Using high-level synthesis languages to accelerate DAQ and processing applications for AMD-XILINX MPSoC-based AMCs

A. Piñas, C. González, M.Ruiz, A. Carpeño, E. Barrera, J. Nieto

a.pinas@upm.es

Outline

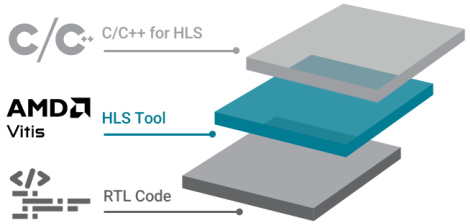
1. Introduction
   - Goal and motivation
   - AMD-Xilinx acceleration methodology
2. Proposed system
   - Hardware Platform
   - Software Platform
3. Use case – Digital Pulse Shape Analysis
4. Summary

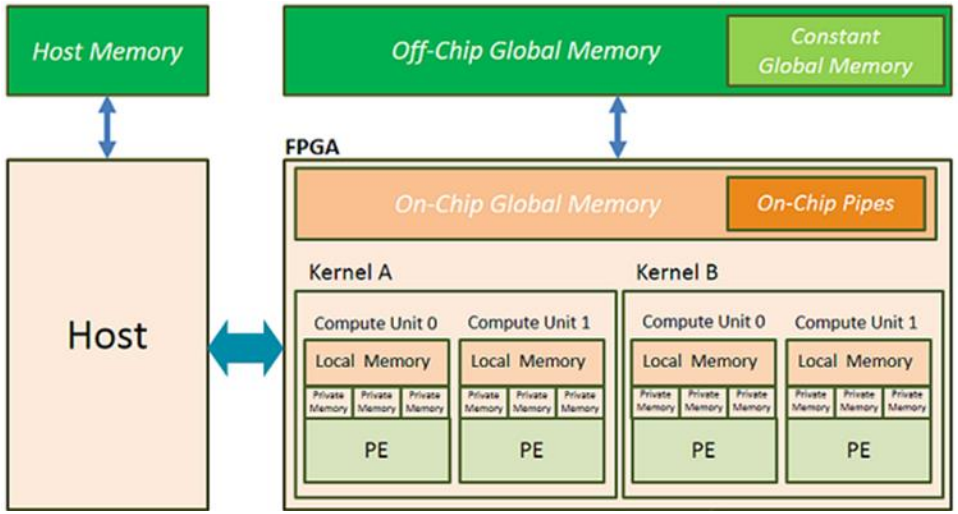
Goal and motivation

- Use the Xilinx acceleration methodology on MicroTCA → Why?
  - Let HDL aside for “application” development, using C-like high-level synthesis languages (HLS) to describe hardware
  - Higher abstraction
  - Reduce development time

Xilinx-AMD acceleration methodology

- Main elements:
  - **Host program**, written in C/C++, runs on a CPU (external or embedded) and makes calls to the XRT and OpenCL API to communicate with the hardware
  - **Kernels** describe hardware functions, written in C/C++ (HLS) or OpenCL, that are implemented in the FPGA
  - Host and kernels exchange data through **global memory**



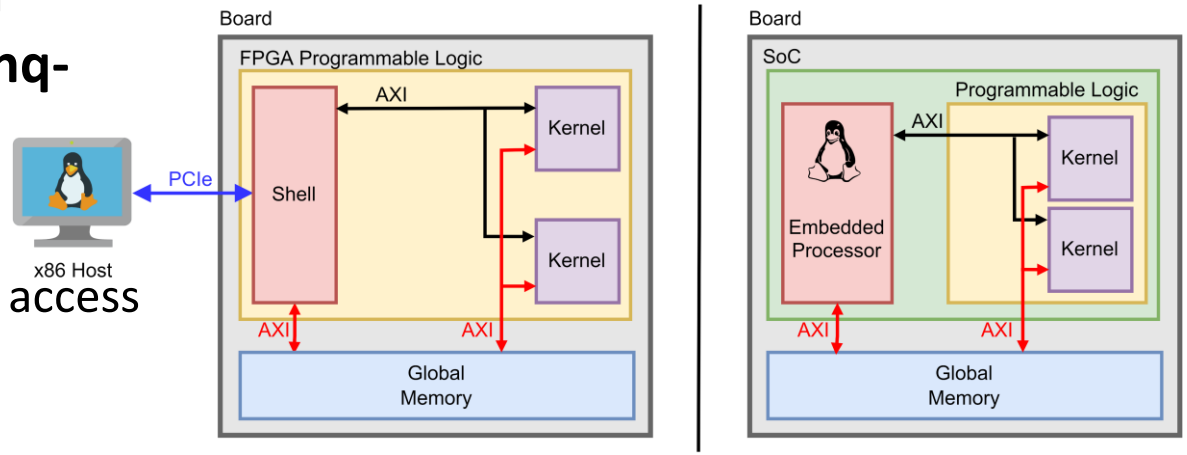


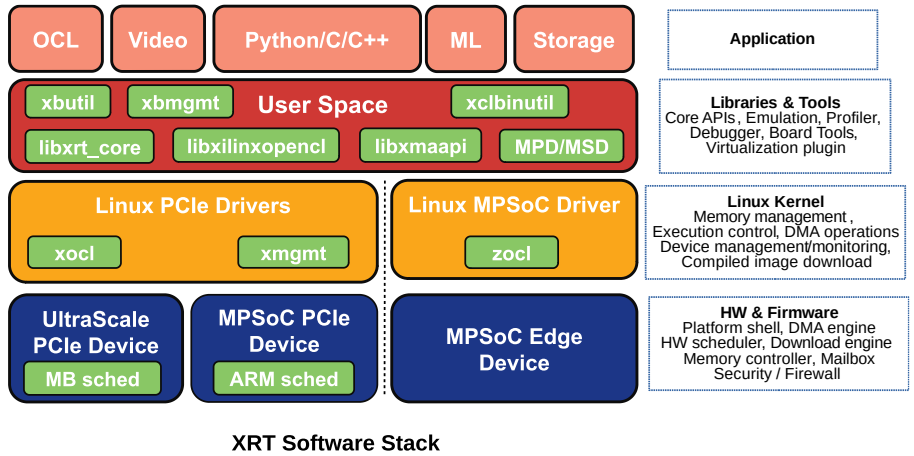
Xilinx-AMD acceleration methodology - Platforms

- This method requires a basic infrastructure, called *Platform*. **Users develop custom platforms for Zynq-based boards!**
- Hardware Platform (Static FPGA region)
  - (Mandatory) AXI for kernel control and global memory access
  - (Mandatory) Clocks, resets, interrupts
  - (Optional) AXI Stream Interfaces

  + (Optional) Application-specific hardware (e.g. ADC/DAC interface, PCIe …)
- ARM Software Platform.
  - A single kernel module (**zocl**) takes care of everything! (Kernel execution management, buffer allocation, memory transfer …)
  - Libraries for accelerated applications

  + Application-specific sources (drivers, libraries, etc…)



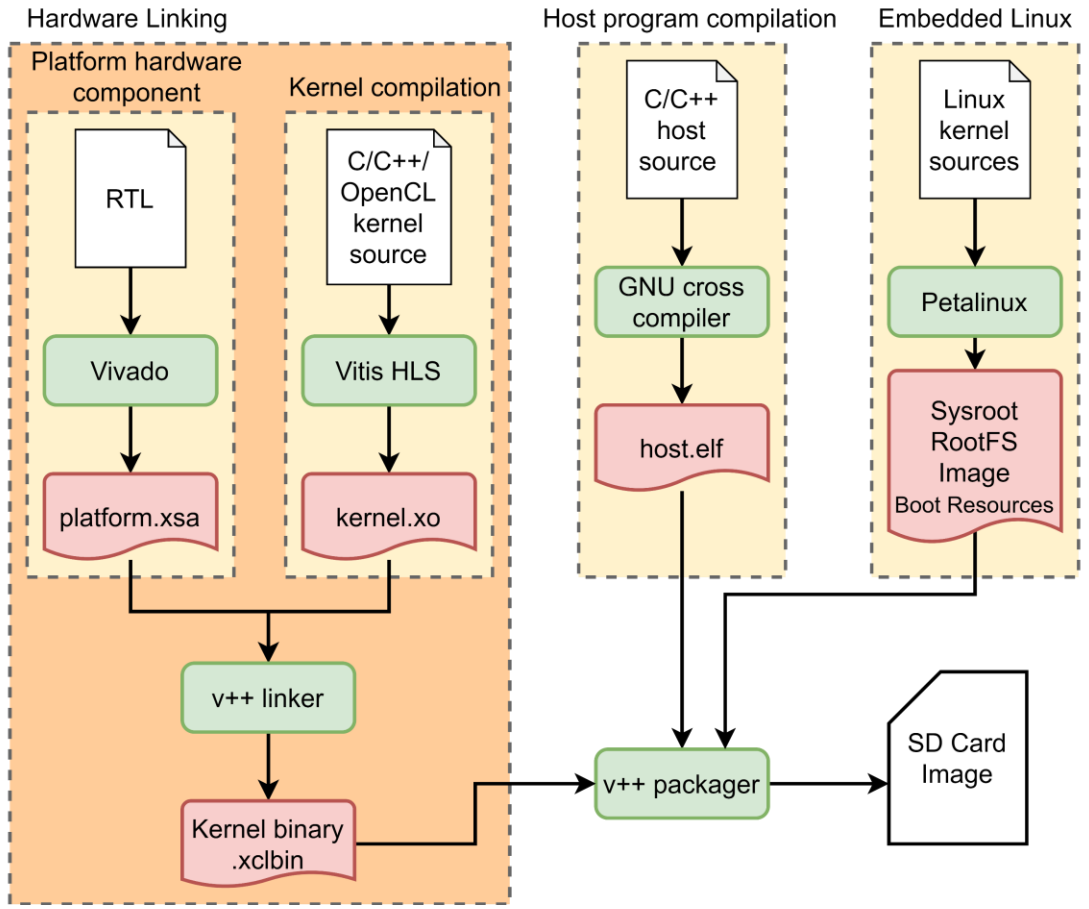


XRT Software Stack

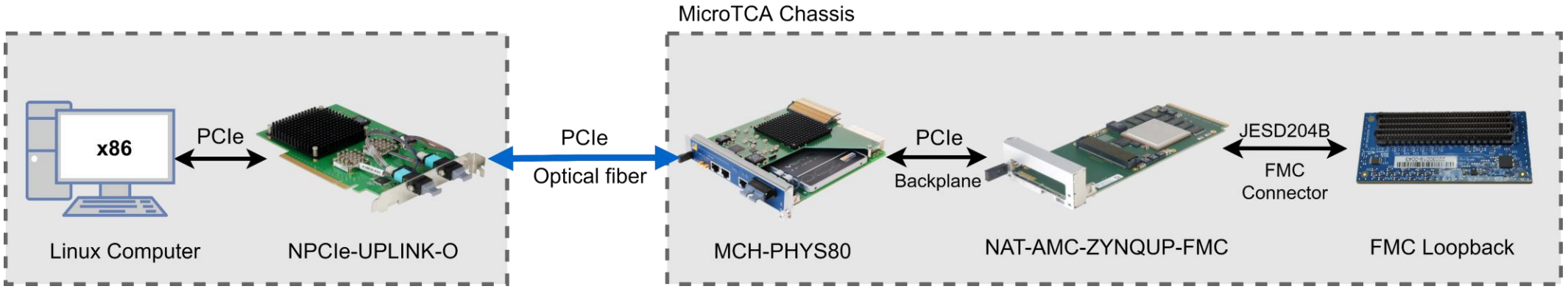
Xilinx-AMD acceleration methodology - Workflow and development tools

Platform
- Hardware platform → Vivado
- Software platform → Petalinux

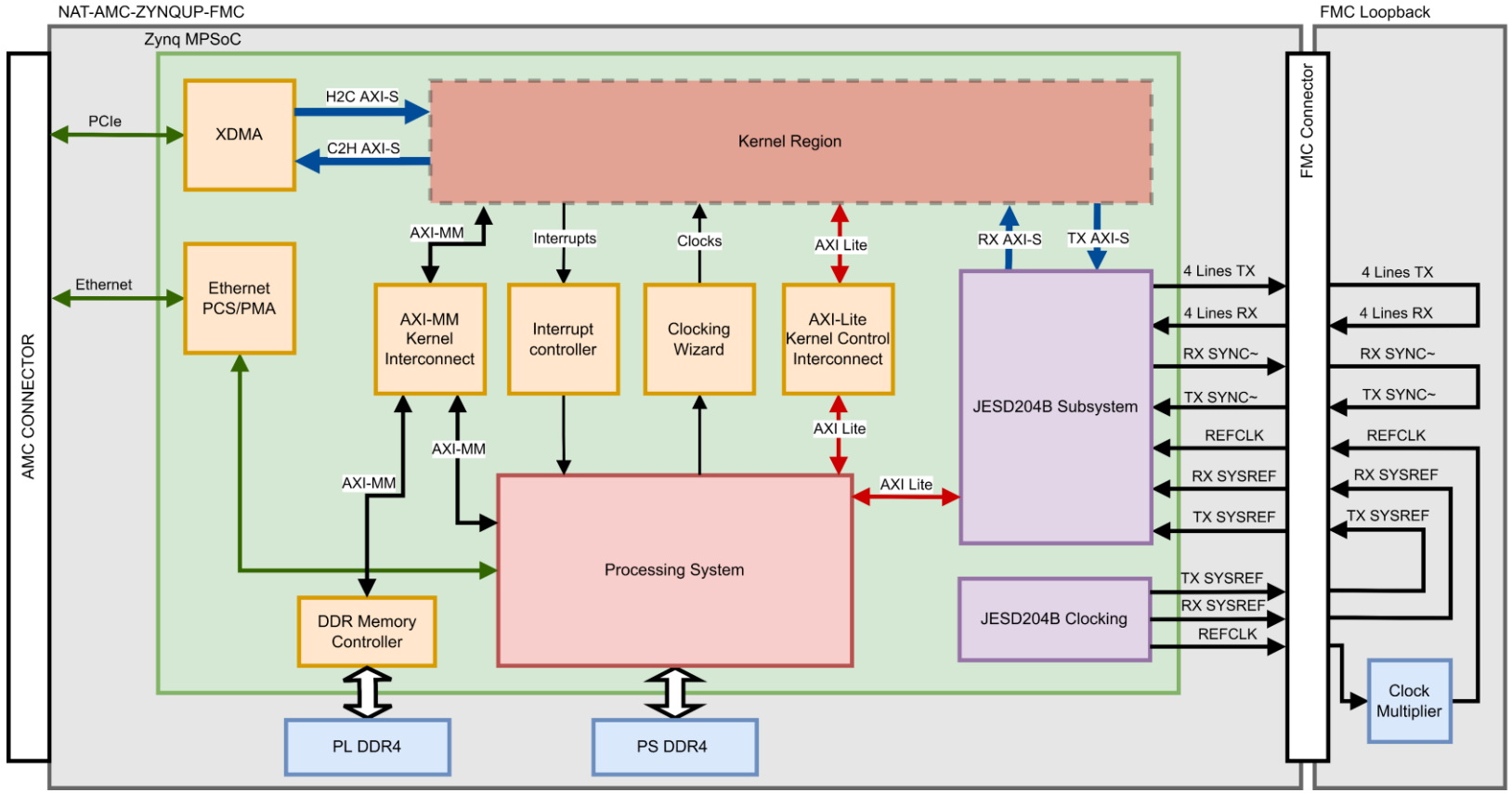
Application
- Kernel compilation → Vitis HLS
- Host compilation → Cross compiler
- Linking and Packaging → Vitis



Hardware setup used



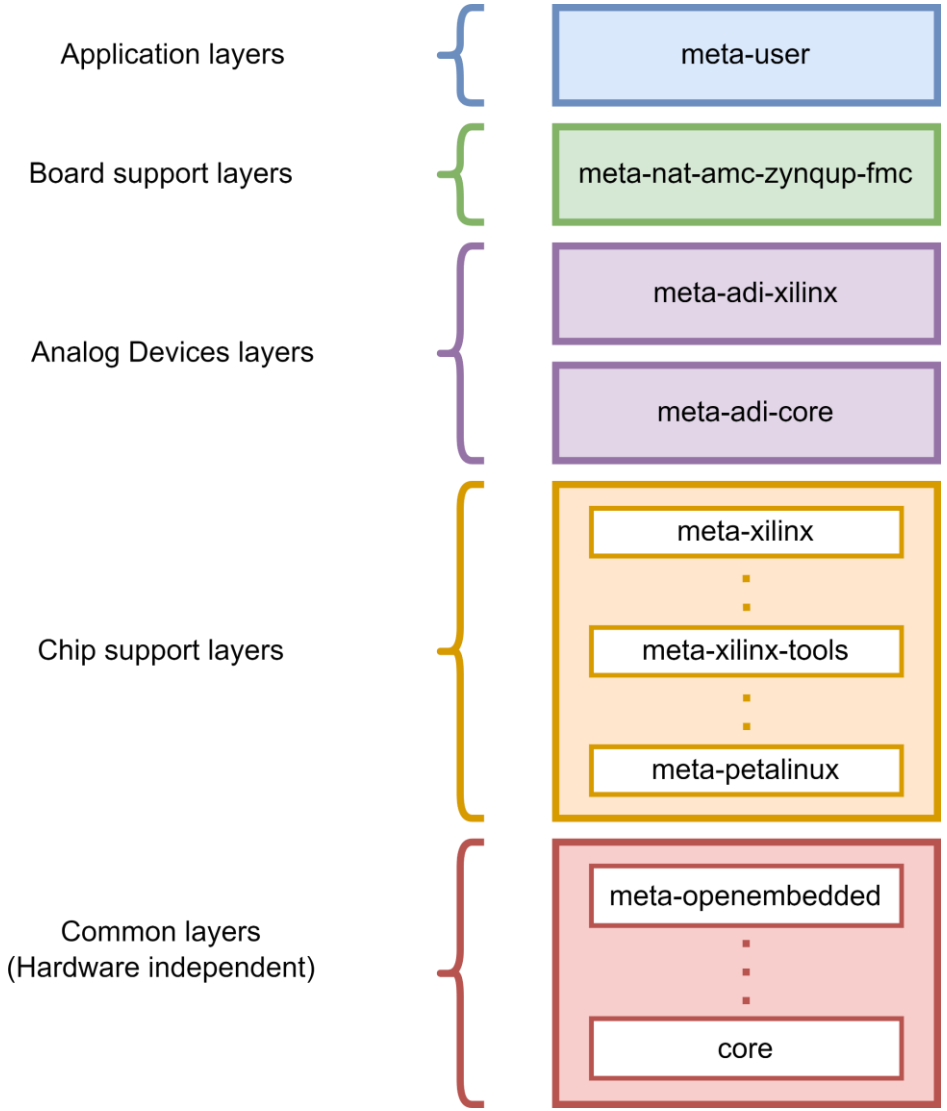
Implemented Hardware Platform



- JESD204B
  - Analog Devices open-source IP blocks
  - 4 TX/RX lanes @ 8 Gbps (easily scalable)
  - The JESD204B signals are routed to the FMC connector, to which a loopback card is connected
- PCIe connectivity
  - 4 Lanes Gen3 @ 8 GT/s
  - XDMA AXI Stream mode
- 2 DDR4 memory banks
  - Kernels can use the Processing System memory or an independent memory bank
- 2 Ethernet interfaces
- AXI Interconnectivity for kernel control and data movement

Software Platform

- Embedded Linux developed using Petalinux (**Yocto-based**)
- Xilinx Runtime Library included in the **meta-xilinx** layer
- Analog devices layers (**meta-adi**)
  - Linux device drivers that manage JESD204 peripherals as well as ADC/DAC and clock chips
  - Drivers are synchronized through a Finite State Machine for proper link bring-up
- Board support layer provided by NAT
- Required device-tree nodes (JESD204B drivers and zocl) added in the user layer, **meta-user**

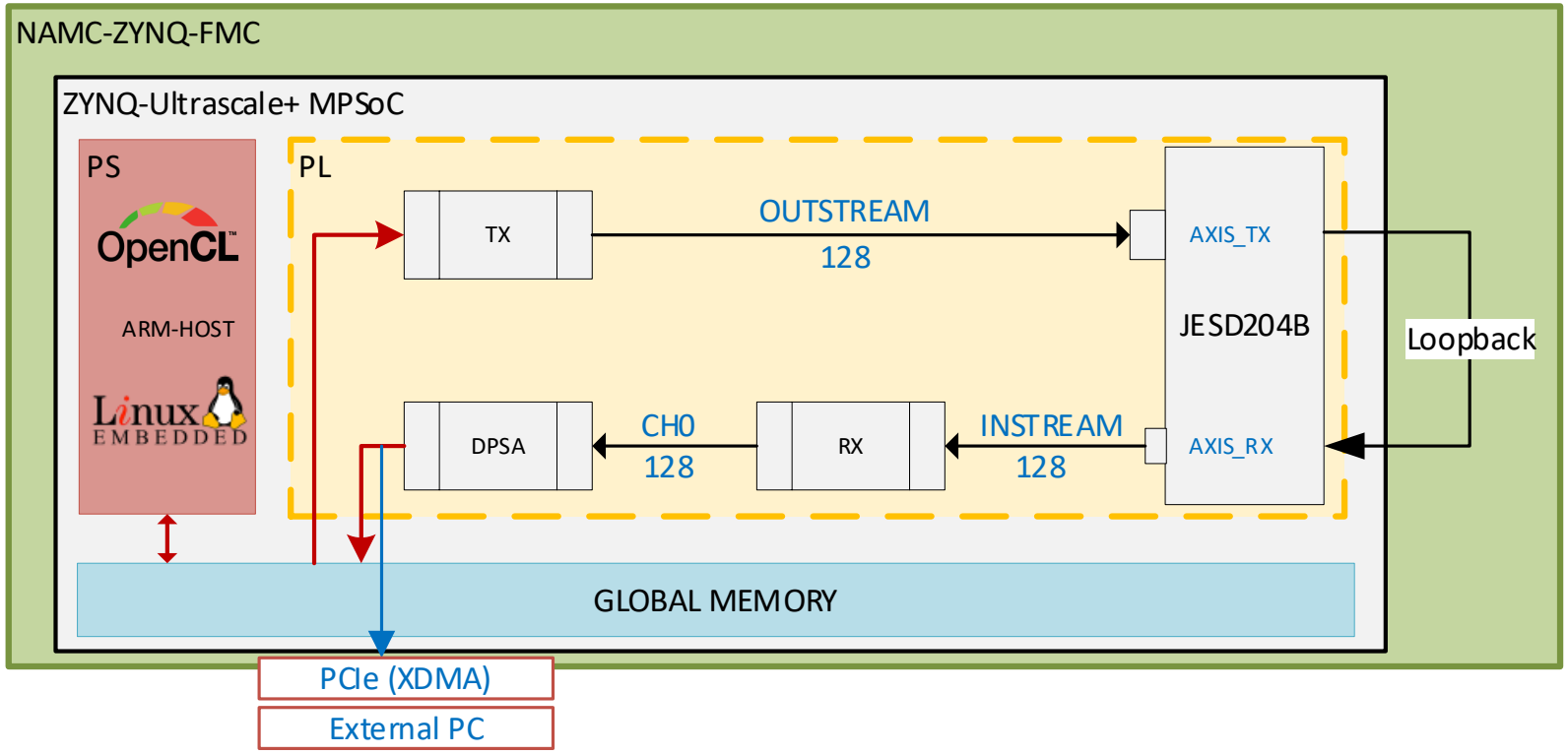


Example use case – Digital Pulse Shape Analysis

- Implementation and verification of a **DPSA** (Digital Pulse Signal Analysis) algorithm to extract physically relevant parameters from the signals provided by BC501A liquid scintillator
- Starting point: Algorithm written in C++ for offline analysis (Spanish Fusion Lab, CIEMAT)
- Validation using signals coming from a real dataset (sampling rate 1GS/s)

Use case – Digital Pulse Shape Analysis

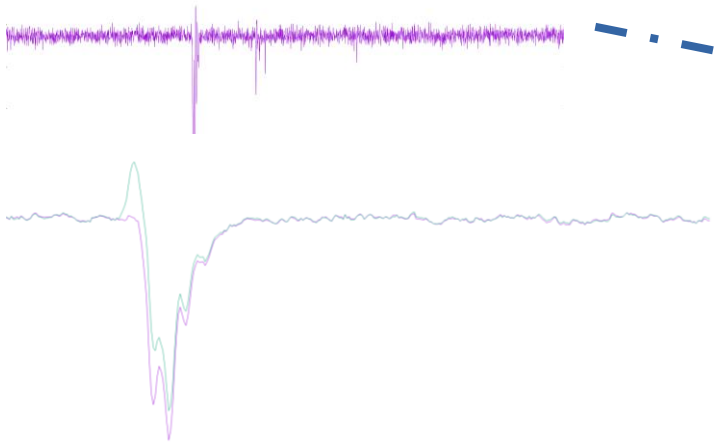
- The host program running in the embedded processor uses the OpenCL API
- Kernels are written in HLS
- The JESD204B transmitter along with the loopback card are used to emulate acquisition
  - The host program stores the signals in global memory, which are sent through the link by a TX kernel



Use case – Digital Pulse Shape Analysis

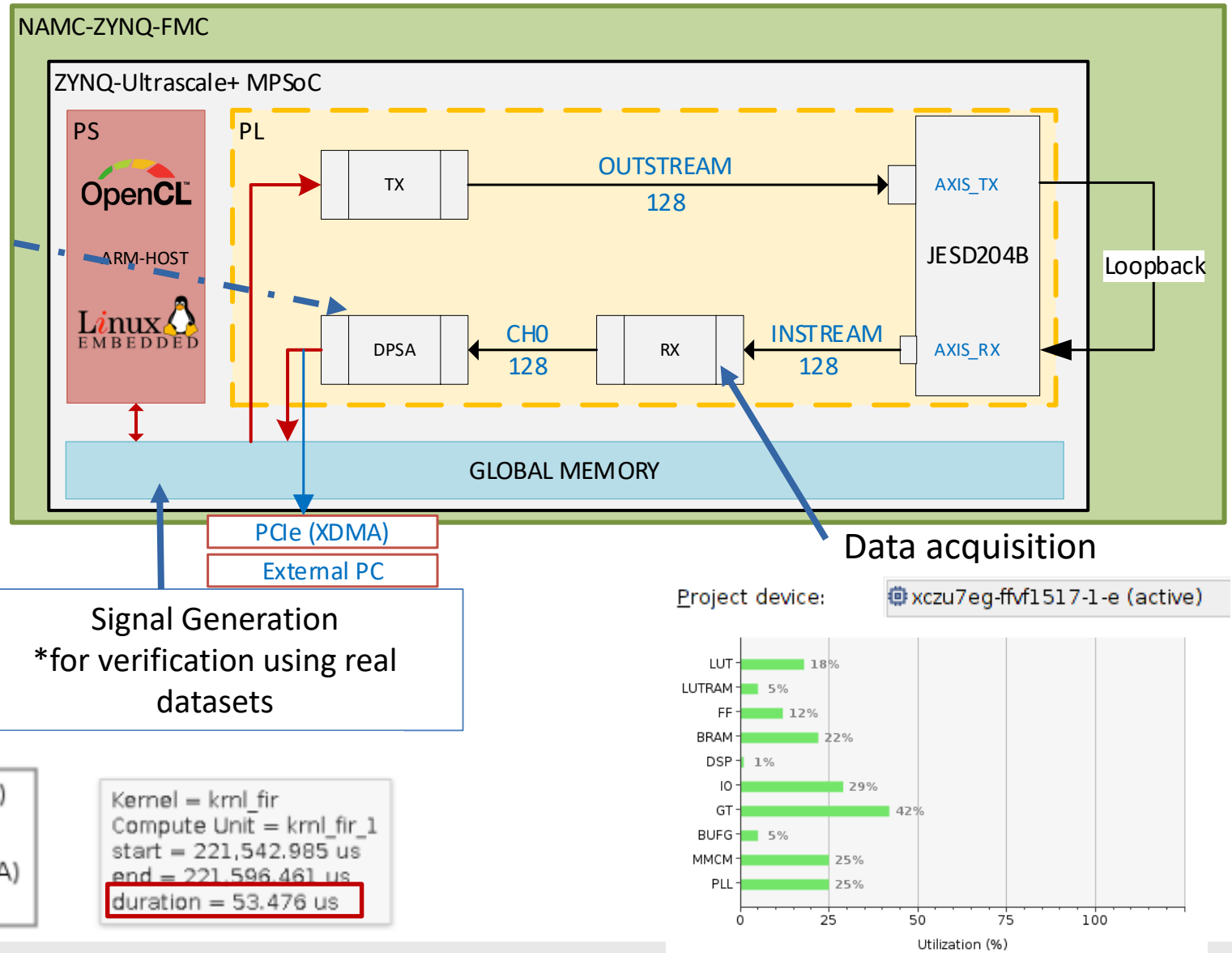
- Sampling rate 1GS/s

1. FIR filter and constant factor discrimination



2. Peak detection = Time, # of peaks per signal and pileups

3. Energies calculation = Integrate over several time intervals



Resources

LUT: 48,144 (20.9 %)
BRAM: 16 (5.13 %)
URAM: 0 (N/A)
Register: 59,070 (N/A)
DSP: 23 (1.33 %)

Kernel = krnl_fir
Compute Unit = krnl_fir_1
start = 221,542.985 us
end = 221,596.461 us
duration = 53.476 us

Summary

- Implementation of a XILINX VITIS platform in an AMC for data acquisition (JESD204B) and processing using HLS/OpenCL.
- The platform can be easily migrated to other Zynq UltraScale+ MPSoC based AMCs
- Implementation of a use case (pulse analysis) demonstrating DAQ/Processing and data movement to host using PCIe
- Use of XILINX standard Linux Kernel drivers (zocl and xdma) and APIs to manage the hardware
- Use of open-source components for JESD204B (IP blocks, Linux Kernel drivers)
- Platform ready for ML applications

Questions?

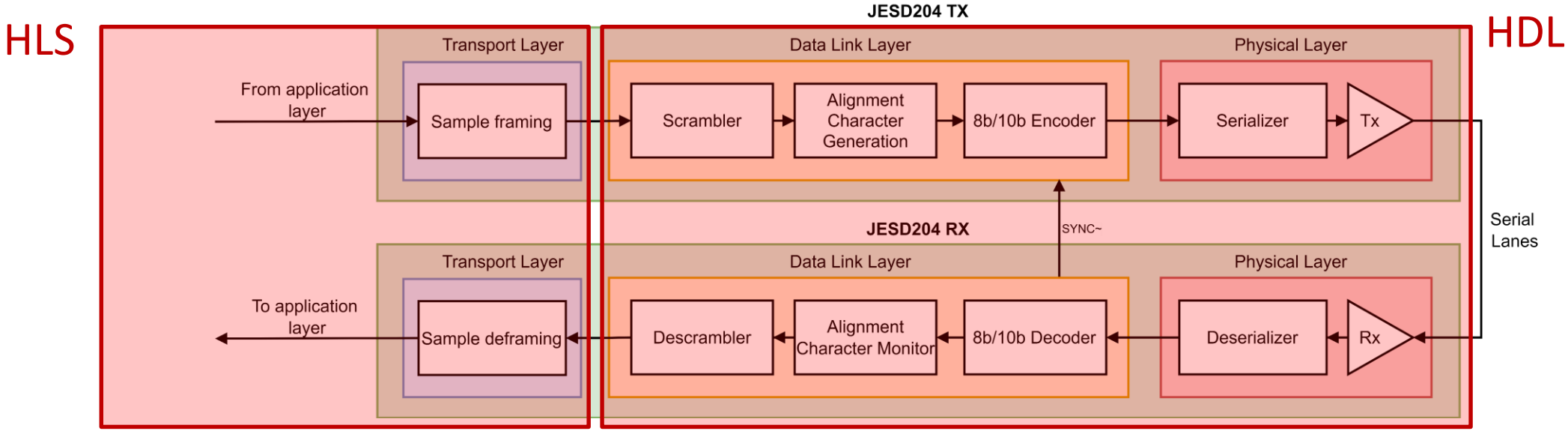
Acknowledgements

1. Proyecto PID2019-108377RB-C33 Funding: MCIN/ AEI / 10.13039/501100011033

2. Proyecto PID2022-137680OB-C33 Funding: MCIN / AEI/ 10.13039/501100011033 / FEDER, UE

JESD204B Implementation

- Physical and Data Link layers implemented using Analog Devices HDL Library. Open-source solution for implementing JESD204B links
- Transport and application layers implemented using HLS



JESD204B subsystem

- Each peripheral has an AXI-Lite based configuration interface
- Samples are sent/received through AXI Stream interfaces
- JESD204B signals are routed to the FMC connector