Using FPGA-based AMC Carrier Boards for FMC to Implement Intelligent Data Acquisition Applications in MTCA.4 Systems using OpenCL

Mariano Ruiz, Antonio Carpeño, **Daniel Rivilla**, and Victor Costa

Universidad Politécnica de Madrid

daniel.rivilla@i2a2.upm.es

GRUPO DE INVESTIGACIÓN EN INSTRUMENTACIÓN Y ACÚSTICA APLICADA

MINISTERIO DE CIENCIA E INNOVACIÓN

AGENCIA ESTATAL DE INVESTIGACIÓN

23rd Real Time Conference

POLITÉCNICA

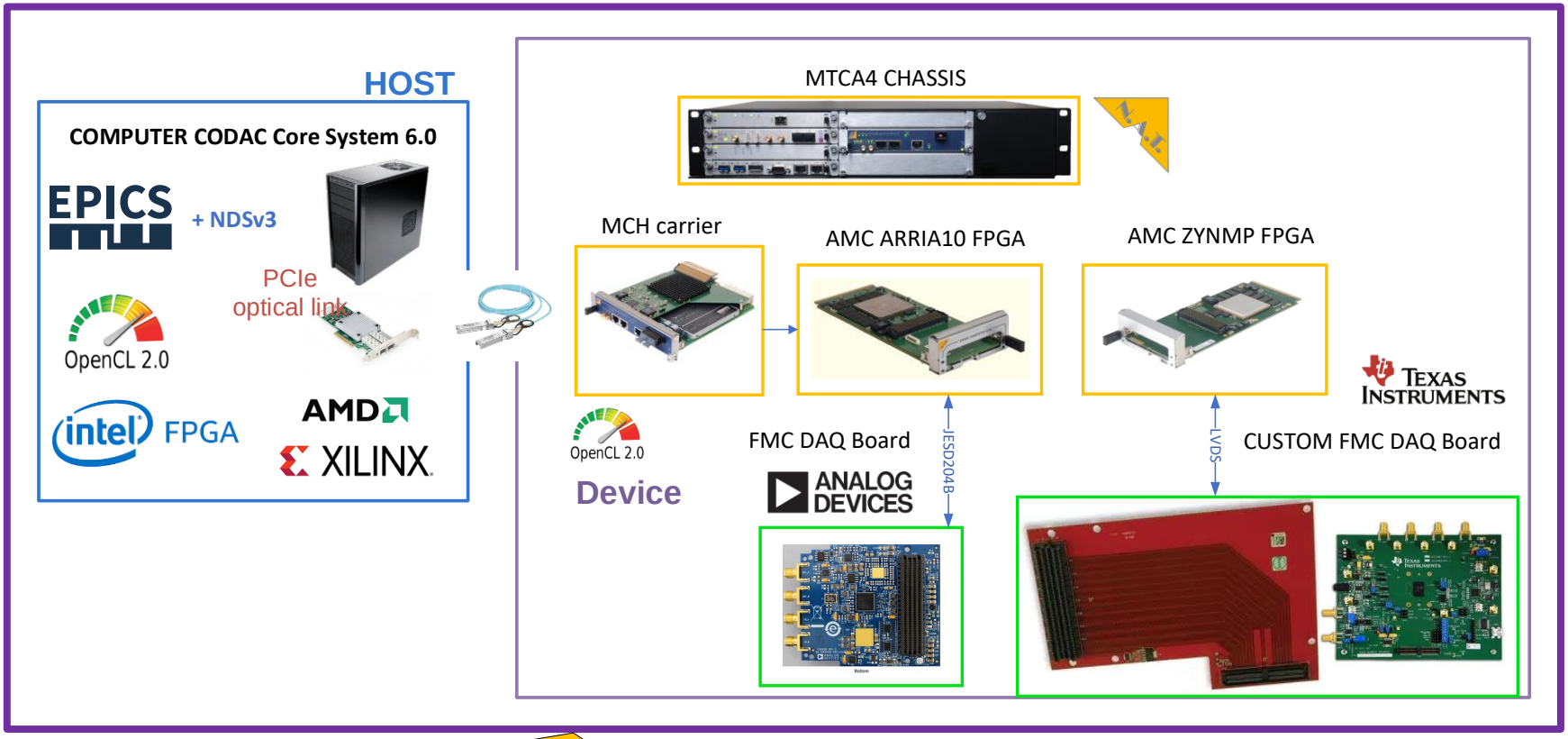
Motivation

- Development of advanced FPGA-based data acquisition systems in MTCA.4 using the hardware acceleration methodologies proposed by Intel FPGA and XILINX. Design cycle based on:
  - The use of FPGAs or SoCs available in AMC boards for MTCA.4 systems
  - The integration of high-speed ADC modules in FMC format linked with the FPGA by LVDS and JESD204B interfaces
  - The development of data acquisition control and processing tasks are developed using OpenCL and/or HLS
  - Software layers implement an Interface with EPICS framework
- Main goal is the reduction of the development time

Outline

1. Software and hardware setup needed for the development
2. Approach and design methodology (hardware acceleration)
   - Intel FPGA and XILINX AMD
   - Reference designs and BSPs
3. Applications to:
   - AMC-NAMC-ARRIA10 + ADC (JESD204B)
   - AMC-NAMC-ZYNQMP + ADC (LVDS)
4. Data Acquisition and processing by using OpenCL/HLS kernels
5. Software layers to interface with EPICS
6. Results and conclusions

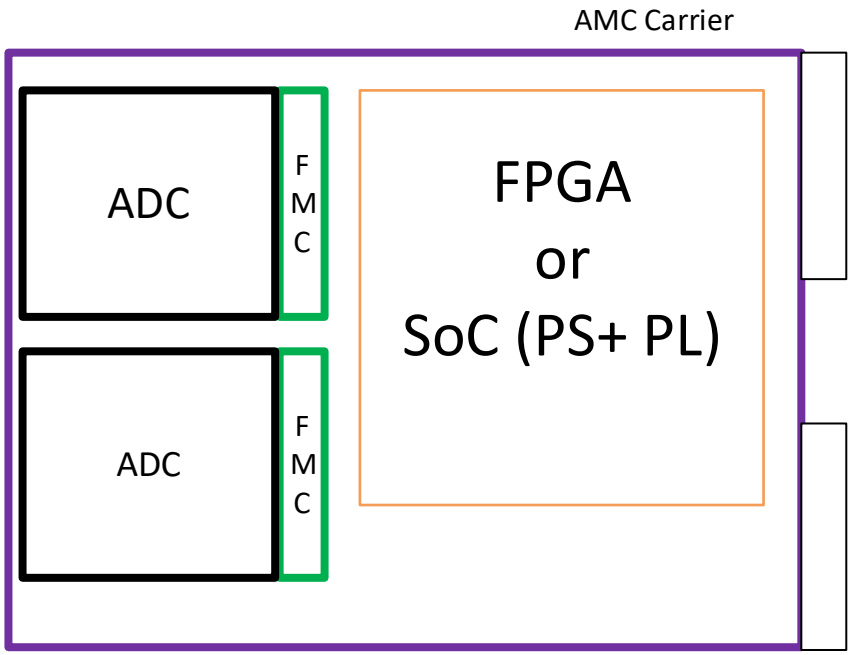
1.- MTCA.4 platform used for the development



Maximize the use of COTS elements to reduce the development time

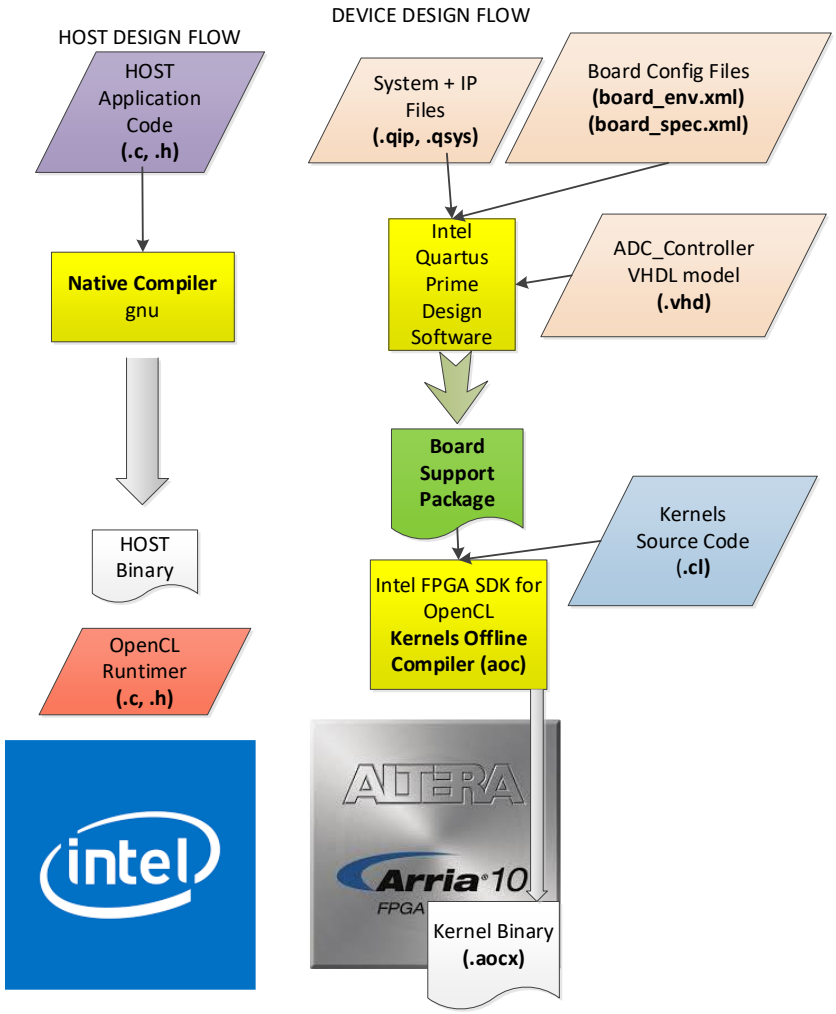
2 Approach

- SoC (PS) or FPGA(PL) implements a high-speed interface with FMC-based DAQ devices
- FPGA (PL) HW for DAQ and processing implemented using OpenCL/HLS (designed following hardware acceleration design flow approach)
- Host computer (ARM for SoC or external PC) running the OpenCL runtime or XRT)



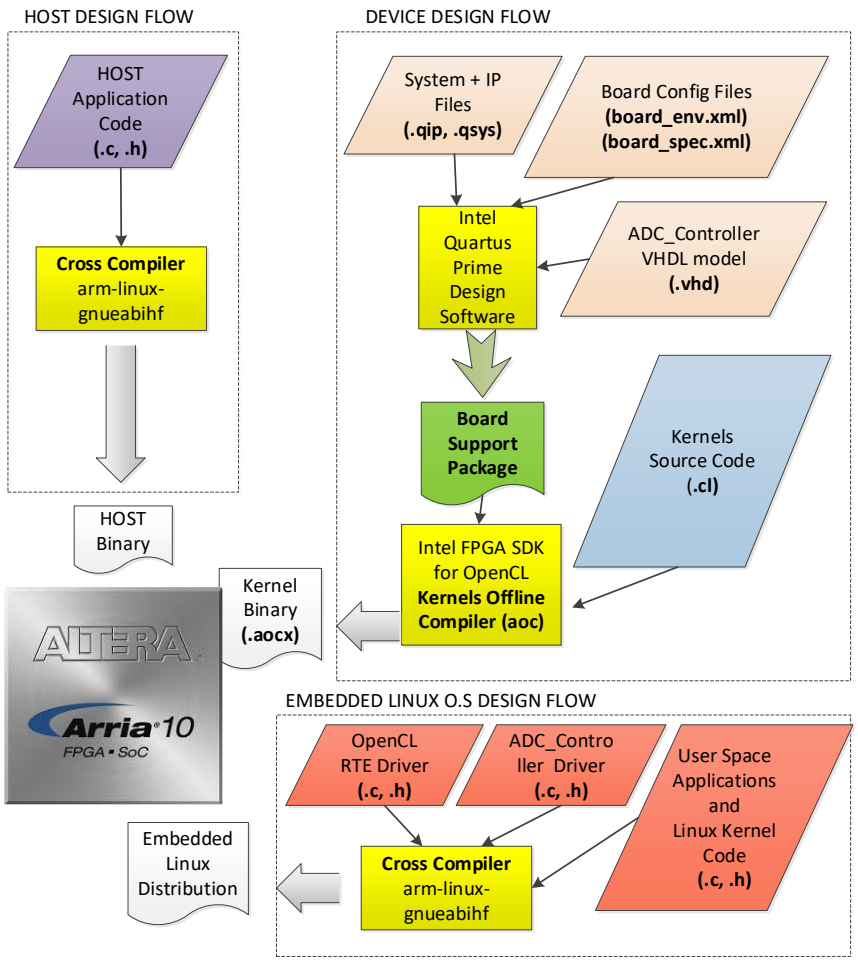
2-Design Methodology using Intel FPGA (for PCIe platforms)

- FPGA BSP:
  - Basic design implemented using Platform designer and Quartus
  - Interface with FMC using JESD204B
  - Static part. Incremental compilation flow
  - Connection with Host through PCIe
- Custom kernels
  - Interface with static part Avalon-ST or AXI-ST
  - User algorithms
  - Partial reconfiguration



2-Design Methodology using Intel FPGA (for SoC platforms)

- FPGA BSP:
  - Basic design implemented using Platform designer and Quartus
  - Interface with FMC using JESD204B
  - Static part (incremental compilation)
  - Connection with Host through Avalon-MM or AXI-MM
- Custom kernels
  - Interface with Static part Avalon ST or AXI-ST
  - User algorithms
  - Partial reconfiguration



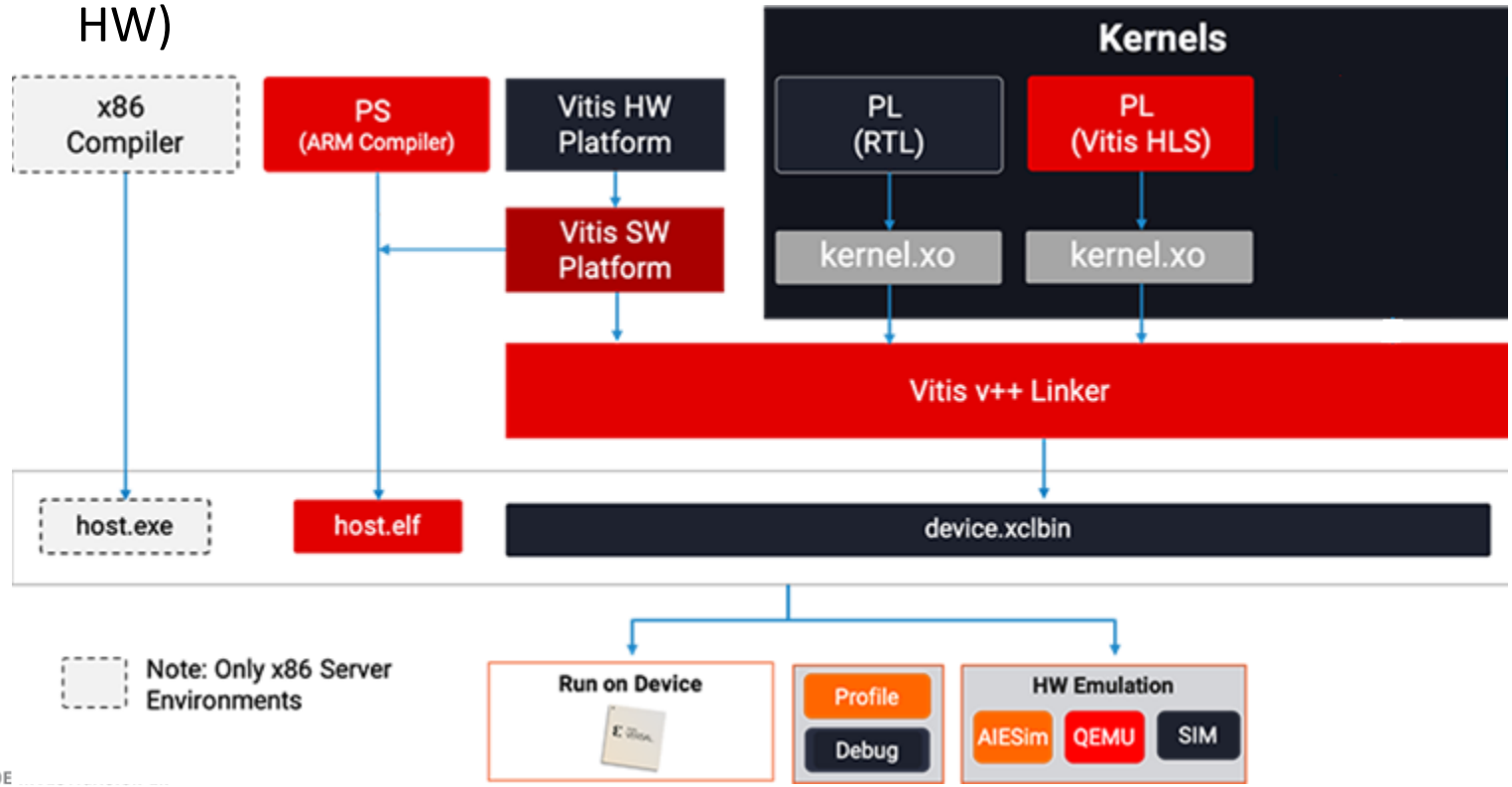
2-Design Methodology using AMD XILINX

BSP: Vitis HW Platforms

- Open for embedded platforms
  - Interface with FMC using AXI
  - Static part
  - Closed for PCIe (only ALVEO HW)

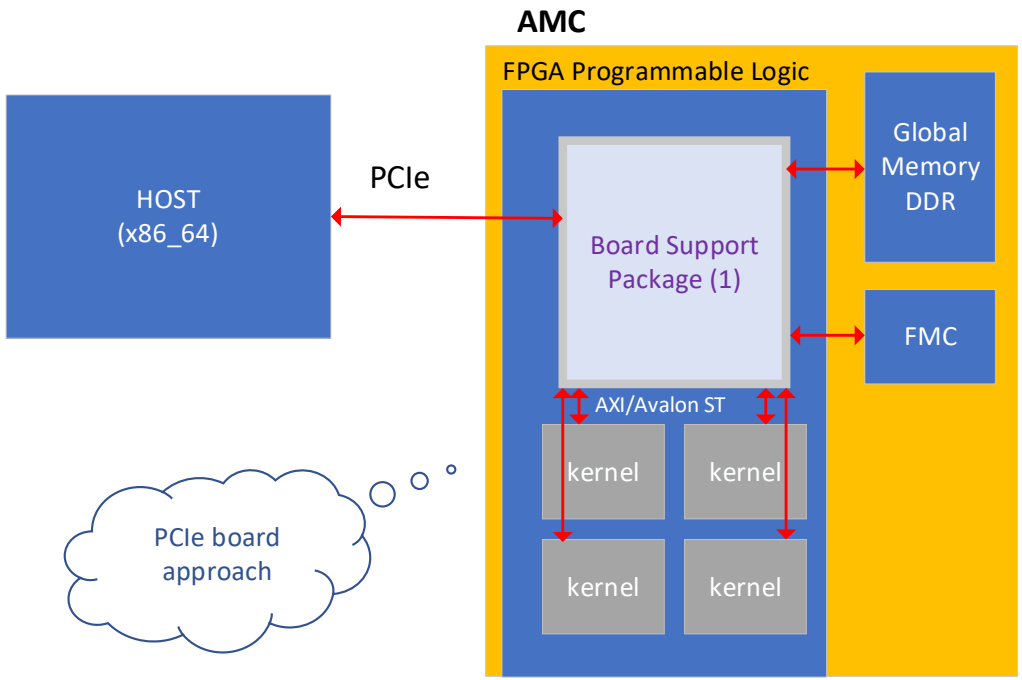
Custom kernels:

- Interface with AXI
- RTL, HLS or OpenCL
- Partial reconfiguration Optional



2.- Application of Reference designs for COTS AMC boards (FPGA-PL)

- Interface to HOST x86_64 using PCIe
- HOST runs OpenCL runtime environment
  - Linux Kernel Module
  - Access to Global Memory
- BSP: basic design (HDL)
  - Provides streaming outputs
  - Interfaces the FMC



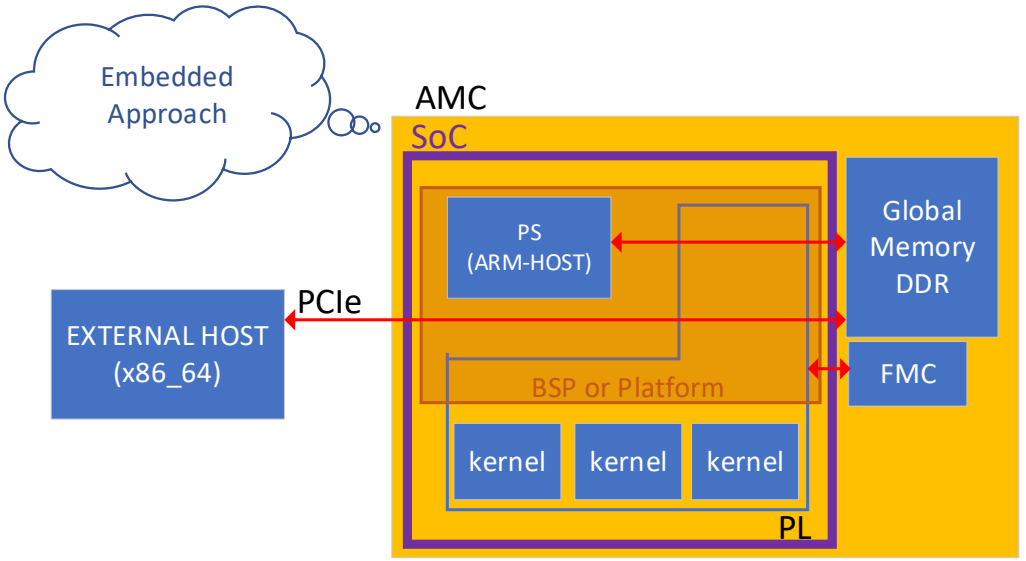
✓ Applicable to commercial and custom designed products (Intel FPGA)

× Design available only for ALVEO HW (XILINX)

1. IntelFPGA: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-guidance/opencl-bsp-support.html
   - Intel® FPGA SDK for OpenCL™ Intel® Arria® 10 GX FPGA Development Kit Reference Platform Porting Guide

2.- Application of Reference designs for COTS AMC boards (SoC)

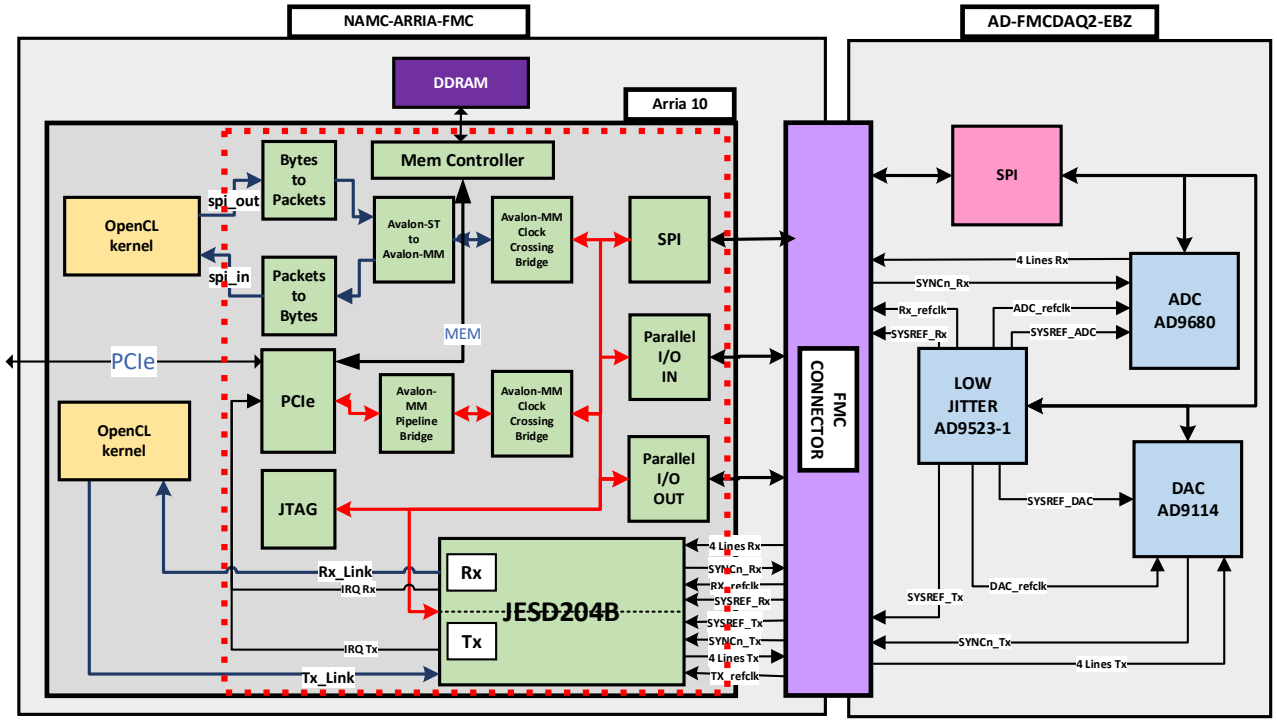
- ARM-HOST (PS) running OpenCL runtime environment
  - Linux Kernel Module
  - Access to Global Memory
- BSP: basic design (HDL)
  - Provides streaming outputs
  - Interfaces the FMC
- External HOST interfaces using PCIe (access to AMC global memory)

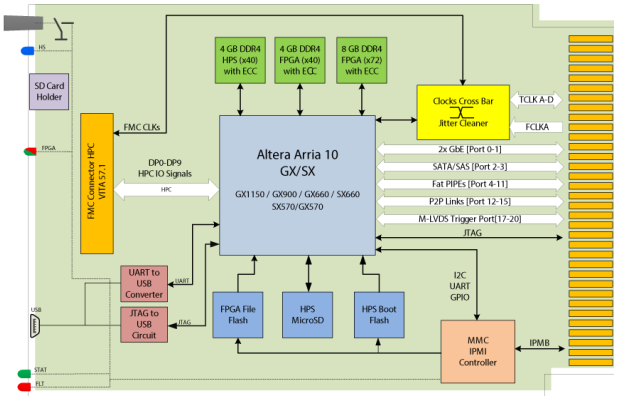


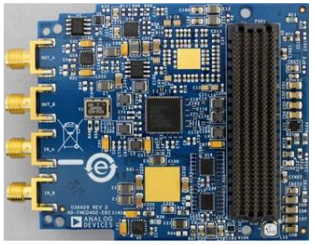
✓ Applicable to commercial and custom designed products (Intel FPGA and XILINX)

1. IntelFPGA: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-guidance/opencl-bsp-support.html
   - Intel® FPGA SDK for OpenCL™ Intel® Arria® 10 SoC FPGA Development Kit Reference Platform Porting Guide
2. XILINX VITIS Embedded Platform Source: https://github.com/Xilinx/Vitis_Embedded_Platform_Source
   - Vitis_Embedded_Platform_Source/Xilinx_Official_Platforms

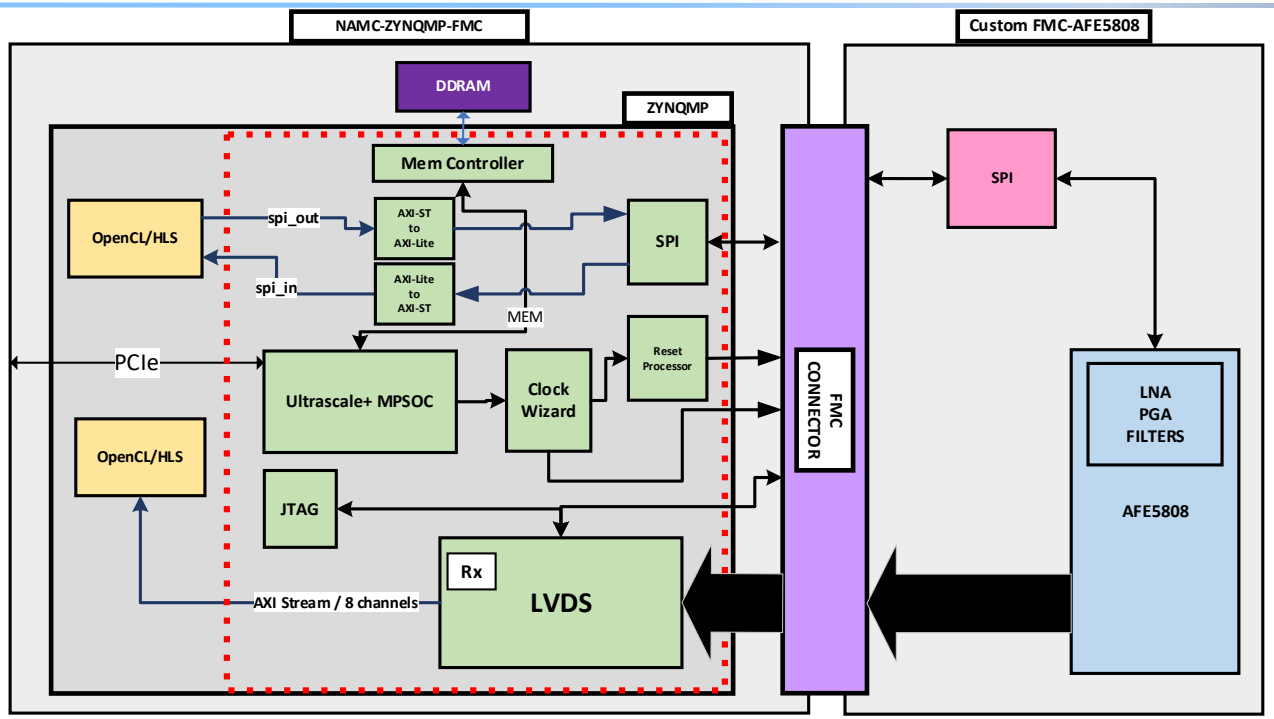
3- Application to NAMC-ARRIA10-FMC & AD-FMCDAQ2-EBZ

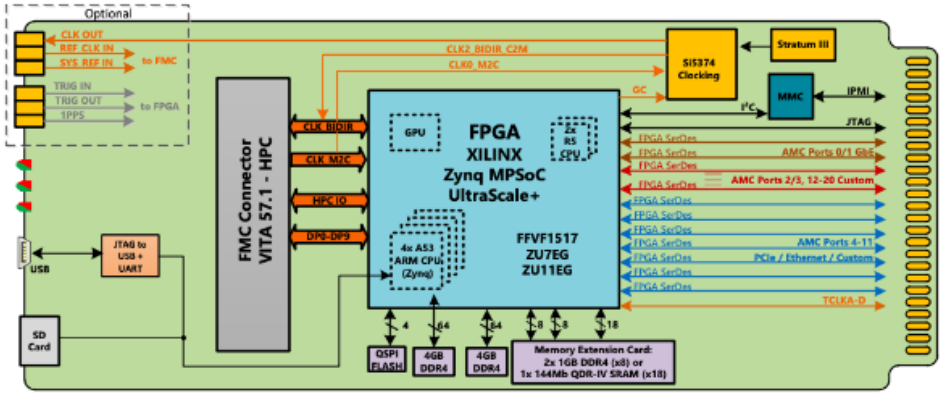


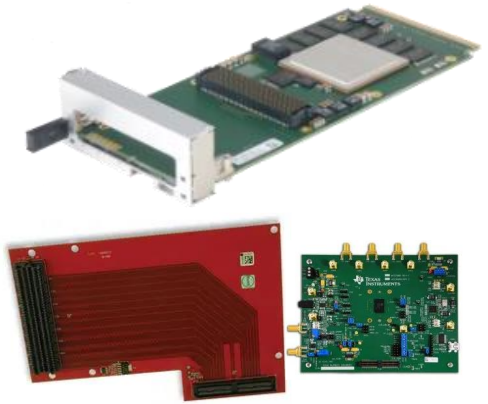




3.- Application to NAMC-ZYNQ-FMC AFE5808AEVM

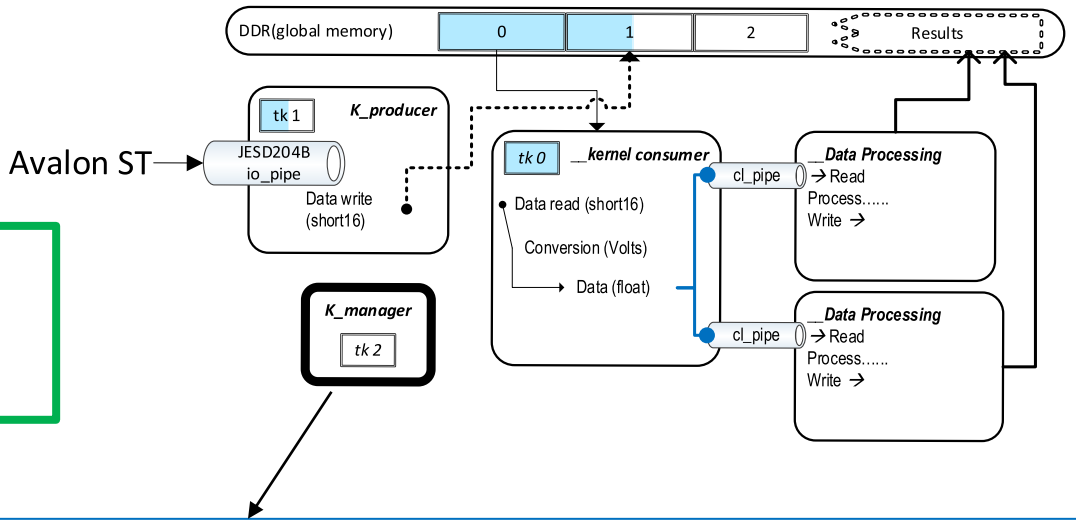






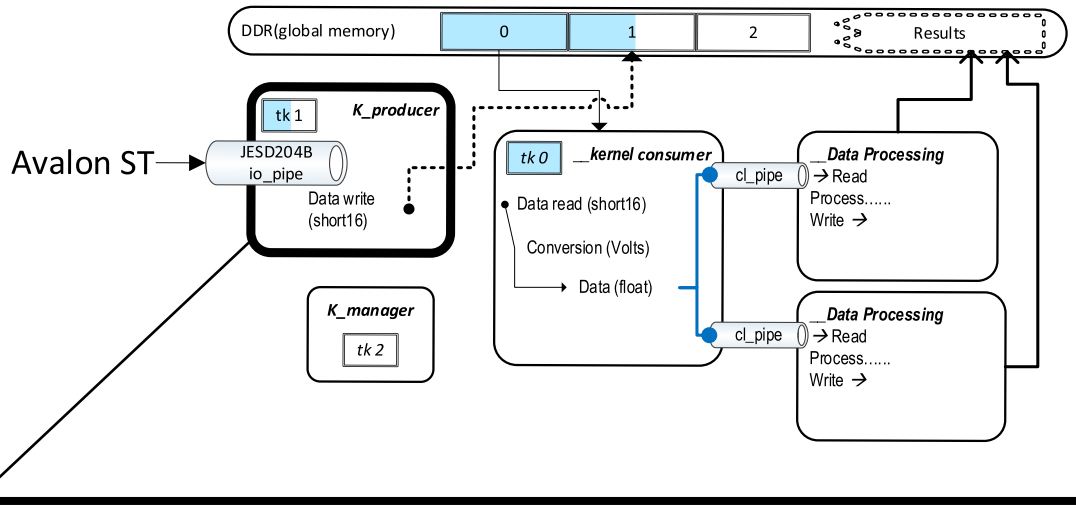
4.- Ping-Pong solution for DAQ and Processing (Intel FPGA OpenCL)

BSP: 15% of LE, 9% of MB.
Consumer-producer logic: 2% of LE and 3% of MB



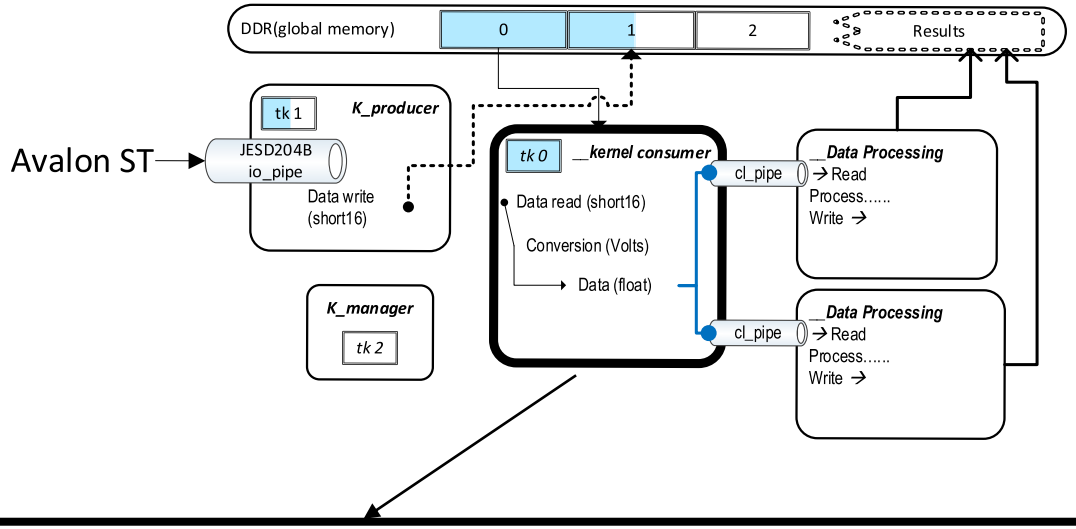

```
__kernel void K_manager(
                const uint bufferSize,
                const uint numberOfBlocks,
                const uint numberOfSubBuffers){
        __private uint offset = 0;

        for (uint i=0;i<numberOfBlocks;i++){
                offset = read_channel_intel(IRIOCL_ConsumerToManager);
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);

                write_channel_intel(IRIOCL_managerToProducer,offset);
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);
        }
}
```

4.- Ping-Pong solution for DAQ and Processing (Intel FPGA OpenCL)




```
__kernel void k_producer(
        __global __attribute__((aligned(64))) volatile ushort16* restrict h_Buffer, const uint bufferSize, const uint numberOfBlocks){
        //Kernel Body
        for (uint i=0;i<numberOfBlocks;i++){
                offset = read_channel_intel(IRIOCL_managerToProducer);
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);

                #pragma ivdep
                for (uint j = offset;j < offset+bufferSize ;j++){
                        hadcsamples = read_channel_intel(rx_link);
                        prv_buffer.s0 = (((hadcsamples.s4 >>(8*3)) & (0xFF))<<8) + ((hadcsamples.s5 >>(8*3)) & (0xFF));
                                .
                                .
                        prv_buffer.sF = (((hadcsamples.s2 >>(8*0)) & (0xFF))<<8) + ((hadcsamples.s3 >>(8*0)) & (0xFF));

                        h_Buffer[j] = prv_buffer;
                        }
                }
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);
                write_channel_intel(IRIOCL_ProducerToConsumer,offset);
```

4.- Ping-Pong solution for DAQ and Processing (Intel FPGA OpenCL)

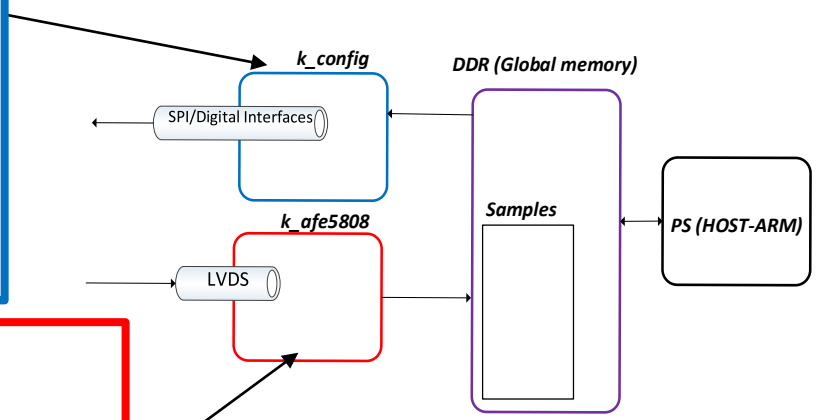



```
__kernel void consumer(
                __global __attribute__((aligned(64))) volatile ushort16* restrict h_Buffer, const uint bufferSize, const uint numberOfBlocks,
                __global ulong* restrict result, __global uchar* restrict clearBuffers

){//Kernel Body
                offset = read_channel_intel(IRIOCL_ProducerToConsumer);
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);

                write_channel_intel(IRIOCL_Consumer_To_pulse_count,offset);
                write_channel_intel(IRIOCL_Consumer_To_Campbelling,offset);

                error = read_channel_intel(IRIOCL_pulse_count_To_Consumer);
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);

                error = read_channel_intel(IRIOCL_Campbelling_To_Consumer);
                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);

                mem_fence(CLK_GLOBAL_MEM_FENCE | CLK_CHANNEL_MEM_FENCE);
                write_channel_intel(IRIOCL_ConsumerToManager,offset);
```

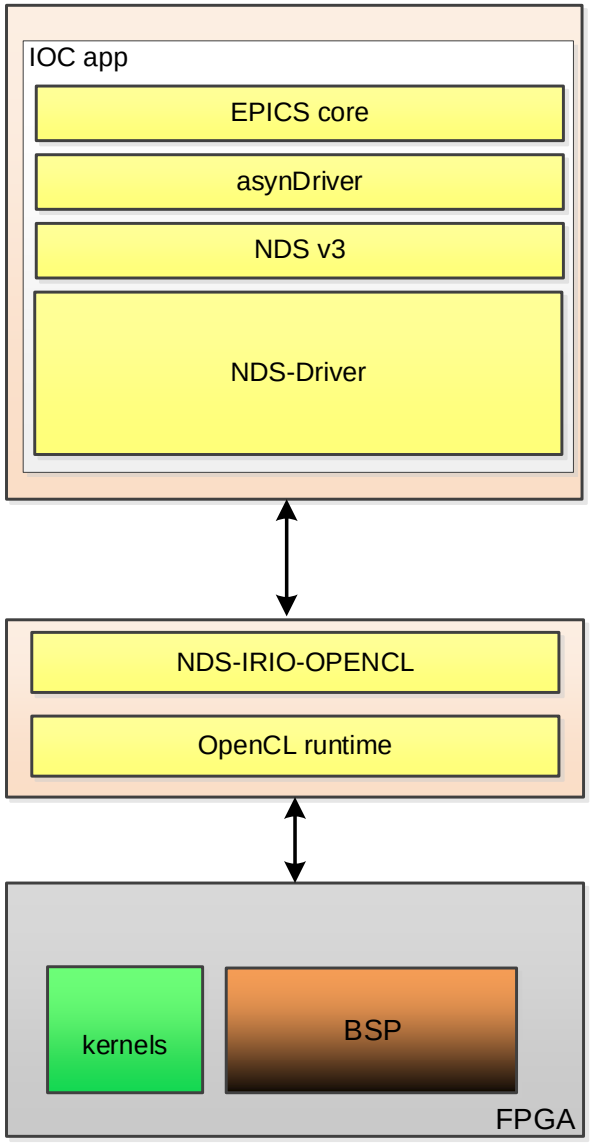
4.- Acquiring data from ADC using streaming (XILINX AMD HLS)


```
extern "C" {
void k_config(hls::stream<pkt_config> &config_out,
                                        short k_excitation_chn,
                                        int k_start_acquisition,
                                        int k_acquisition_samples,
                                        short k_test){

#pragma HLS INTERFACE axis port = config_out

        pkt_config value;

        value.data.range(95,80) = k_test;
        value.data.range(79,64) = k_excitation_chn;
        value.data.range(63,32) = k_acquisition_samples;
        value.data.range(31,0) = k_start_acquisition;

        config_out.write(value);
}
}
```

```
extern "C" {
void k_afe5808(hls::stream<pkt_afe5808> &afe5808_in,
                pkt_afe5808_out *afe5808_out,
                int samples){

#pragma HLS INTERFACE axis port=afe5808_in
#pragma HLS INTERFACE m_axi port = afe5808_out offset = slave bundle = gmem
#pragma HLS INTERFACE s_axilite port = afe5808_out bundle = control
#pragma HLS INTERFACE s_axilite port = samples bundle = control
#pragma HLS INTERFACE s_axilite port = return bundle = control

        for (unsigned int i = 0; i < samples; i++){
        #pragma HLS LOOP_TRIPCOUNT min = samples max = samples
        #pragma HLS PIPELINE II = 1
                pkt_afe5808 value = afe5808_in.read();
                afe5808_out[i] = value.data;
        }
}
}
```



5 Interface with EPICS using ITER Nominal Device Support



Two main layers

- NDS-IRIO-OPENCL
  - Interface with OpenCL runtime
  - Based on NDS ITER driver development methodology for diagnostics
- NDS-EPICS
  - Connect any NDS device driver to EPICS

6. Conclusions

- Advantages:
  - Simplification of the development of advanced HW applications using OpenCL and HLS
  - Use of OpenCL runtime that avoids the use of custom Linux kernels modules and API to interface with the FPGA-PL
  - XILINX and Intel FPGA provide reference designs in git repositories (they can be adapted to custom HW easily)
- Drawbacks:
  - BSP development stills needs the use of HDL to implement the interface
  - Documentation from the manufacturers not completed yet

7. References

- [1] *Development of deep learning applications in FPGA-based fusion diagnostics using IRIO-OpenCL and NDS.* M Astrain, M Ruiz, A Carpeño, S Esquembri, D Rivilla. Fusion Engineering and Design 168, 112393
- [2] *Nominal Device Support (NDSv3) as a Software Framework for Measurement Systems in Diagnostics*. R Lange, M Astrain, V Costa, J Moreno, D Rivilla, M Ruiz, D Sanz. 18th International Conference on Accelerator and Large Experimental Physics
- [3] *Real-Time Implementation of the Neutron/Gamma discrimination in an FPGA-based DAQ MTCA platform using a Convolutional Neural Network*. M Astrain, M Ruiz, AV Stephen, R Sarwar, A Carpeño, S Esquembri, IEEE Transactions on Nuclear Science 68 (8), 2173-2178
- [4] *A methodology to standardize the development of FPGA-based high-performance DAQ and processing systems using OpenCL*. M Astrain, M Ruiz, A Carpeño, S Esquembri, E Barrera, J Vega. Fusion Engineering and Design 155, 111561
- [5] *Application of Heterogeneous Computing Techniques for the Development of an Image-Based Hot Spot Detection System Using MTCA*. S Esquembri, J Nieto, A Carpeño, M Ruiz, M Astrain, V Costa, A de Gracia. IEEE Transactions on Nuclear Science 68 (8), 2151-2158

Acknowledgments

- This work was supported under grant PID2019-108377RB-C33 funded by MCIN/AEI/ 10.13039/501100011033 and Grant PEJ-2019-AI/TIC-14507.