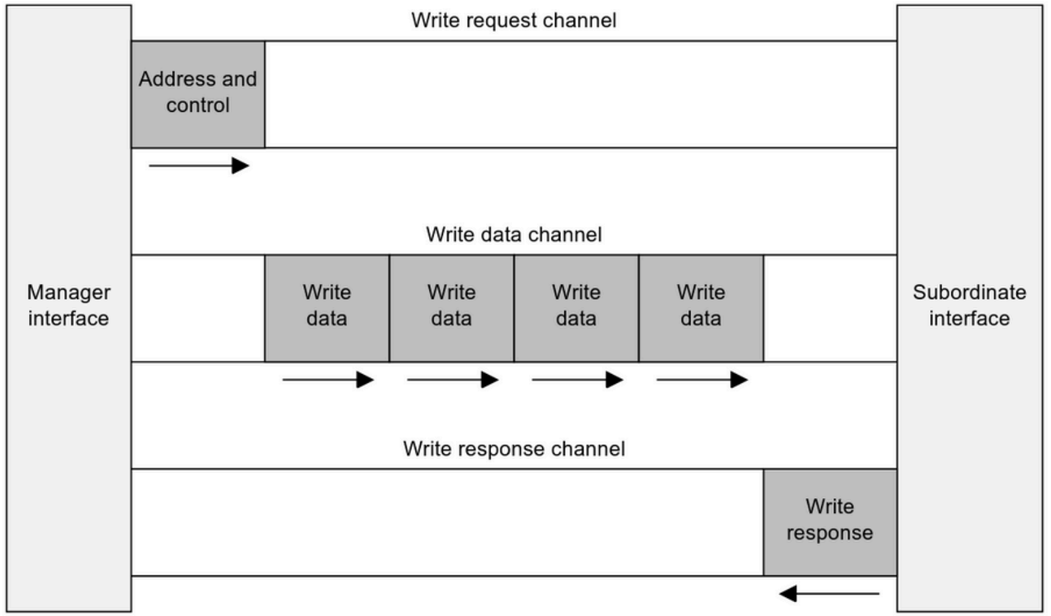
- Driver implementation in host CPU for communication with XDMA
- Driver implementation in device CPU for managing internal DMAs
- NoC configuation for internal communication
- Interrupt system development
- Multiple cores executing algorithms

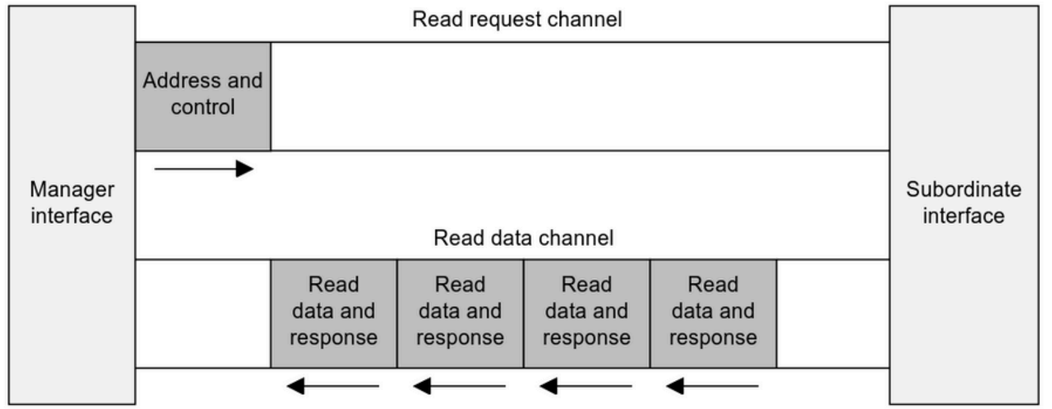


2 Methods - AXI4 Memory Map and Lite

- The AXI4 Memory Map is transactions-based and defines five independent channels
- Multiple Outstanding Transactions (OT)
- Most common protocol in FPGA + CPU based devices

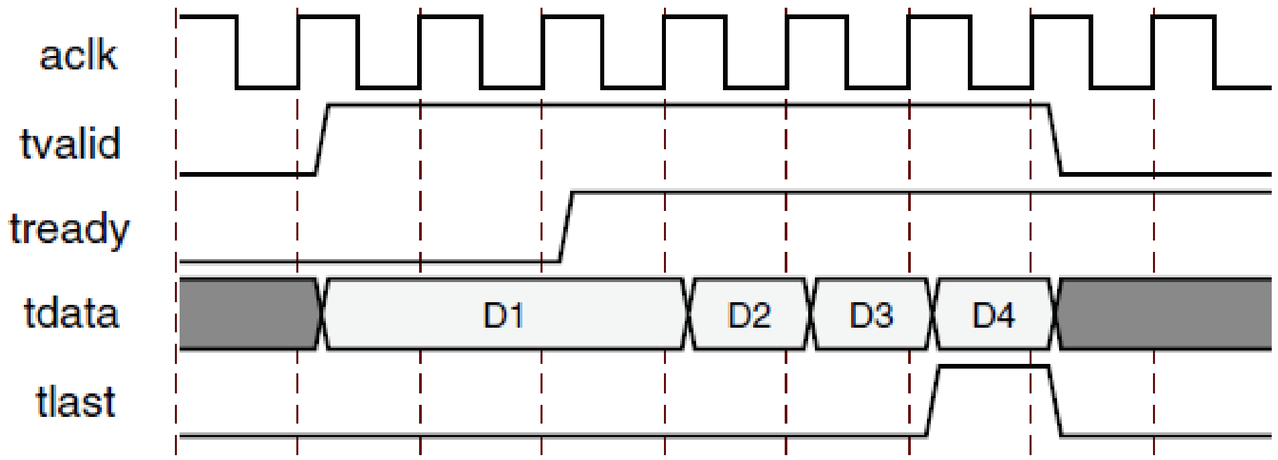


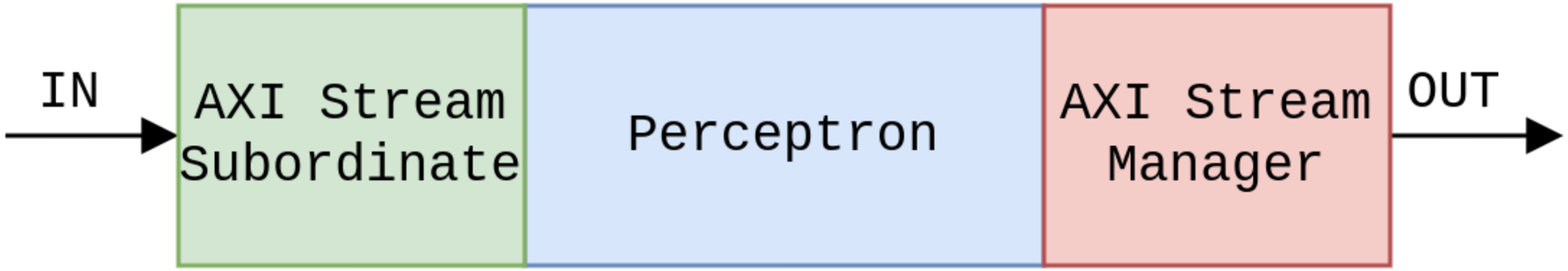




2 Methods - AXI4 Stream Interface

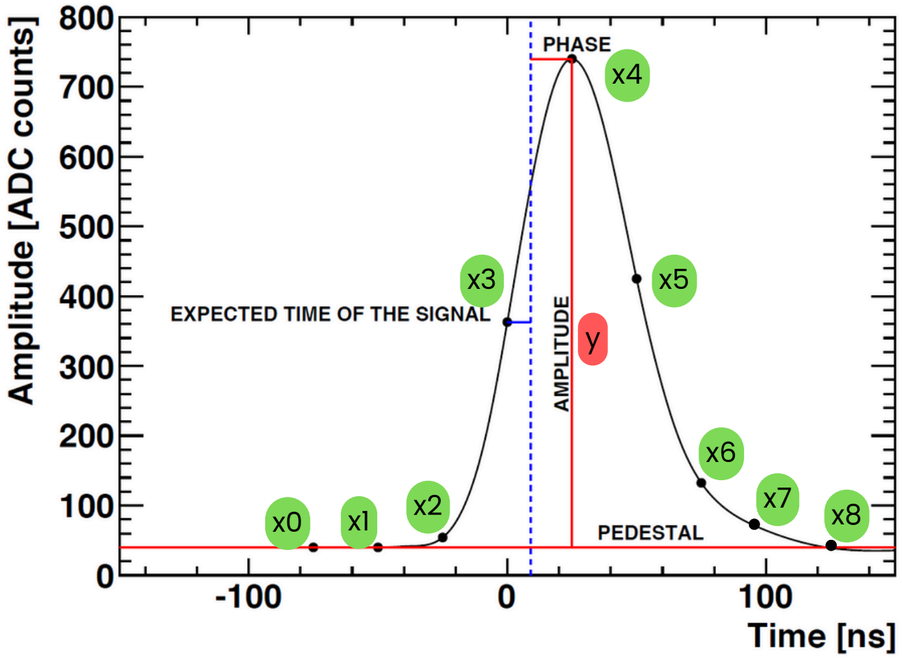
- The AXI4 Stream interface is a point to point link where the transmitter is known as a master or manager, and the receiver a slave or subordinate
- Basic handshake
- There are 4 important signals

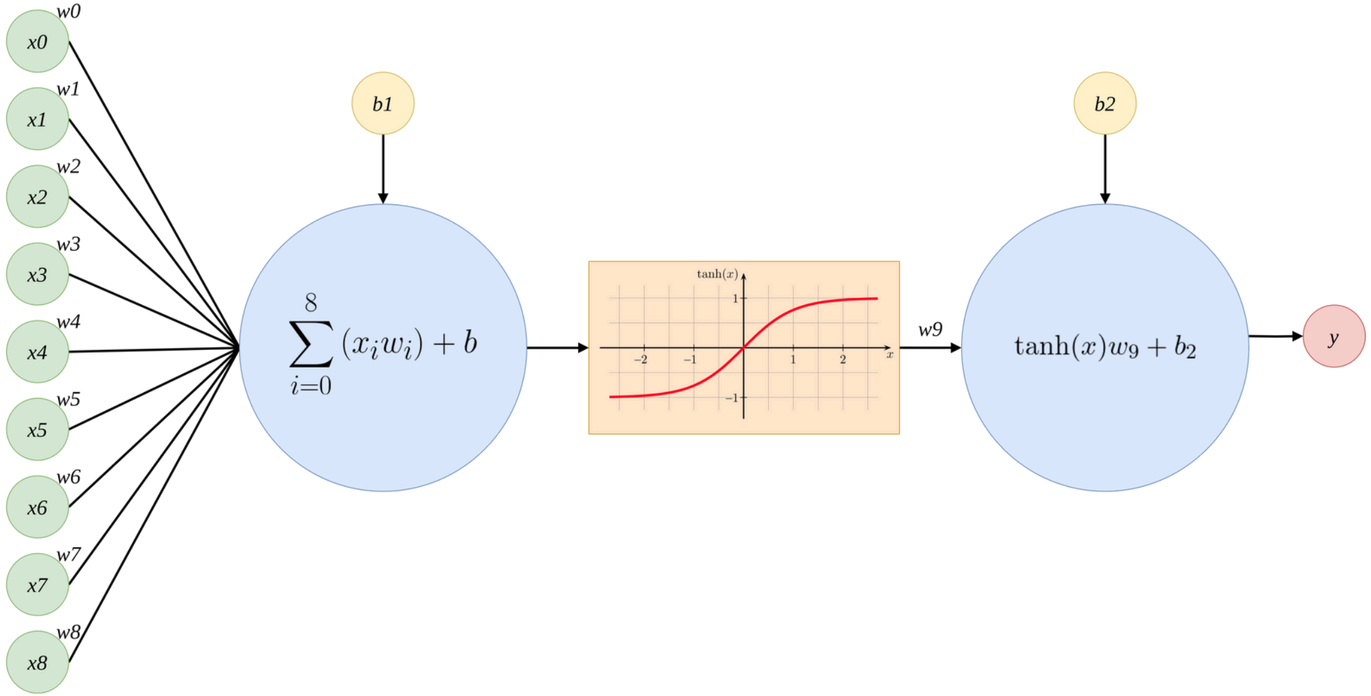




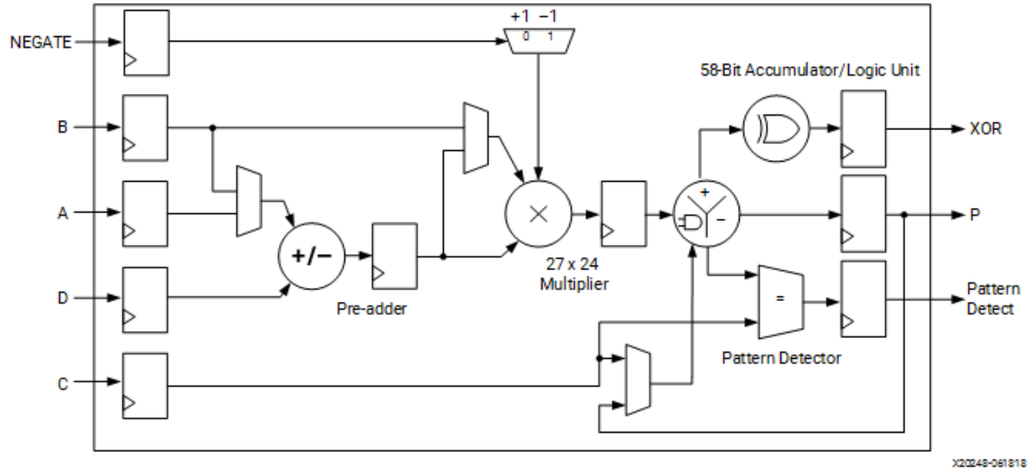
2 Methods - Modified Perceptron

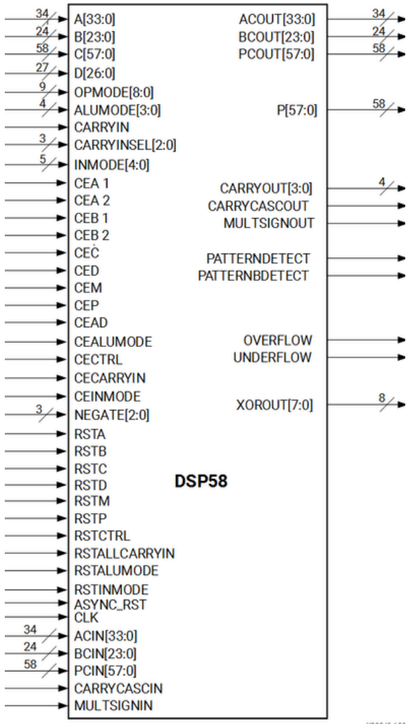
- Read-out window of 9 BC
  - Sliding 1 BC for each new window
- Target the true amplitude of the central BC in the window
- Hidden layer and the output layer
- Hyperbolic Tangent





2 Methods - DSP58 Multiply-Accumulate





vs.

```
entity macc is
    generic(
        ...
    );
    port(
        clk : in std_logic;
        a : in sfixed(g_MACC_A_POS downto g_MACC_A_NEG);
        b : in sfixed(g_MACC_B_POS downto g_MACC_B_NEG);
        c : in sfixed(g_MACC_C_POS downto g_MACC_C_NEG);
        out_reg : in std_logic;
        ce : in std_logic;
        opmode : in std_logic;
        p : out sfixed(g_MACC_P_POS downto g_MACC_P_NEG)
    );
    attribute dsp_folding : string;
    attribute dsp_folding of macc : entity is "yes";
end macc;
architecture rtl of macc is
  ...
begin
    process(clk)
    begin
        if (rising_edge(clk)) then
            if (ce = '1') then
                s_a <= a;
                s_b <= b;
                s_c <= c;
                s_mult <= s_a * s_b;
                s_p <= resize(s_mult + s_old_add, s_p'high, s_p'low);
            end if;
        end if;
    end process;
 end rtl;
```

DSP58 Highlights:

- 27-bit x 24-bit multiplier
- 58-bit adder/accumulator
- 116-bit wide XOR function
- 4 registers for full pipeline

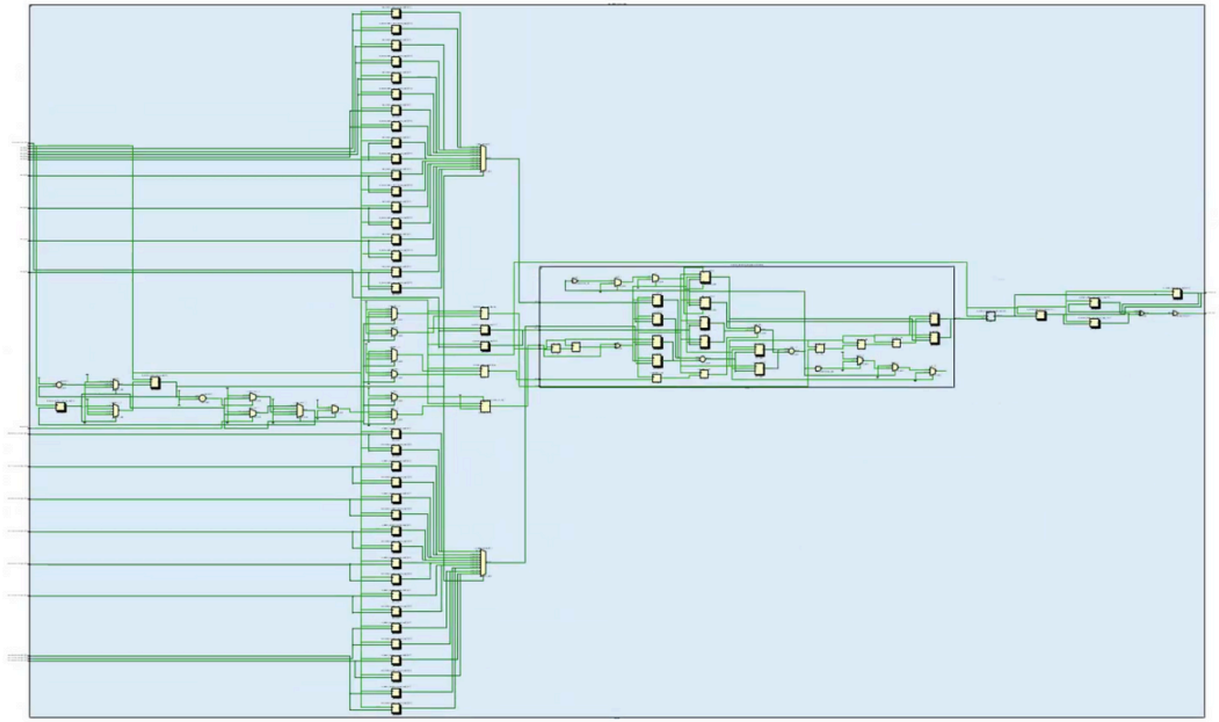
VC1902:

- 1968 DSP58 engines
- 1070 MHZ max frequency

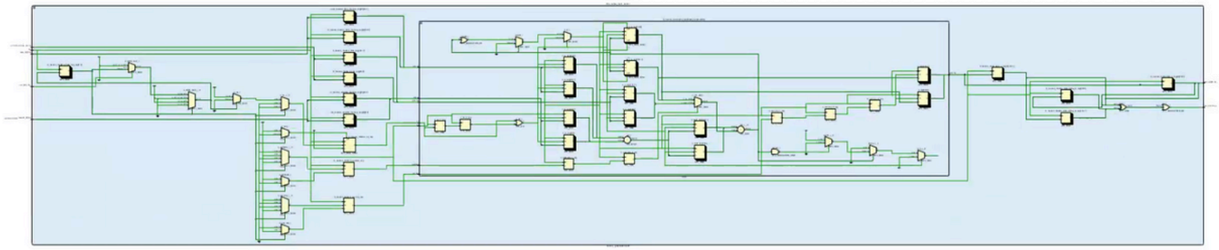
- The DSP58 can be instantiated with a primitive, or coded with RTL.
  - With instantiation more complex structures can be implemented
  - With RTL more flexibility between devices is achieved

2 Methods - RTL Level Hidden Layer and Output Layer

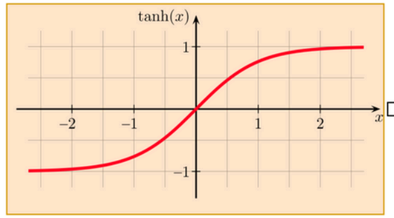
Hidden Layer



Output Layer



- RTL Design of the two layers of the Neural Nework
- VHDL-2008 standard
- Fixed point arithmetic
- Synthesis and implementation in Vivado
- Activation function tanh(x) quantized over 5000 values



QUANTIZED
5000 values
[-0.7, 0.8]

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]

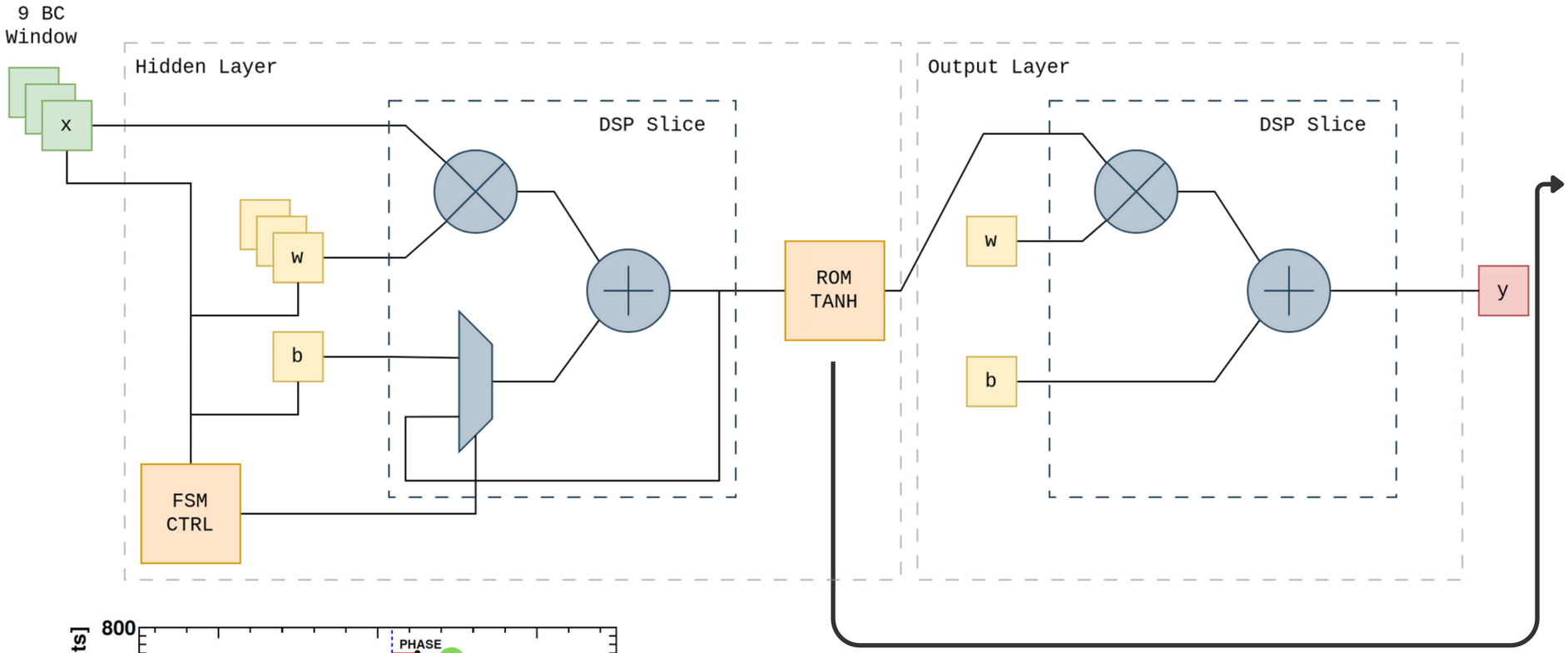
Published 2 September 2019 • © 2019 IOP Publishing Ltd and Sissa Medialab

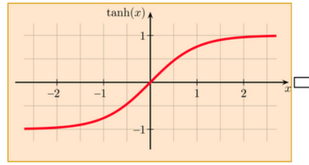
Journal of Instrumentation, Volume 14, September 2019

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

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

2 Methods - RTL Level Modified Perceptron



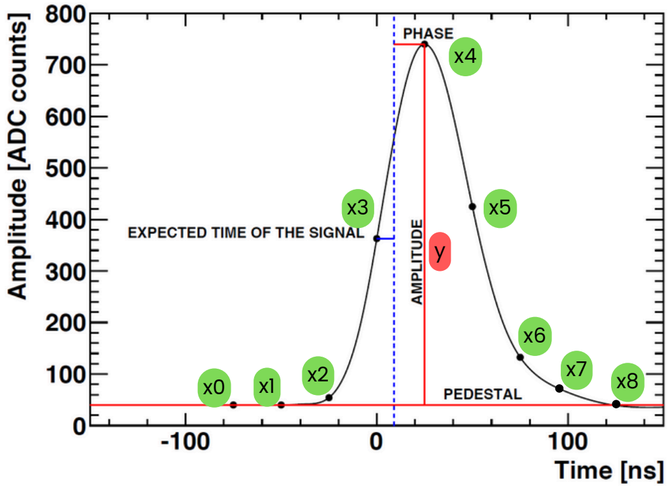


QUANTIZED
5000 values
[-0.7, 0.8]

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

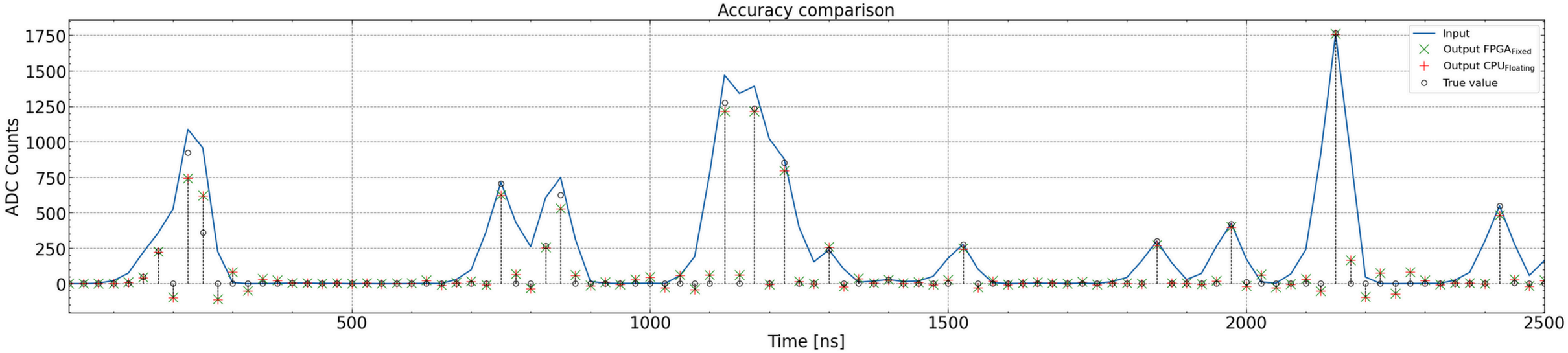


- Input: 9 BC (Samples) shift register sliding window
- Output: True amplitude of the central BC of the input window
- DSP Engine based implementation
- FSM to control the accumulator counter
- TANH implemented as a ROM of 5000 quantized values

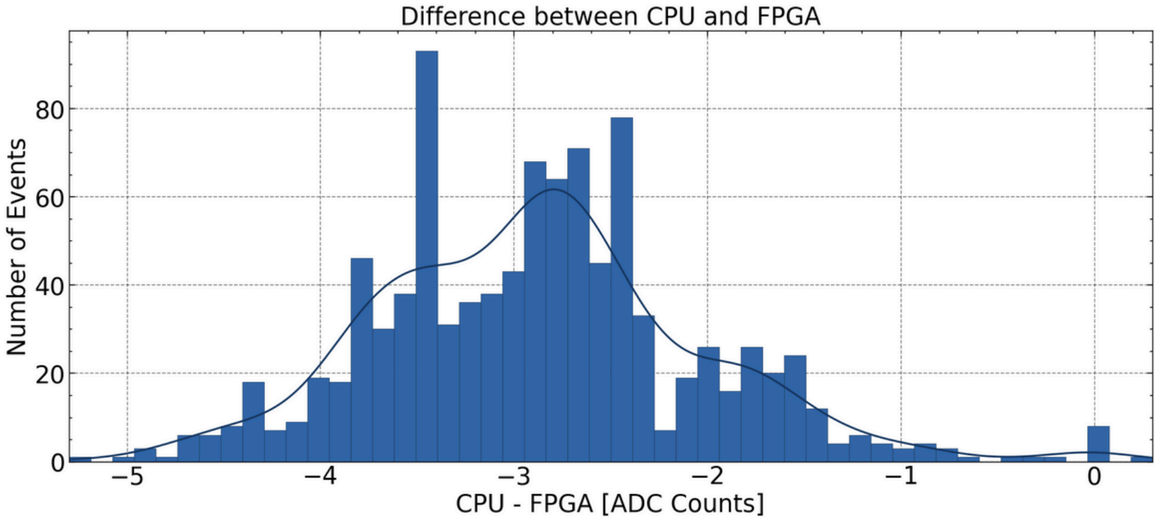
2 Methods - Modified Perceptron Resource Utilization

Resource	Utilization	Available	Utilization %
LUT	2099	899840	0.23
FF	531	1799680	0.03
DSP	6	1968	0.30
IO	228	692	32.95
BUFG	1	980	0.10

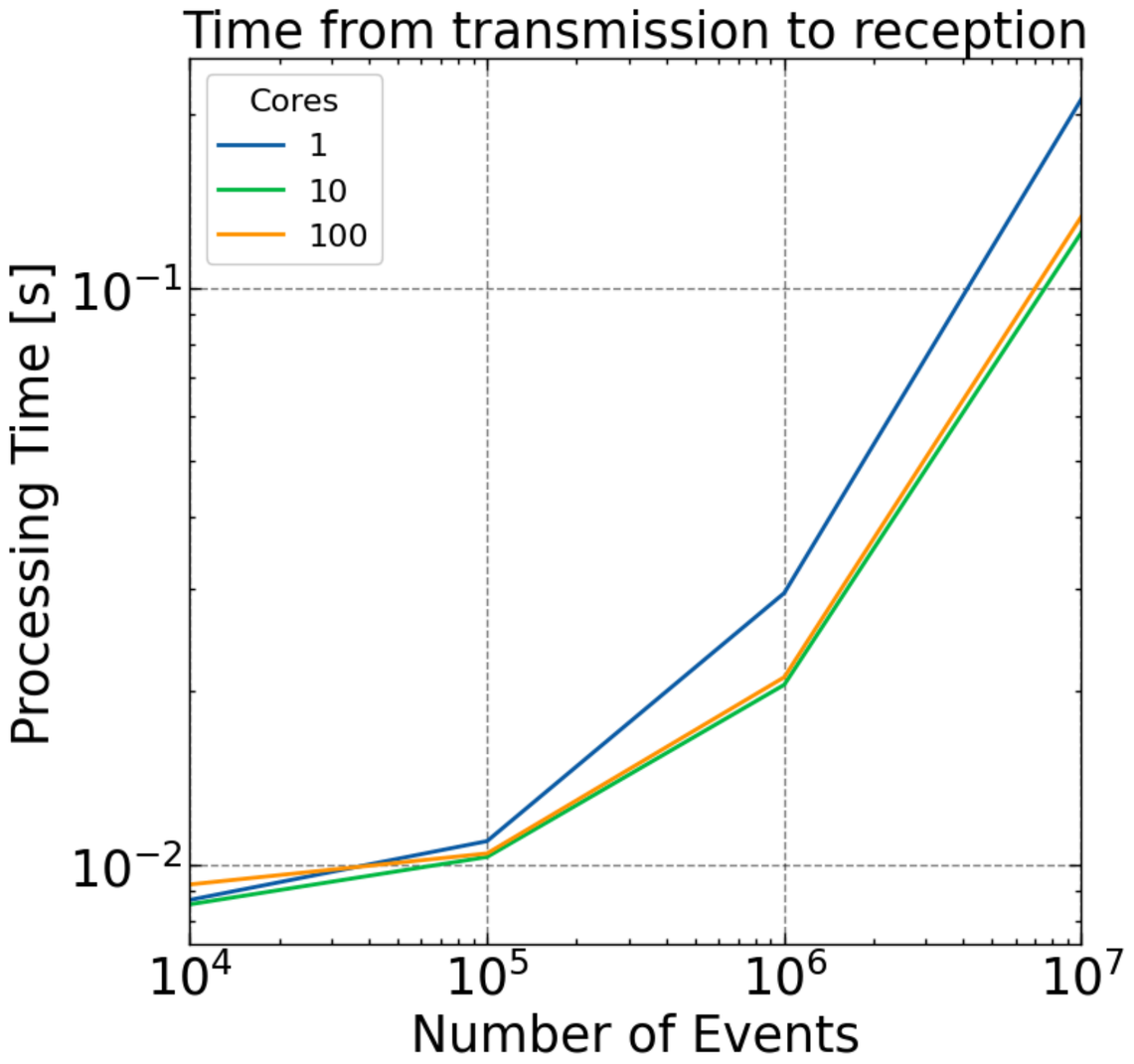
3 Results - Accuracy Comparison CPU vs. FPGA



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



3 Results - Time Comparison Between Different Cores

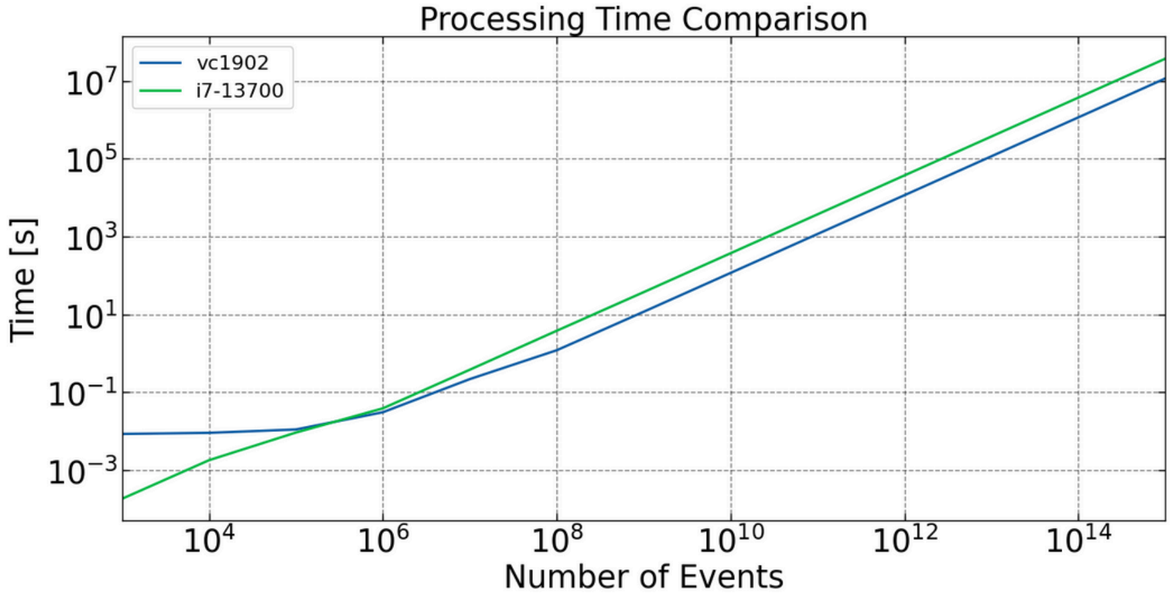


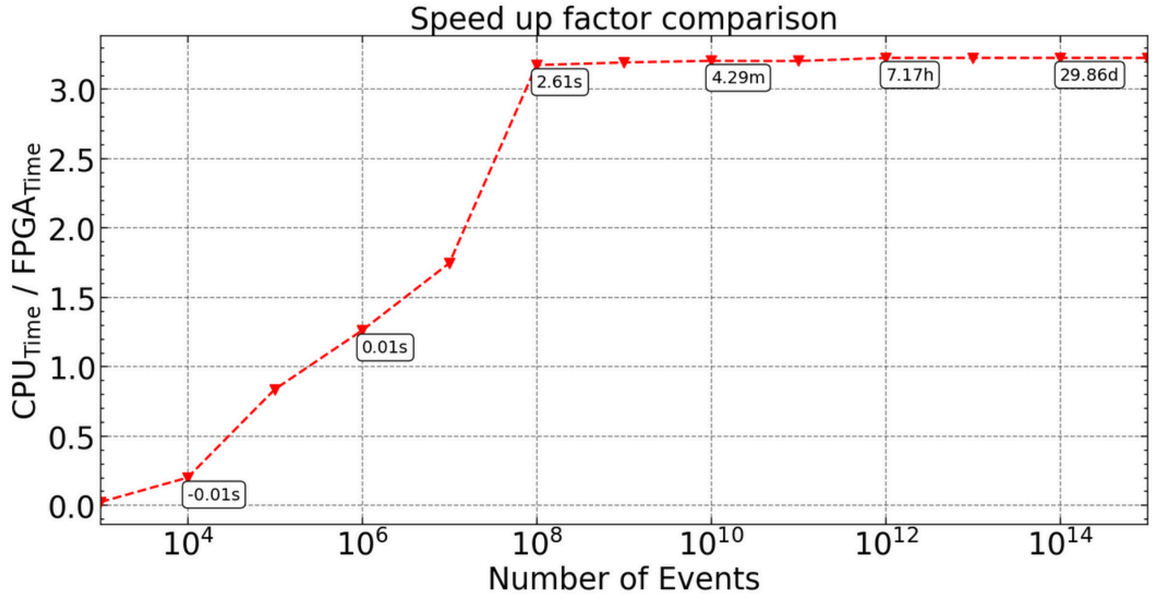
The number of cores is dependent of:

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

```
if (Processing_BW > (NoC and DDR_BW)) then
  Backpressure
end if;
```

3 Results - Time Comparison CPU vs. FPGA





- 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**

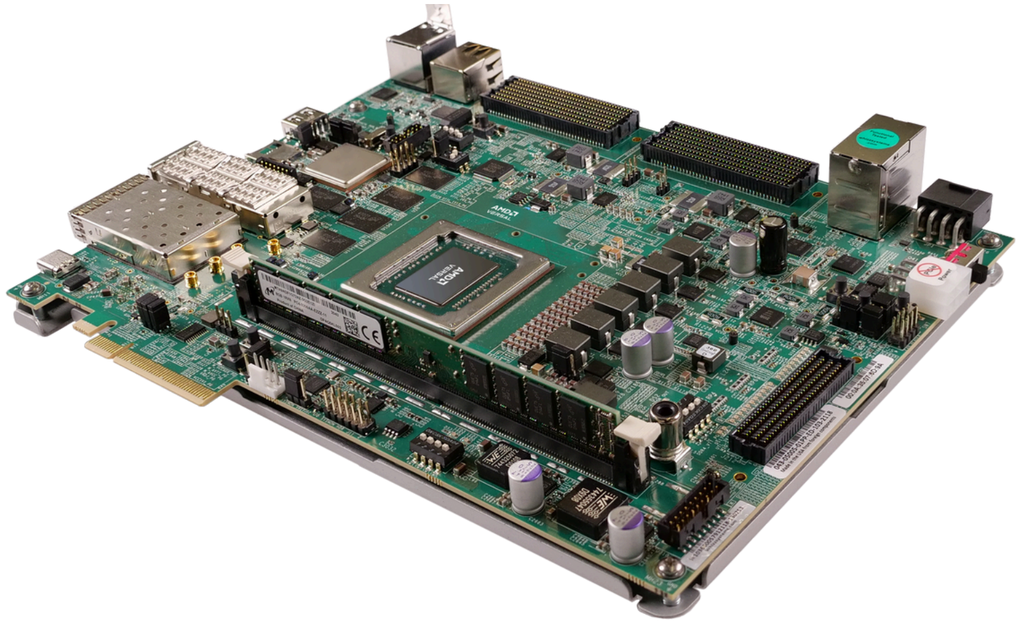
4 Summary

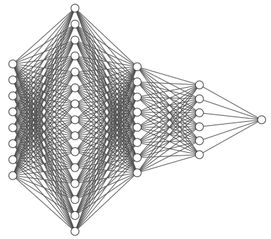
Summary:

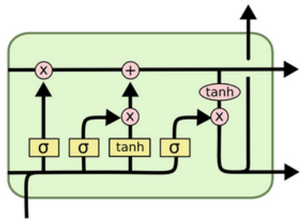
- FPGA implementation of deep learning algorithms improves the efficiency over traditional CPU.

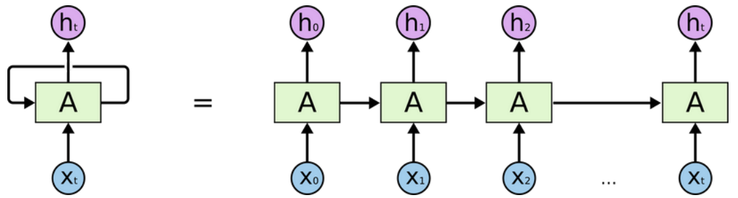
Future work:

- More complex deep learning algorithms will be implemented.
- Power consumption will be measured and monitored.
- AI Engines utilization and optimization.
- Optimization in terms of latency and power consumption





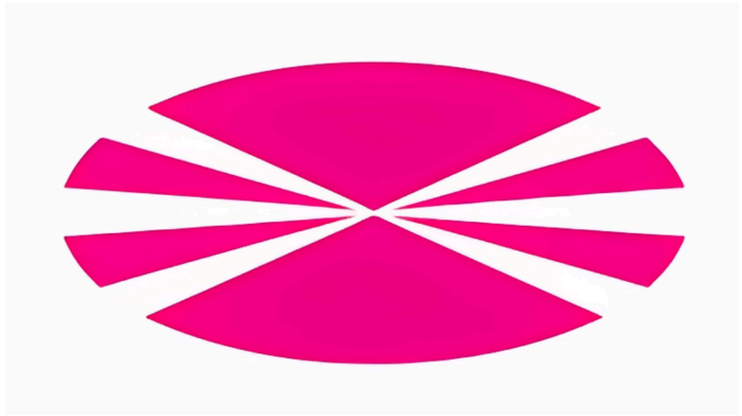




Funding

“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)”

Versal ACAP processing for ATLAS-TileCal signal reconstruction



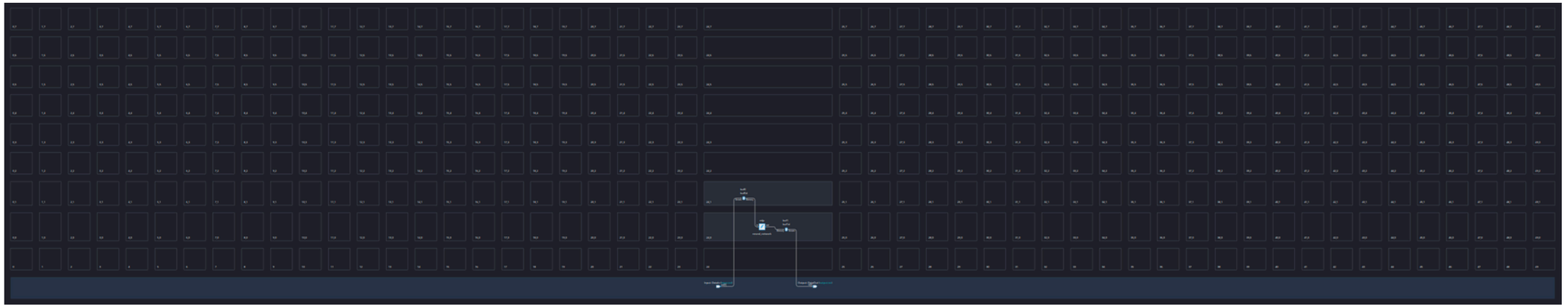
2nd Computing Challenges Workshop (COMCHA), A Coruña
October 2nd - 4th, 2024

Francisco Hervas, Luca Fiorini, Alberto Valero,
Héctor Gutiérrez, Francesco Curcio

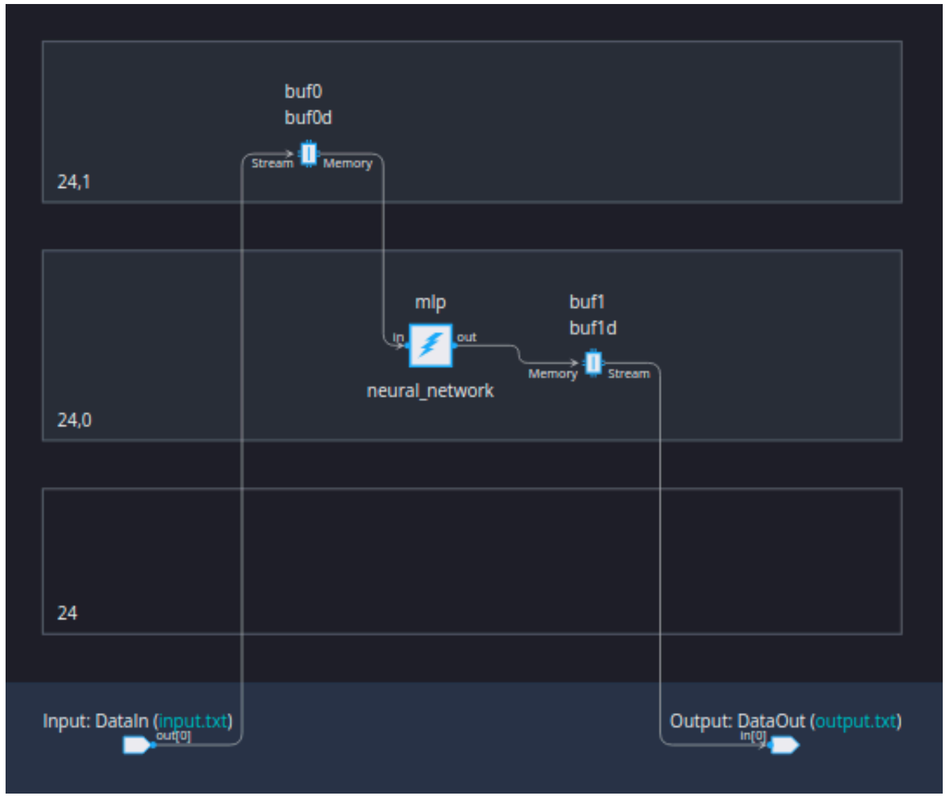
HIGH-LOW
TED2021-130852B-100

Backup

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

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

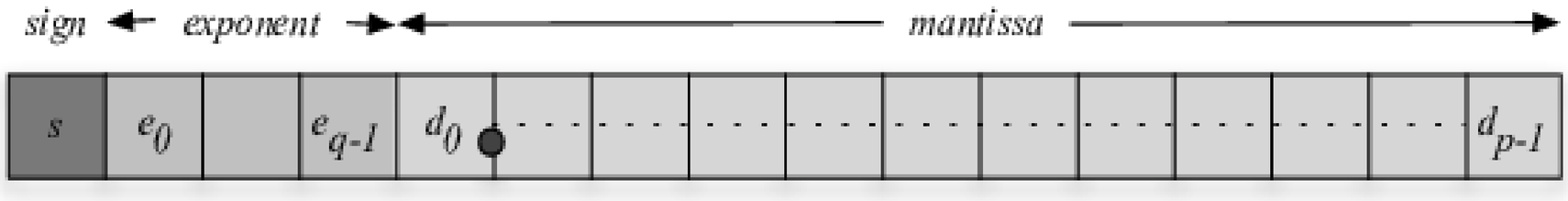
Test The Board

Obtain Linux Prompt

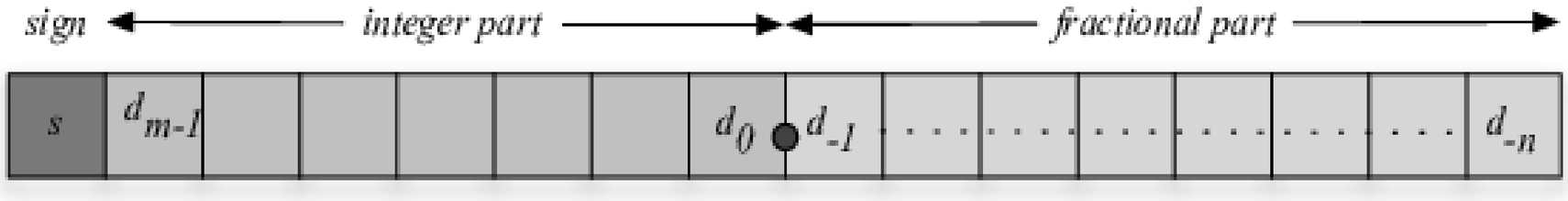
Run Demos & Designs

Develop Using Tools

Fixed point vs. Floating point



Floating-Point Format



Fixed-Point Format

Complete RTL