Methodology to standardize the development of FPGA-based intelligent DAQ and processing systems on heterogeneous platforms using OpenCL

M. Astrain¹, M. Ruiz¹, S. Esquembri¹, A. Carpeño¹, E. Barrera¹, J. Vega²

¹Instrumentation and Applied Acoustic Research Group, Universidad Politécnica de Madrid, Madrid, Spain

²Laboratorio Nacional de Fusión, CIEMAT, Madrid, Spain

miguel.astrain@i2a2.upm.es
Technical University of Madrid

• With nearly 40,000 students and an annual budget of 400M€, UPM is one of the leading technical universities in Spain. Its organized in 20 engineering schools and 200 Research Groups

• UPM one of the top Spanish Universities in the European Research Ranking

• I2A2-UPM, research and development group working in advanced instrumentation systems:
  • Embedded platforms using SoCs, FPGAS, Embedded Linux
  • Software development including development of EPICS device supports/drivers
  • Collaborator of ITER CODAC since 2009
  • Specific training in Embedded Systems, FPGA and SoCs development
  • Engineers from UPM working in the EPICS community (FRIB and ITER)
OUTLINE

- Motivation
- System Architecture
- OpenCL standard
- Development cycle
- Results
- Conclusions
Motivation

• Simplify the development and integration of advanced DAQ systems
  DAQ system using FPGAs
  • Can we avoid or minimize the use of Hardware Description Languages?
  • Can we standardize the DAQ functionality to shift development effort only to the data processing?

• Standardize the integration of EPICS device drivers
  • Can we standardize the software to simplify the development/maintenance of applications?
    • asynDriver
    • Nominal Device Support
    • Same API and Kernel Linux Device Driver for all hardware devices (using OpenCL).
Our previous experience with DAQ-FPGA and EPICS

IRIO  https://github.com/irio-i2a2

miguel.astrain@i2a2.upm.es  Slide 5
FPGA DAQ-based systems in ITER Fast Controllers

- Reusable for different applications
- Only Products from One Manufacturer
- Design Using LabVIEW/FFPA
- Hardware I/O
- Fast Controller

- PXI Chassis
  - FPGA Board
    - IRI0 Templates
    - Custom function
- MTCA Chassis
  - AMC + FPGA
    - Custom

Application independent
Application dependent
Hardware & device drivers
Software

Desing Using HDLs
Enhancements

• Simplify the development of DAQ system using FPGAs
  • Can we avoid or minimize the use of Hardware Description Languages? Yes, we can use OpenCL as a high level synthesis language
  • Can we standardize the DAQ functionality to shift development effort only to the data processing? Yes, we can propose a model using high level description in OpenCL with solutions already given for Data Acquisition
  • Can we standardize the software to simplify the development/maintenance of applications? Yes, we can use NDSv3 modular design to separate and trace functionality changing its description in OpenCL. This functionality is even supported “in runtime” thanks to partial reconfiguration

• Let’s have a look to OpenCL
OpenCL: Overview

HLS + COMPUTING MODEL

- A **host** and multiple **devices** (CPU, GPU, FPGA)
- Computation is divided into **tasks** called **Kernels**
- There is one or several **queues** that send the **Kernels** to execute concurrently
- Memory organized in **buffer/image**. Memory model
OpenCL FPGA Device: Kernel and BSP concept

- **Kernels**: The processing pieces of OpenCL that go into the FPGA and are allocated in a specific FPGA partition to allow partial reconfiguration.
- **Kernels are written in C (OpenCL C)**.
- Kernels have access to all device memory layers.
- Parallelization is achieved replicating kernels and using loop pipeline.
- **BSP**: The part of OpenCL that goes into the FPGA and is FIXED.
  - Manages the access to hardware (DDR, PCIe, I/O).
  - Manages the Queues to the kernels.
OpenCL Host

• **Short:** The part of OPENCL that goes into the C++ drivers (HOST)
• Host sends **commands** and can read/write Global Memory (DDR) (SLOW!) and controls kernels execution, synchronization tasks, data-flow to and from device
• Critical processes can be organized with a chain of pipes (HDL=AXI ST)
• Data can be gathered from I/O, but kernels are launched with commands
Recap: NDSv3 + OpenCL

Functionality
- Health Monitoring
- Output Channels
- DAQ Channels
- Timing

HW
- IRIO - OpenCL HOST
- PCIe

SW
- NDSv3 Driver
  - Device
  - Health Monitoring
    - Temperature
    - Power
    - Current
    - Voltage
  - Waveform Generation
    - Update Rate
    - WaveType
    - Amplitude
    - Frequency
    - Value
  - Timing
    - Start Time
    - End Time
    - Mode
  - DAQ
    - Sampling Rate
    - Gain
    - Offset
  - Channel
    - Data
    - Data Out

FPGA
- IRIO-OpenCL GEN
  - Kernels
- IRIO-OpenCL DAQ
  - Kernels

ADC DAC PLL

FMC DAQ + I/O

DDR
- Management
- SPI
- JESD204B Interface

miguel.astrain@i2a2.upm.es
Results

COMPUTER CODAC Core System 6.0 + NDSv3

OpenCL 2.0 + Intel HOST

PCle optical link

MTCA4 CHASSIS

AMC ARRIA10 FPGA

Device

OpenCL 2.0

FMC DAQ Board

HOST

Everything is COTS!!!
Results

- Solution integrated into ITER CODAC Core System 6.0
- Simple kernel programming
- Access to profiling tools to tune performance.
- No large overhead due to the BSP (~15%)
- Implemented solution on a Neutron Flux Measurement use case at 2 channels @ 1GS/s
- Integrated into EPICS thanks to NDSv3
- Kernel functional validation using emulation

```c
__kernel void ADC_reader (  
    uint size,  
    __global uint8* restrict data)  
{
    for (uint i = 0; i < size; i++){
        data[i] = read_channel_Intel(rx_link);
        mem_fence(CLK_CHANNEL_MEM_FENCE);
    }
}
```
Conclusions

• Standardization of the development of FPGA-based DAQ devices using OpenCL
• Tested in a MTCA platform using an ARRIA 10 FPGA
• Reuse of FPGA development using an OpenCL hardware description of a DAQ device
• OpenCL enables C-like development of FPGA with lots of OpenCL algorithms examples
• OpenCL handles data transfers and device interface, hardware abstraction
• Combined with NDSv3 a modular solution was developed, simplifying the interface with EPICS. IRIO-OpenCL
• You only need to focus on “your specific algorithm”

“One driver to rule them all...”
Future Work

- Implementation of Machine Learning Applications
- Expand IRIO-OpenCL functionalities
- **Implementation of use cases (possible collaborations)**
  
  IRIO-OpenCL Team -> Contact us!!

UPM Big Science contact: jm.cogollor@upm.es

I2A2-UPM contact: mariano.ruiz@upm.es
Methodology to standardize the development of FPGA-based intelligent DAQ and processing systems on heterogeneous platforms using OpenCL

Thank You!

Questions?

M. Astrain¹, M. Ruiz¹, S. Esquembri¹, A. Carpeño¹, E. Barrera¹, J. Vega²

¹Instrumentation and Applied Acoustic Research Group, Universidad Politécnica de Madrid, Madrid, Spain

²Laboratorio Nacional de Fusión, CIEMAT, Madrid, Spain

miguel.astrain@i2a2.upm.es
This work was supported in part by the Spanish Ministry of Economy and Competitiveness, Projects Nº ENE2015-64914-C3-3-R and Madrid regional government (YEI fund), Grant Nº PEJD-2018-PRE/TIC-8571

The Intel® FPGA SDK for OpenCL™ is based on a published Khronos Specification.
Altera, Arria, Intel, the Intel logo, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of the Khronos Group™.
*Other names and brands may be claimed as the property of others.
Development cycle: OpenCL

Three main scenarios for a new application:

1- New algorithm

2- New algorithm + FMC module

3- New algorithm + FMC + AMC + FPGA
Neutron Flux Measurement

FPGA

HOST APPLICATION

CPU

CODAC Core System 6.0

FPGA Logic
ARRIA 10
Reconfigurable Kernels

FMC DAQ Device

JESD204B

12 bits ADC
(1 GS/s)

12 bits DAC
(1 GS/s)

Low-jitter
Clock Generator

CLKREF, SYSREF

ST Avalon Port

SPI Master

JESD204B

IRIO-OpenCL
SPI CONFIG
Kernel

IRIO-OpenCL
SPI Config

IRIO-OpenCL
DAQ
Kernel

IRIO-OpenCL
NFM

IRIO-OpenCL
GEN

Sample

Global Buffers

Pipes

Main control, Other systems, etc

ST Avalon Port

IRIO-OpenCL
Kernel

IRIO-OpenCL
Kernel

IRIO-OpenCL
DAQ
Kernel

SamplingRate

BlockSize

Kernel Arguments

12 bits ADC

1 GS/s

JESD204B

ST Avalon Port

IRIO-OpenCL
SPI Config

FIFO

Gen

IRIO-OpenCL
DAQ

IRIO-OpenCL
NFM

Results

Global Buffers

IRIO-OpenCL
GEN

Main control, Other systems, etc

ST Avalon Port

IRIO-OpenCL
Kernel

IRIO-OpenCL
Kernel

IRIO-OpenCL
DAQ
Kernel

SamplingRate

BlockSize

Kernel Arguments

12 bits ADC

1 GS/s

JESD204B

ST Avalon Port

IRIO-OpenCL
SPI Config

FIFO

Gen

IRIO-OpenCL
DAQ

IRIO-OpenCL
NFM

Results

Global Buffers

IRIO-OpenCL
GEN

Main control, Other systems, etc

ST Avalon Port

IRIO-OpenCL
Kernel

IRIO-OpenCL
Kernel

IRIO-OpenCL
DAQ
Kernel

SamplingRate

BlockSize

Kernel Arguments

12 bits ADC

1 GS/s

JESD204B

ST Avalon Port

IRIO-OpenCL
SPI Config

FIFO

Gen

IRIO-OpenCL
DAQ

IRIO-OpenCL
NFM

Results

Global Buffers

IRIO-OpenCL
GEN

Main control, Other systems, etc

ST Avalon Port

IRIO-OpenCL
Kernel

IRIO-OpenCL
Kernel

IRIO-OpenCL
DAQ
Kernel

SamplingRate

BlockSize

Kernel Arguments

12 bits ADC

1 GS/s