Real-time implementation of intelligent data processing applications: gamma/neutron discrimination and hot-spot identification

mariano.ruiz@upm.es
Universidad Politécnica de Madrid

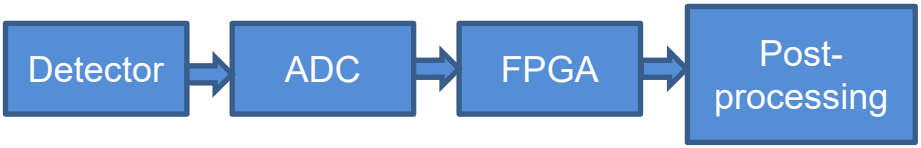
M. Ruiz[1], M. Astrain[1], S. Esquembri[1], J. Nieto[1], A. Carpeño[1], C. González[1], A. Murari[2], and A. V. Stephen[3]
[1]Universidad Politécnica de Madrid
[2]Consorzio RFX (CNR, ENEA, INFN, Universita' di Padova, Acciaierie Venete SpA)
[3] Culham Center for Fusion Energy, UKAEA

Main goal and outline

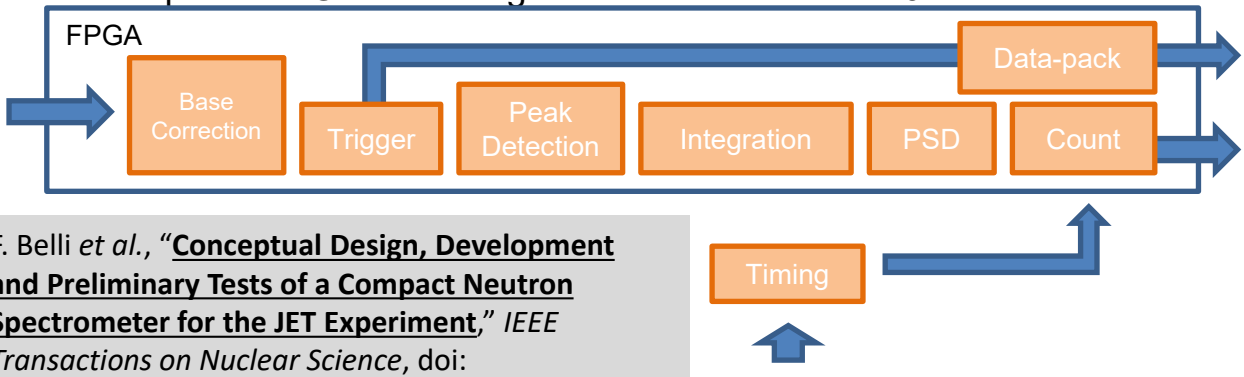
- Demonstrate the feasibility of implementing advanced data processing applications in the fusion environment using heterogeneous platforms (FPGAs and GPUs)
- Two use cases:
  - Neutron/gamma discrimination (JET)
  - Hot-spot detection, ITER Upper Visible/Infrared Wide Angle Viewing System (UWAVS)
- Show the methodology, hardware, and software elements used in the implementation
- Results and Conclusions

Context on Neutron-gamma discrimination

- Architecture of **Traditional FPGA-based DAQ systems**



- Example of FPGA Neutron gamma discrimination in JET

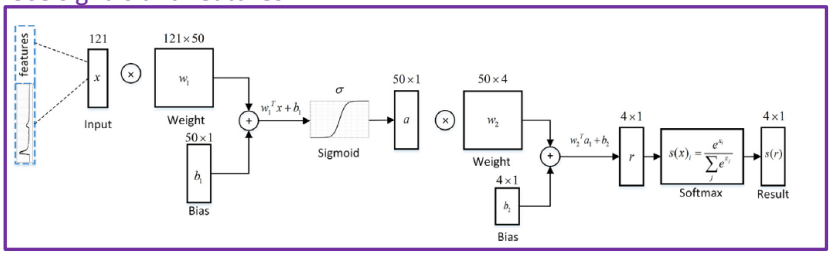


F. Belli *et al.*, "**Conceptual Design, Development and Preliminary Tests of a Compact Neutron Spectrometer for the JET Experiment**," *IEEE Transactions on Nuclear Science*, doi: 10.1109/TNS.***2012***.2208763.

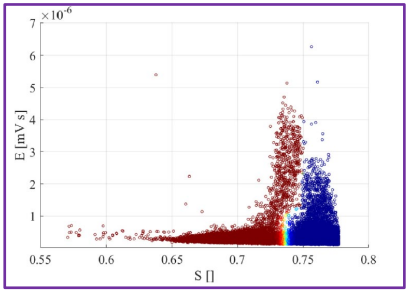
Context on Neutron-gamma discrimination

- Extensively researched topic
- Latest techniques involve Machine learning (Neural Networks, CNN, Support Vector Machine)
- Fusion datasets cannot be easily split (Complex problem, specially for low Energy)

Use signals and features



Unsupervised classifier



Classification of Gama-Neutron suing SVM
Reproduced from [Gelfusa]. JET Pulse no. 90653

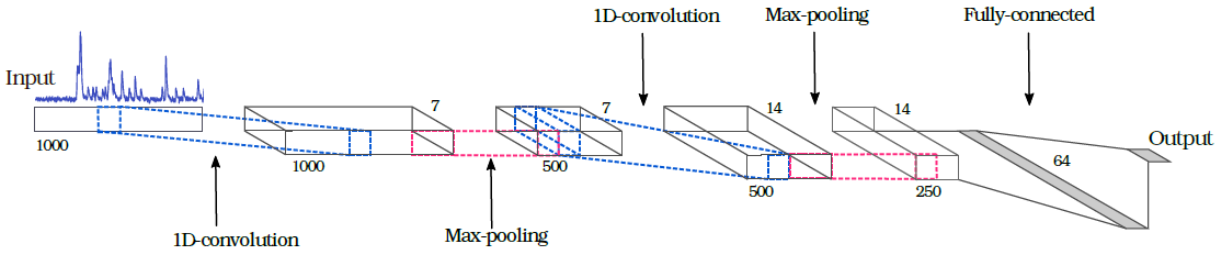
Multi Layer perceptron developed by [C. FU]

C. Fu, A. et al, "***Artificial neural network algorithms for pulse shape discrimination and recovery of piled-up pulses in organic scintillators***," *Annals of Nuclear Energy*, doi: 10.1016/j.anucene.**2018**.05.054

M. Gelfusa *et al.*, "**Advanced pulse shape discrimination via machine learning for applications in thermonuclear fusion**," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, doi: 10.1016/j.nima.**2020**.164198

Context on Neutron-gamma discrimination

- Model using CNN (common in image processing).
  - Signals are only baseline corrected
  - Output can be expanded to more than just 2 classes for more real signal classification



Reproduced from [A. Vacheret]

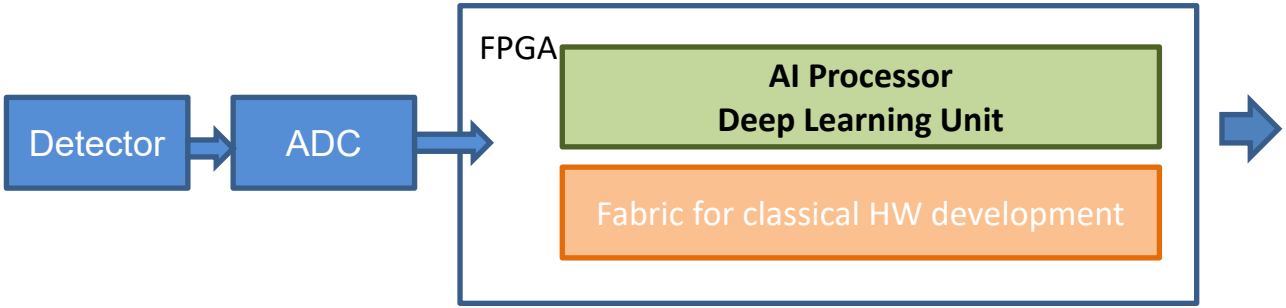
A. Vacheret, R. Taylor, D. Saunders, S. Kleinegesse, and J. Griffiths, **"Pulse Shape Discrimination and Exploration of Scintillation Signals Using Convolutional Neural Networks**," Machine Learning: Science and Technology, **2020**.

Context on Neutron-gamma discrimination

- Can we use a NN in a DAQ system?
- What is the architecture of this NN?
- What are the advantages/drawbacks?

→Classification problem
→Many choices
→Potentially better results

→Black box
→Intensive computing

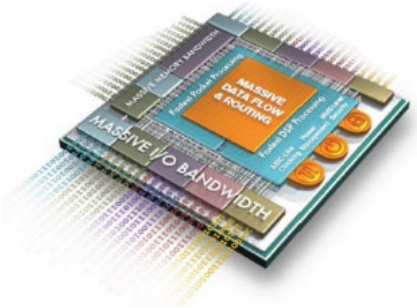


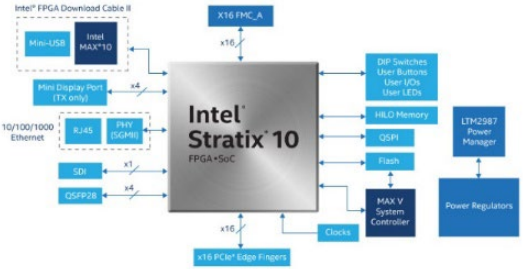
Basic tools

- New trends in FPGA technologies: promoting the use of High-level languages (HLS) or OpenCL versus Traditional FPGA or SoC (Hard to develop)
- New Heterogeneous Hardware platform: FPGA + ARM Cores **+** AI cores, ACAPs. Tensor cores, with ADCs
- Much more raw computing power than traditional approach.
- Industry interest in AI.

XILINX

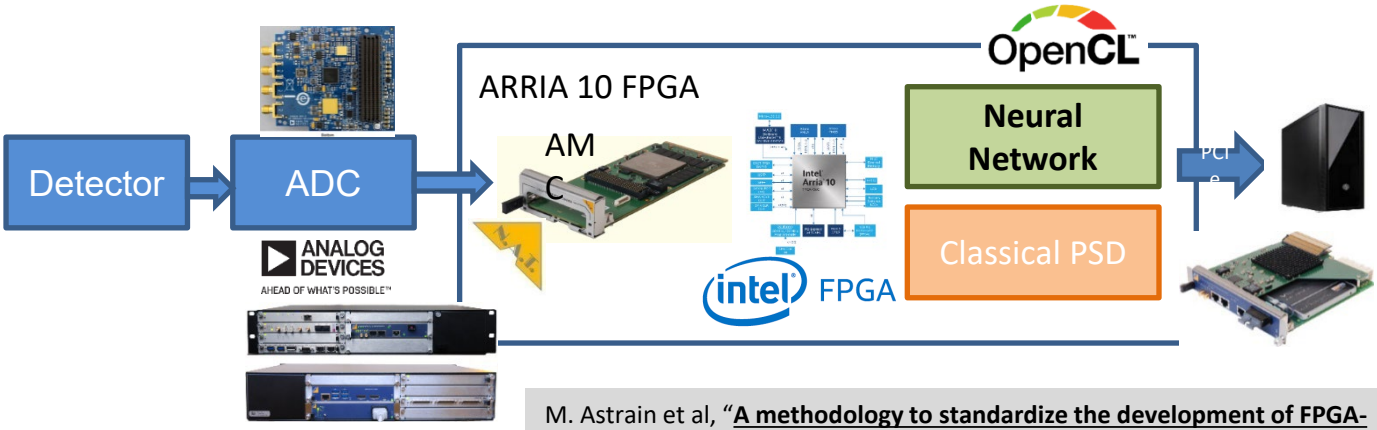
intel FPGA





Objective

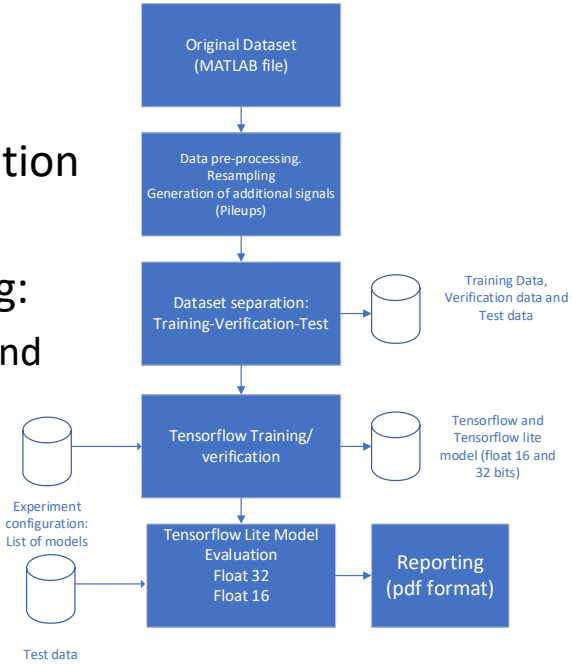
- Evaluate the implementation of a prototype using ITER technologies (MTCA and CODAC Core environment– RHEL)
  - Use of commercial AMC module and FMC ADC (NAT and Analog Devices)
  - Integration with EPICS
  - Board Support Package developed using IntelFPGA methodology
- Hardware development using OpenCL C like programming (custom CNN Neural network).



M. Astrain et al, "**A methodology to standardize the development of FPGA-based high-performance DAQ and processing systems using OpenCL**," *Fusion Engineering and Design*, doi: 10.1016/j.fusengdes.**2020**.111561

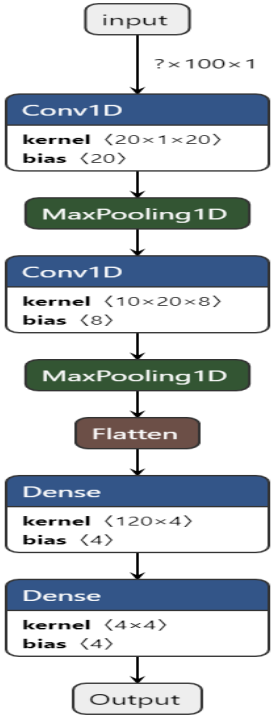
Convolutional Neural network

- Using 1D CNN
- JET Pulses re-sampled to 250MS/s
- Generate Pile-ups from pulses
- 33% split data train-verification-test
- Modeled for float 32 and 16 (Quantization did not provide useful results yet)
- Created in a python environment using:
  - Anaconda, python 3.8,tensoflow(keras) and tensorflow-lite



1D Convolutional Neural Network

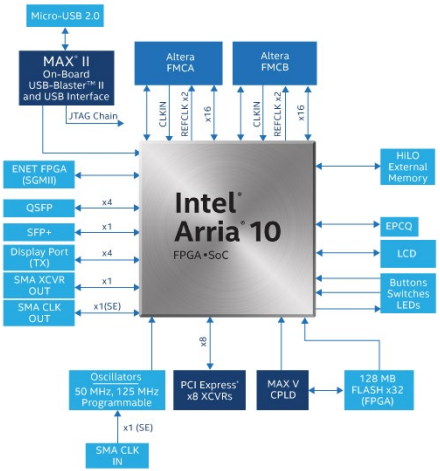
- Solution:
  - 7 Layers with 2532 weights
  - Accuracy of 99.5% in TensorFlow
  - Performance of 40 us in a desktop computer
- Advantages using CNN
  - More robust than MLP when there is a signal displacement
  - Automatic learning of features. The filters focus on different patterns, similar to the classic PSD techniques
- Disadvantage
  - Second CNN layer requires most part of computational resources
  - Data re-ordering required for calculations (implies accessing to memory resources in the FPGA. This slows the solution)



Netron Representation

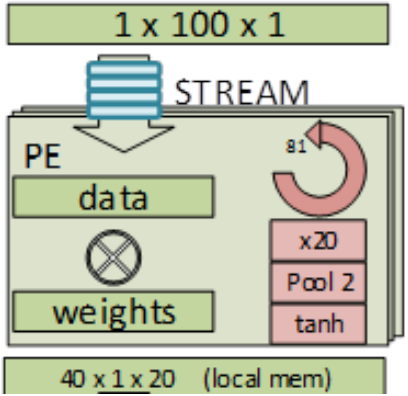
FPGA Implementation (ARRIA 10)

- Complete NN implementation using OpenCL (float32)
- Board Support Package for OpenCL 17.1
- Utilizing hardened floating-point variable-precision digital signal processing (DSP) blocks.
- 1D CNN can fit the inference parameters inside the FPGA RAM blocks (avoiding the use of external DRAM)
- External access to DRAM is minimized (global memory)
- Data streams from the FMC I/O board



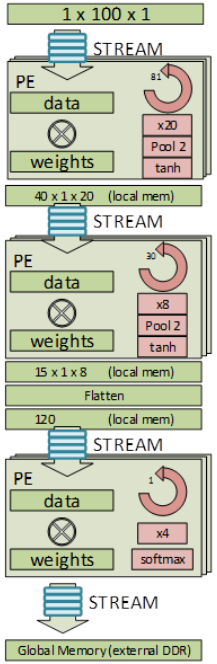
FPGA Implementation

- Processing elements in the FPGA take data streams
- ML Convolution operations:
  - Calculate the dot product and accumulate (weights)
  - Add offset (bias)
- Weights and data between layers is streamed to M20k blocks
- Activations of (tanh) neuron are costly in computing
  - But ReLu operations where unsuccessful in training
- Replication of the Processing Elements enables parallelization



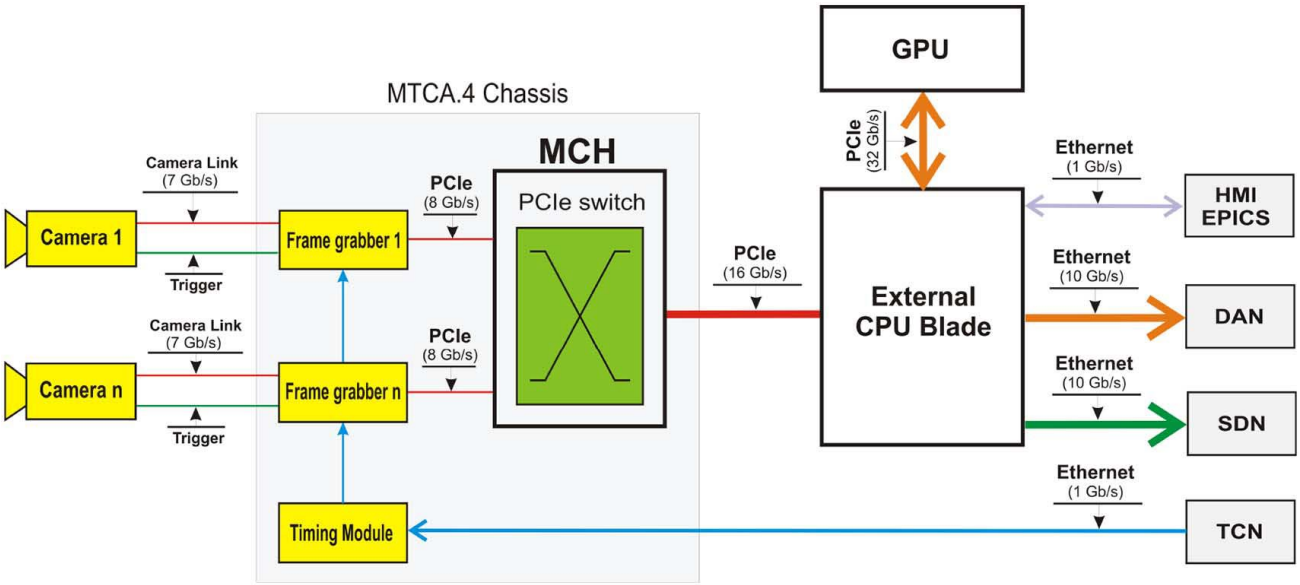
Results

- 2532 weights CNN 32bit floats.
- 150MHz kernel clock frequency.
- 40% area usage.
- Low DSP usage (higher parallelization potential!).
- 25kevent/s working with internal RAM blocks.
- Maximum error in the classification with the test signals of 0.8% (gamma), 0.9% (neutron), 2.5% (close pileup), and 1.1% (noise).
- Moving to an Intel-FPGA Stratix device could achieve around 100k events/s (this is under development).



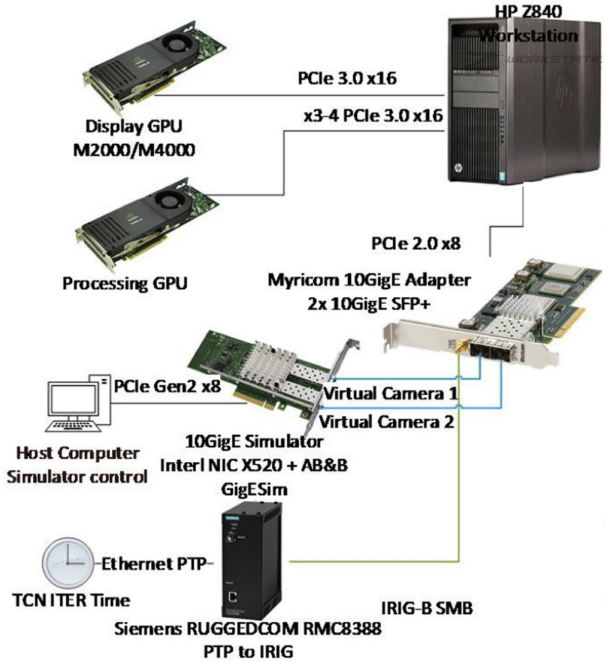
M. Astrain et al., "**Real-Time Implementation of the Neutron/Gamma Discrimination in an FPGA-Based DAQ MTCA Platform Using a Convolutional Neural Network**," *IEEE Transactions on Nuclear Science DOI: 10.1109/TNS.**2021**.3090670*

Context on ITER image acquisition systems in MTCA

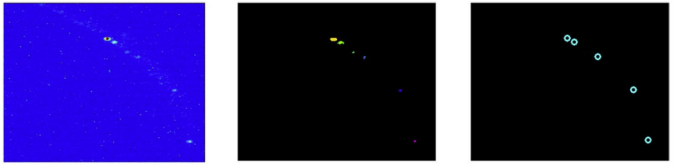


D. Makowski et al, "**High-Performance Image Acquisition and Processing System with MTCA.4**," *IEEE Transactions on Nuclear Science DOI: DOI: 10.1109/TNS.**201**5.2415582*

Context on hot-spot detection system: PCIe

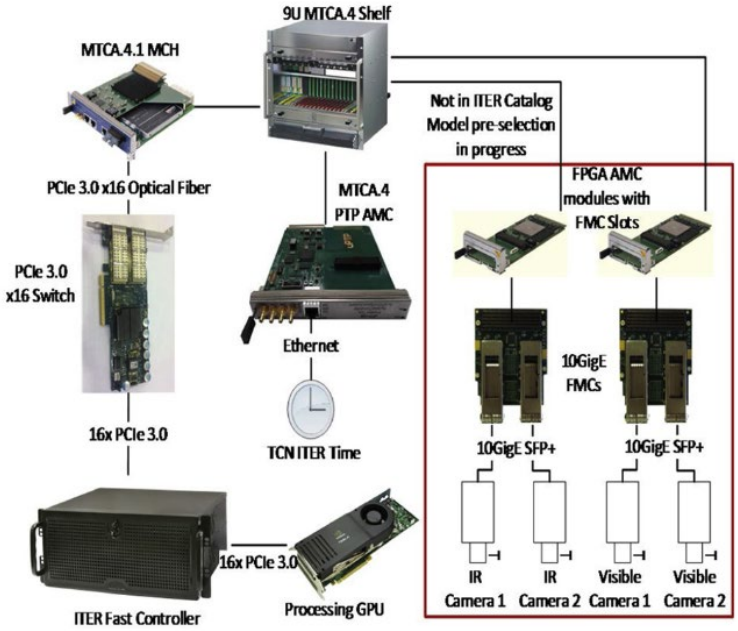


HOT SPOT DETECTION WITH raw image from JET near IR image (JPN80951, KL1-P4DA)



V. Martin et al, "**ITER upper visible/infrared wide angle viewing system: I&C design and prototyping status**," *Fusion Engineering and Design DOI: https://doi.org/10.1016/j.fusengdes.**2019**.04.015*

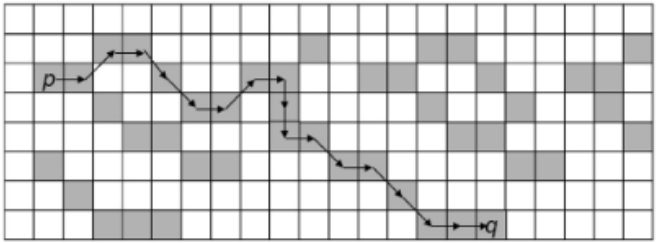
Context on hot-spot detection system: MTCA



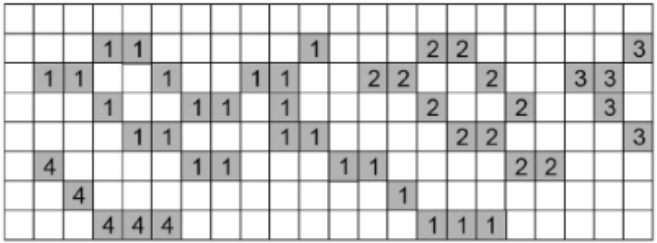
- ITER upper visible/infrared wide angle viewing system
- Preliminary design in 2018
- MTCA system
- FPGA, GPU and CPU processing
- It is based on the CCL algorithm

V. Martin et al, "**ITER upper visible/infrared wide angle viewing system: I&C design and prototyping status**," *Fusion Engineering and Design DOI: https://doi.org/10.1016/j.fusengdes.**2019**.04.015*

Context on Connected Component Labeling



(a)



(b)

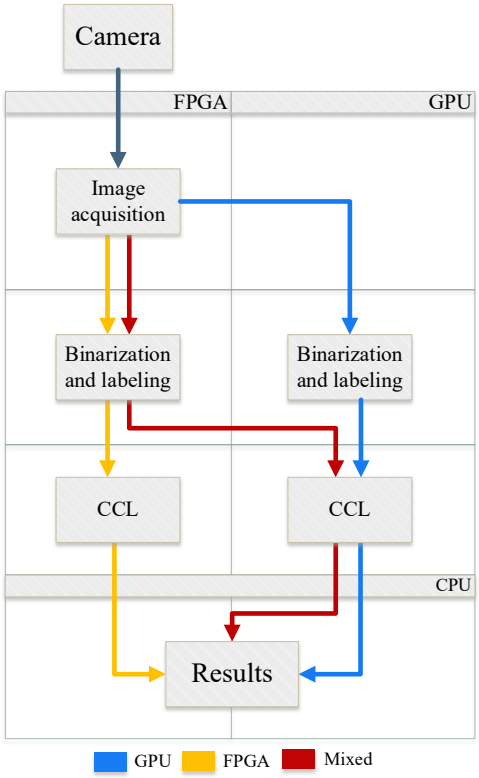
- Computationally cost
- Computation time depends on the number and distribution of foreground pixels and threshold chosen

L. He et al, "**The connected-component labeling problem: A review of state-of-the-art algorithms**"
*Pattern Recognition DOI: https://doi.org/10.1016/j.patcog.**2017**.04.018*

ITER Fast Controller: HW/SW

- Hardware
  - Industrial computer
    - MTCA chassis
    - MCH
    - AMC module with an Intel SoC (Arria 10)
    - FMC frame-grabber
  - NVIDIA and AMD GPUs
- Software:
  - ITER CODAC Core System (RHEL)
  - Nominal Device Support approach
  - Standalone C++ application and EPICS IOC

Hot-spot detection algorithm



- Monochrome detector
- Steps:
  - Detector calibration
  - Image binarization (clarify relevant pixels)
  - Pixel labeling
  - **Detection and labeling of the components using the CCL algorithm** ()
  - Component union (using centroids)
  - Blobs filtering
  - Temporal persistence
- Total algorithm performance depends on the HW chosen to execute the steps

Results: GPU performance

GPU Performance for CCL With Different Image Sizes and 20% of Foreground Pixels. AMD Radeon

Foreground Pixel Density: 20%	GPU			
	AMD Radeon RX 5700			
Nº. pixels	Original Implementation		Modified Implementation	
	Cycles/ Px	Px/s	Cycles/ Px	Px/s
256x256	16.63	9.71E+07	17.81	9.07E+07
512x512	4.86	3.33E+08	5.16	3.13E+08
1024x1024	2.06	7.84E+08	2.00	8.08E+08
2048x2048	0.86	1.87E+09	0.71	2.28E+09
4096x4096	0.70	2.31E+09	0.55	2.94E+09

GPU Performance for CCL With 1024 × 1024 Image Size and a Variable Number of Foreground Pixels. NVIDIA GeForce

Images size: 1024*1024	GPU			
	GeForce RTX 2080 SUPER			
Foreground Pixel Density (%)	Original Implementation		Modified Implementation	
	Cycles/ Px	Px/s	Cycles/ Px	Px/s
10	0.60	2.75E+09	0.37	4.41E+09
20	0.73	2.27E+09	0.50	3.32E+09
30	0.78	2.11E+09	0.56	2.95E+09
40	0.88	1.87E+09	0.68	2.43E+09
50	1.01	1.64E+09	0.79	2.08E+09
60	1.20	1.38E+09	1.05	1.57E+09
70	1.07	1.54E+09	0.98	1.68E+09
80	1.09	1.51E+09	1.07	1.55E+09
90	1.15	1.44E+09	1.00	1.65E+09

GPU Performance for CCL With Different Image Sizes and 20% of Foreground Pixels. NVIDIA GeForce

Foreground Pixel Density: 20%	GPU			
	GeForce RTX 2080 SUPER			
Nº. pixels	Original Implementation		Modified Implementation	
	Cycles/ Px	Px/s	Cycles/ Px	Px/s
256x256	3.11	5.30E+08	4.30	3.84E+08
512x512	0.95	1.73E+09	1.31	1.26E+09
1024x1024	0.73	2.27E+09	0.50	3.32E+09
2048x2048	0.43	3.88E+09	0.24	7.00E+09
4096x4096	0.35	4.67E+09	0.18	9.34E+09

GPU Performance for CCL With 512 × 512 Image Size and a Variable Number of Foreground Pixels. NVIDIA GeForce

Images size: 512*512	GPU			
	GeForce RTX 2080 SUPER			
Foreground Pixel Density (%)	Original Implementation		Modified Implementation	
	Cycles/ Px	Px/s	Cycles/ Px	Px/s
10	0.91	1.82E+09	1.11	1.49E+09
20	0.95	1.73E+09	1.31	1.26E+09
30	1.22	1.35E+09	1.43	1.15E+09
40	1.33	1.24E+09	1.60	1.03E+09
50	1.61	1.03E+09	1.83	9.02E+08
60	1.97	8.39E+08	2.20	7.51E+08
70	1.77	9.32E+08	2.01	8.22E+08
80	1.91	8.63E+08	2.29	7.21E+08
90	1.87	8.82E+08	2.22	7.44E+08

Results: FPGA performance

ARRIA 10 PERFORMANCE FOR THE CCL WITH 512 × 512 IMAGE SIZE AND A VARIABLE NUMBER OF FOREGROUND PIXELS

Images size: 512*512	FPGA	
	Intel FPGA Arria 10 SoC	
Foreground Pixel Density (%)	Cycles/ Px	Px/s
10	9.40	1.35E+07
20	16.13	7.88E+06
30	22.94	5.54E+06
40	29.70	4.28E+06
50	36.74	3.46E+06

ARRIA10 INTELFPGA, RESOURCES USED FOR THE IMPLEMENTATION

Resource Usage				
Intel FPGA Arria 10 SoC				
	ALUTs	FFs	RAMs	DSPs
Kernel Subtotal	7004	10209	1069	0
Channel Resources	60	484	2	0
Global Interconnect	1344	2870	18	0
Board Interface	66800	133600	182	0
Total	75208 (17%)	147230 (17%)	1273 (65%)	0 (0%)

- Arria 10 FPGA limitations:
  - Internal memory to store frames
  - Space to replicate the HW elements

Comparison GPU vs FPGA

PROCESSING TIMES PER HARDWARE DEVICE AND ALGORITHM STEP WITH 512 × 512 IMAGE SIZE AND A VARIABLE NUMBER OF FOREGROUND PIXELS

Images size: 512*512 Modified Implementation

Foreground pixel density (%)	AMD (Base clock: 1610 MHz)		NVIDIA (Base clock: 1650 MHz)		FPGA (Kernel clock: 127 MHz)	
	Bin. (us)	CCL (us)	Bin. (us)	CCL (us)	Bin. (ns)	CCL (ms)
10	143.5	597.4	32.5	143.7	7.9	18.0
20	181.3	654.9	33.6	173.2	7.9	31.9
30	213.7	811.1	33.5	194.3	7.9	45.7
40	244.2	869.9	33.6	220.0	7.9	59.6
50	278.8	1070.5	33.5	257.2	7.9	73.5

Conclusions

- OpenCL used for coding in GPUs and FPGAs
  - simplifies the development
  - Requires some tuning to achieve the performance in FPGAs
  - Generally, GPUs provide better performance but buffer movement to GPUs has latency
  - FPGAs embedded with the data acquisition hardware
- Use of advanced state-of-the-art Machine Learning methods to implement the n/g discrimination problem using JET Pulses
  - 1D CNNs achieve high accuracy in the classification.
  - 1D implementations are simple enough to fit in the FPGA using OpenCL. The whole NN can be implemented in an FPGA.
  - 25kevents/s but higher performance FPGAs might be able to achieve a real-time classification system with around 100kevent/s
- Use of advanced images processing in heterogeneous systems
  - Frame-grabber is connected to an FPGA that implements binarization and labeling
  - CCL requires 20ms in an ARRIA 10 device
    - Moving to Stratix 10 improves kernel clock and provides more internal memory (estimated CCL time around 5ms)
    - Implementing multiple CCL compute units (5) in Stratix 10 will provide a total execution time below 1ms.

Acknowledgement

- Funding: Grants PID2019-108377RB-C33 and PID2019-108377RB-C31 funded by MCIN/AEI/10.130- 39/501100011033; Comunidad de Madrid grant number PEJ-2019-AI/TIC-14507.
- Euratom Research and Training Programme (Grant Agreement No. 101052200—EUROfusion).