ML Applications In TID-AIR

Ryan Herbst & Gabriel Blaj
TID-AIR Electronics Systems & Detectors

ML-at-SLAC 1st Workshop 2/18/2019

(rherbst@slac.stanford.edu)

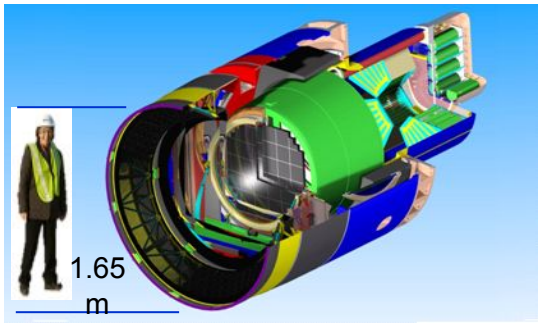
SLAC TID-AIR
Technology Innovation Directorate
Advanced Instrumentation for Research Division

U.S. DEPARTMENT OF ENERGY
Office of Science

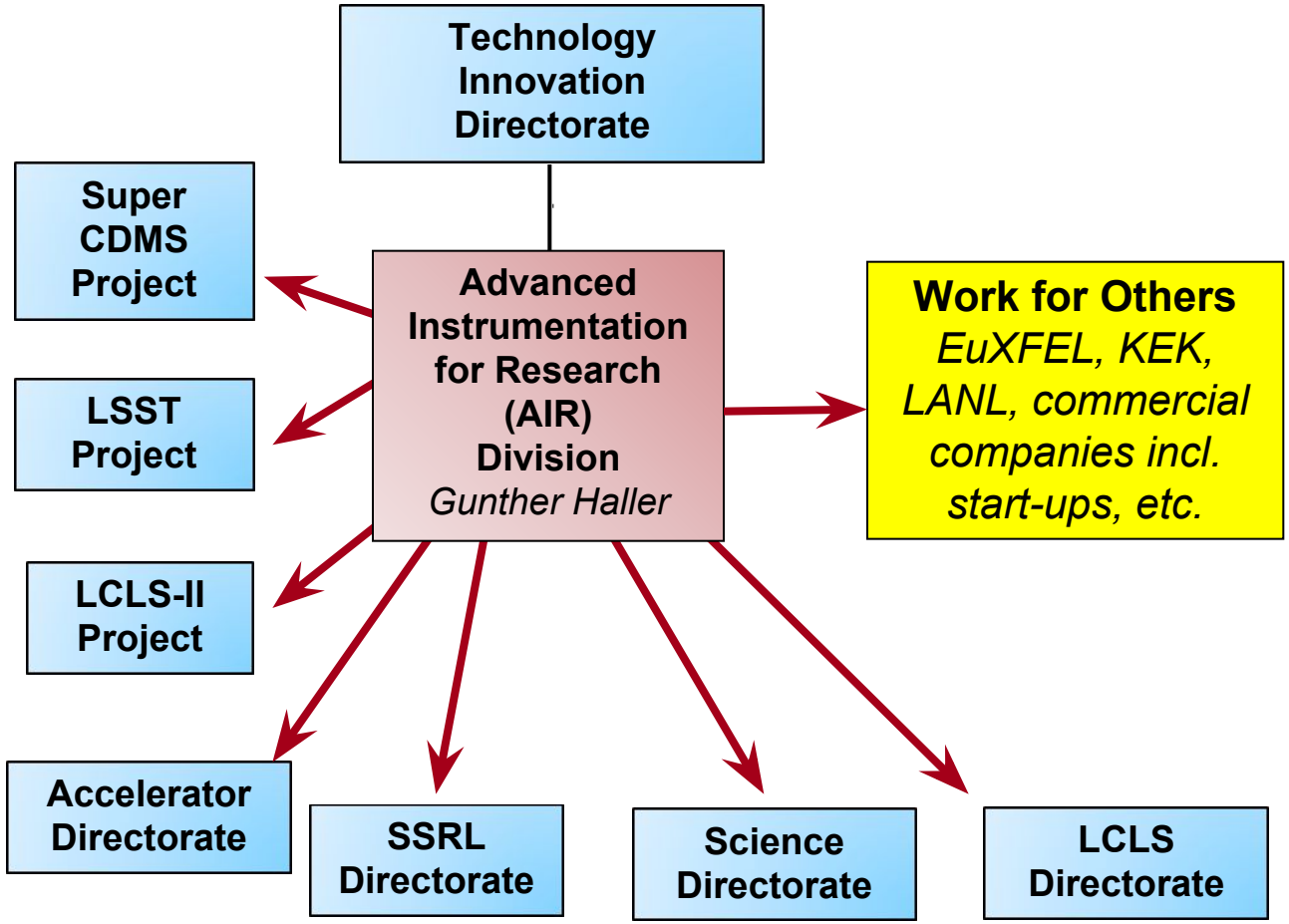
SLAC NATIONAL ACCELERATOR LABORATORY

TID-AIR provides Systems & Components for SLAC & National / International Projects

- Provides engineering for SLAC directorates
- All *SLAC developed* LCLS/LCLS-II detectors
- HEP: ATLAS, CDMS, HPS, etc.
- LSST, CMB, Fermi …
- **Non-SLAC Projects**



LSST: World's largest digital camera



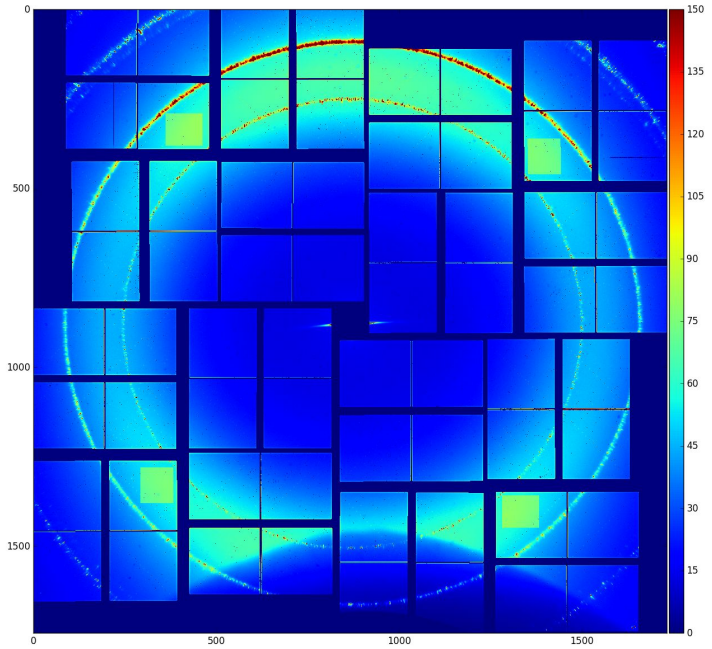
- Strong team of 8 engineers with at least 10 years of experience in FPGA & firmware design
- Core firmware library and build system utilized by numerous laboratories

LCLS-II: Data Reduction for 2D Detectors at 100kHz

SLAC

- A typical LCLS camera at 120 Hz:
  - ~500 MB/s raw data rate (~petabytes/year)
    - 2.2 Mpixel, 2 Bytes/pixel

- LCLS-II: towards 100 kHz:
  - Accelerator and detector development underway
  - ~500 GB/s data rate
    - Current DAQ limitations:
      - Data rate
      - Storage

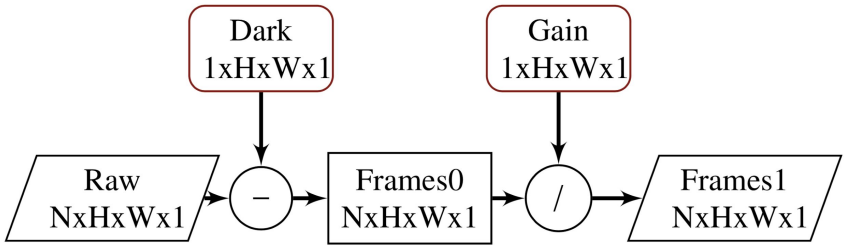


https://confluence.slac.stanford.edu/display/PSDMInternal/2017-03-21+Background+subtraction

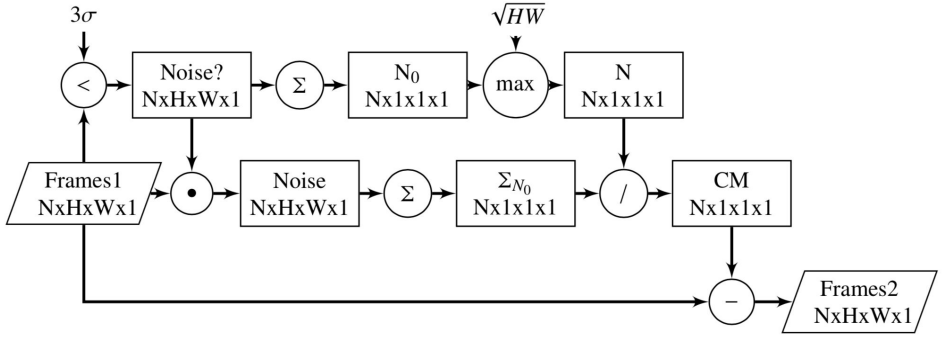
- Online data reduction:
  - Detector corrections in real time
  - Extract sparse photons (retaining full photon information)
    - 1-3 orders of magnitude better compression than raw images
  - Save successful events

Dark, Gain and Common Mode Correction in TF

- Dark and Gain Correction:



- Common Mode Correction:

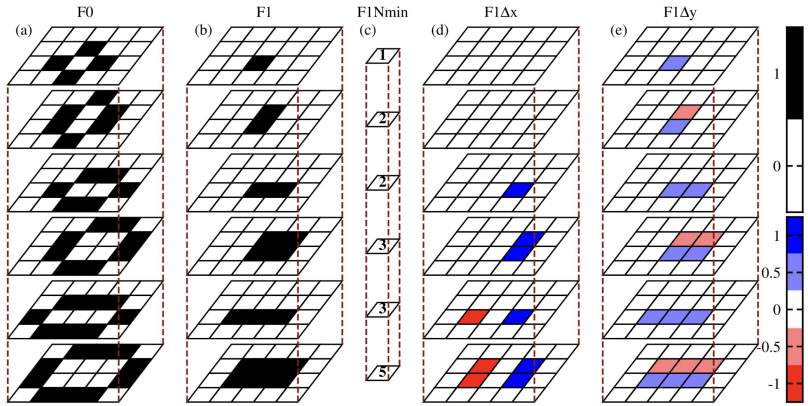


- Straightforward
  - Easily extended to next generation auto gain-switching detectors at LCLS (not shown)
- Relatively complex
  - Robust at relatively high photon occupancy
    - ~1% pixels with 0 photons

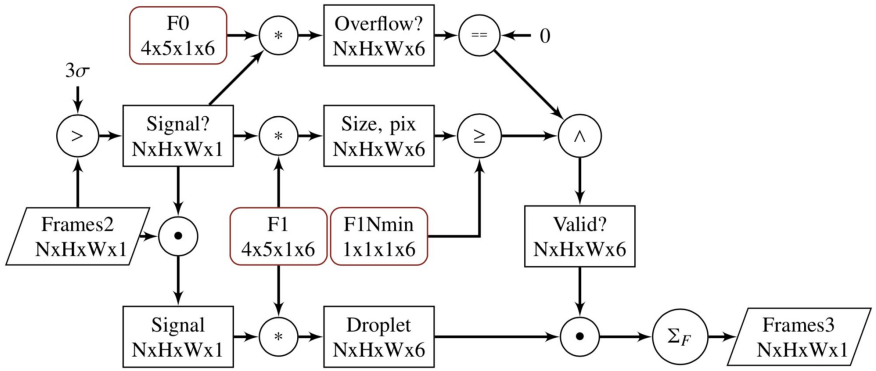
Blaj et al., *Ultrafast Processing of Pixel Detector Data with Machine Learning Frameworks*, SRI Proceedings, 2018. accepted

“Droplet” Photon Charge Reconstruction in TF

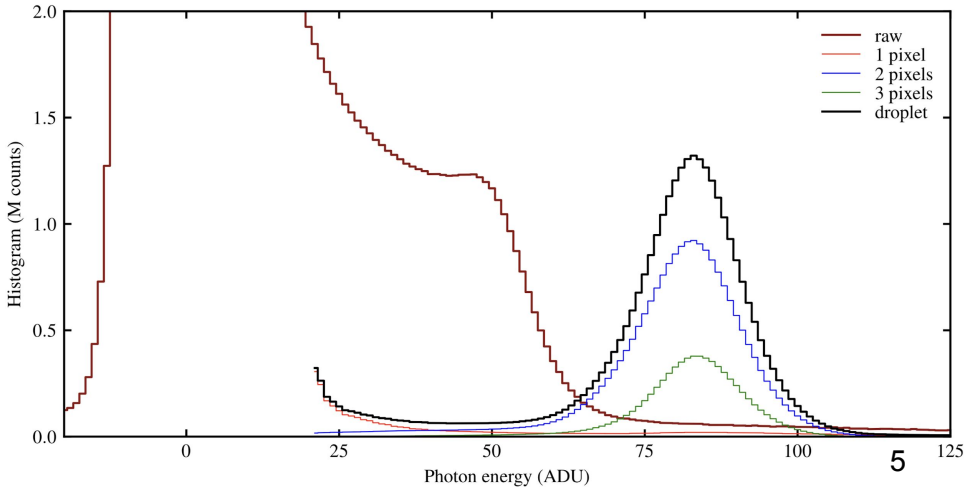
- Weights



- Photon Charge Summing
  - 8 lines of TF code



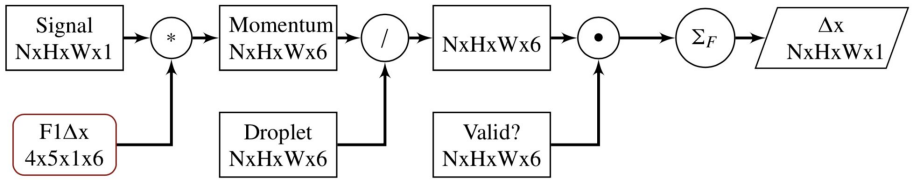
- Using CNNs to calculate total charge for sparse photons and allocate to appropriate pixel
- Several discrete weights
  - easy quantization and increased FPGA speed



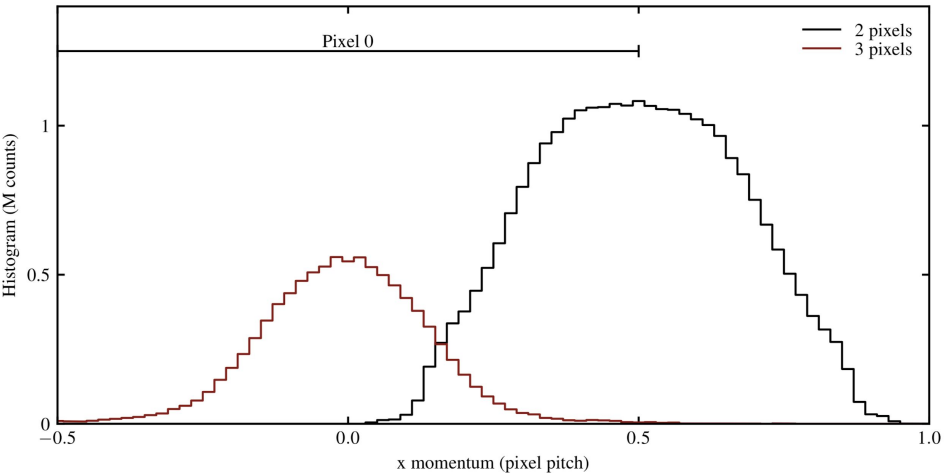
Blaj et al., *Ultrafast Processing of Pixel Detector Data with Machine Learning Frameworks*, SRI Proceedings, 2018. accepted

Sparsification and Subpixel Resolution in TF

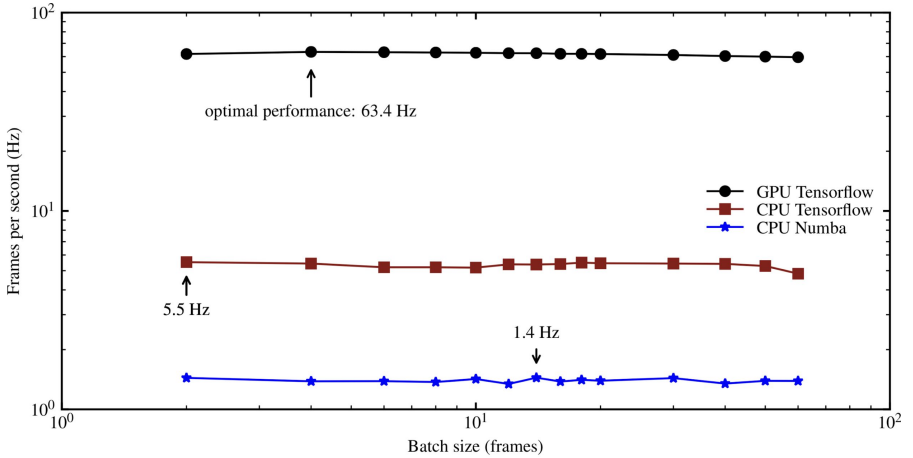
- Sparsification:
  - straightforward (dedicated tf function)
- Subpixel resolution:
  - Model:



  - ~ 5 μm subpixel accuracy for ePix100



- Speed:



- $300 consumer GPU:
  - ~45x faster than compiled parallel code on 4 CPUs
- Tensorflow:
  - Ultracompact
    - Entire data processing pipeline: ~50 lines of code

Blaj et al., *Ultrafast Processing of Pixel Detector Data with Machine Learning Frameworks*, SRI Proceedings, 2018. accepted

Machine Learning in TID\AIR

SLAC

- **Published**: Machine Learning (ML) approach for 2D detector data streams
  - 100x cost reduction compared with Psana and LCLS-II Data Reduction Pipeline
  - G. Blaj, C. Chang, C. Kenney, *Ultrafast processing of pixel detector data with machine learning frameworks*, AIP Conf. Proc. **2054**, 060077 (2019) https://doi.org/10.1063/1.5084708
- **Proposed**: LDRD to develop a distributed FPGA system to deploy this ML model (and any other model) for real time, full speed 100kHz LCLS-II detector data streams;
  - 1,000x cost reduction projected for real time analysis and storage of every event in LCLS-II at full speed
  - Based on existing, widely deployed FPGA Reconfigurable Cluster Element architecture developed at SLAC TID\AIR https://www.slac.stanford.edu/pubs/slacpubs/16000/slac-pub-16182.pdf
  - G. Blaj, R. Herbst, J. Thayer, C. Chang
  - NOT funded in FY 2019
- **Currently** training new generations of ML models for:
  - Increased accuracy compared to standard Psana and the state of the art algorithms
  - Easy deployment in future distributed FPGA systems
- **Involved** with Silicon Valley community working on FPGA ML compilers
  - FPGA companies and users are developing single FPGA coprocessor frameworks (insufficient for demanding, real time, high throughput applications)
  - No one is currently developing distributed FPGA systems and are highly interested (e.g., Xilinx, personal communication)

Deploying ML To Hardware
Common Approaches

SLAC

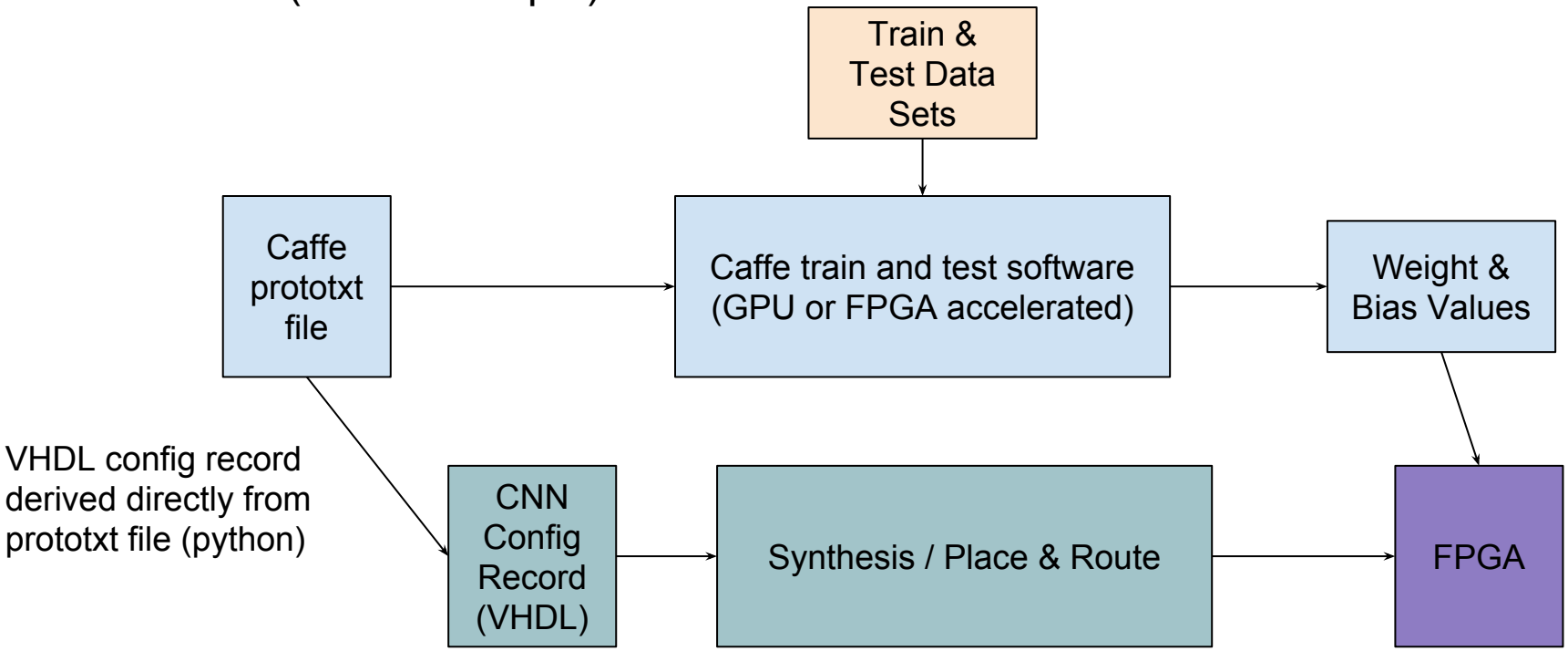
- Four categories of approaches to ML in FPGAs *
  - Single processing engine
    - Systolic array, processing each layer sequentially
      - Homogeneous network of tightly coupled data processing units each processing a portion of the network data
    - Software based processing with FPGA coprocessor
  - Streaming architecture
    - One processing engine per network layer
    - Synchronous dataflow (SDF) model for mapping CNNs to FPGAs
    - Networks can be deployed across multiple FPGAs
  - Vector processor
    - Instructions specific to accelerating the operations of convolutions
    - Software driven processing with FPGA coprocessor
  - Neurosynaptic processor
    - Map digital neurons and their interconnecting weights
    - ASIC based processing engines with configurable routing

* arXiv:1612.07119 [cs.CV]: “FINN: A Framework for Fast, Scalable Binarized Neural Network Inference”

Proposed Framework For Deploying Machine Learning In DAQ & Trigger Systems

SLAC

- Firmware framework for compiling ML networks into one or more FPGAs
  - Clean interface between layers allows for inter-FPGA serialization over point to point or network based protocols
- VHDL based as opposed to Vivado HLS
  - Current experience with Vivado HLS has exposed weaknesses
- Working design flow for deploying neural networks in FPGA auto generated from Caffe (as an example) model:



Proposed VHDL Based Framework
Proof Of Concept (2017)

SLAC

- VHDL record driven generation of CNN synthesized into a pipelined classification engine]
- Deployed in TID-AIRs Firmware Library SURF
  - https://github.com/slaclab/surf
- LeNet (single written decimal recognition) Example:

```
constant CNN_LENET_C : CnnLayerConfigArray(5 downto 0) := (

  0 => genCnnConvLayer (strideX   => 1,  strideY   => 1,
                        kernSizeX => 5,  kernSizeY => 5,
                        filterCnt => 20,
                        padX      => 0,  padY      => 0,
                        chanCnt   => 10, rectEn    => false),

  1 => genCnnPoolLayer (strideX   => 2, strideY   => 2, kernSizeX => 2, kernSizeY => 2),

  2 => genCnnConvLayer (strideX   => 1,  strideY   => 1,
                        kernSizeX => 5,  kernSizeY => 5,
                        filterCnt => 50,
                        padX      => 0,  padY      => 0,
                        chanCnt   => 50, rectEn    => false),

  3 => genCnnPoolLayer (strideX   => 2, strideY   => 2, kernSizeX => 2, kernSizeY => 2),

  4 => genCnnFullLayer ( numOutputs => 500, chanCnt => 50, rectEn => true ),

  5 => genCnnFullLayer ( numOutputs => 10, chanCnt => 1, rectEn => false ));
```

Existing Frameworks Under Investigation

SLAC

- The following frameworks are being studied to see if they can be used or serve as a guide for a SLAC framework

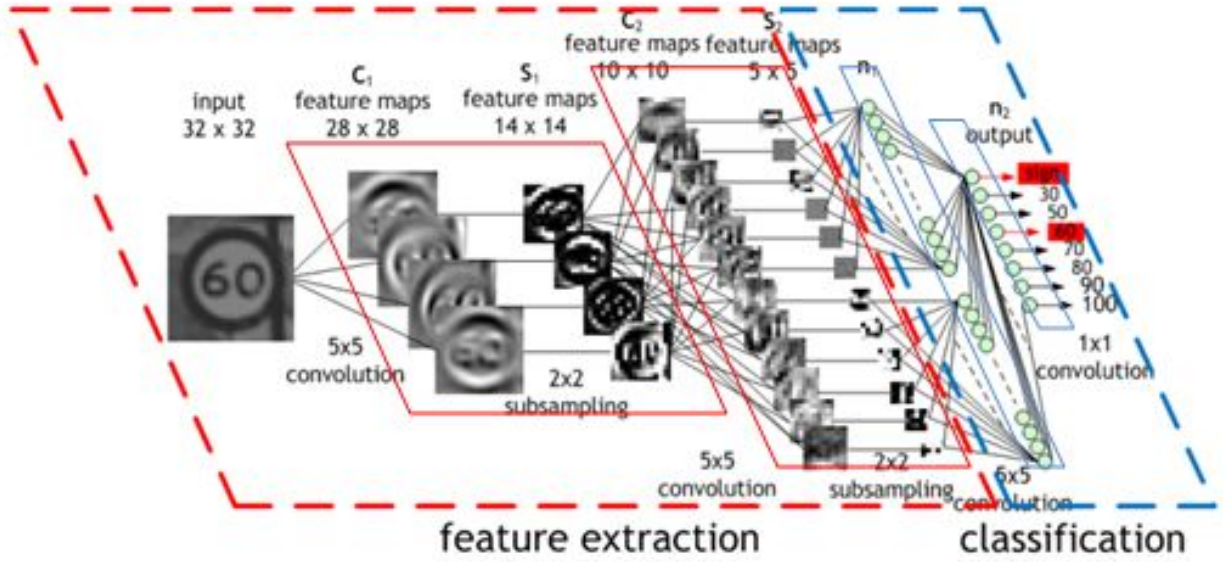
- Xilinx: **FINN**
  - Open Source
  - Xilinx Vivado HLS
  - Binarized Neural Networks (BNNs)
  - https://github.com/Xilinx/FINN

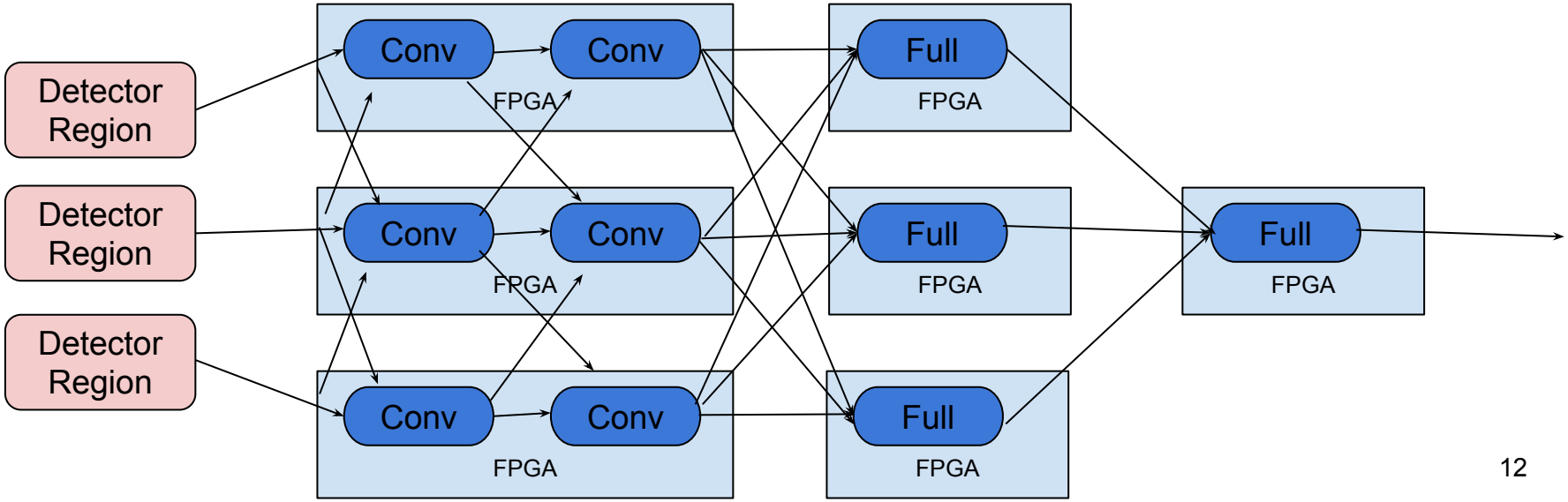
- CERN, Columbia, Fermilan, MIT, UI, etc: **hls4ml**
  - Open Source
  - Xilinx Vivado HLS
  - Demonstrated streaming inference with small networks
  - https://hls-fpga-machine-learning.github.io/hls4ml/

- Imperial College London: **fpgaConvNet**
  - Source Not Available
  - Xilinx Vivado HLS
  - http://cas.ee.ic.ac.uk/people/sv1310/fpgaConvNet.html

Deploying Machine Learning In Hardware
Matching Deployment To Common DAQ Structures

SLAC

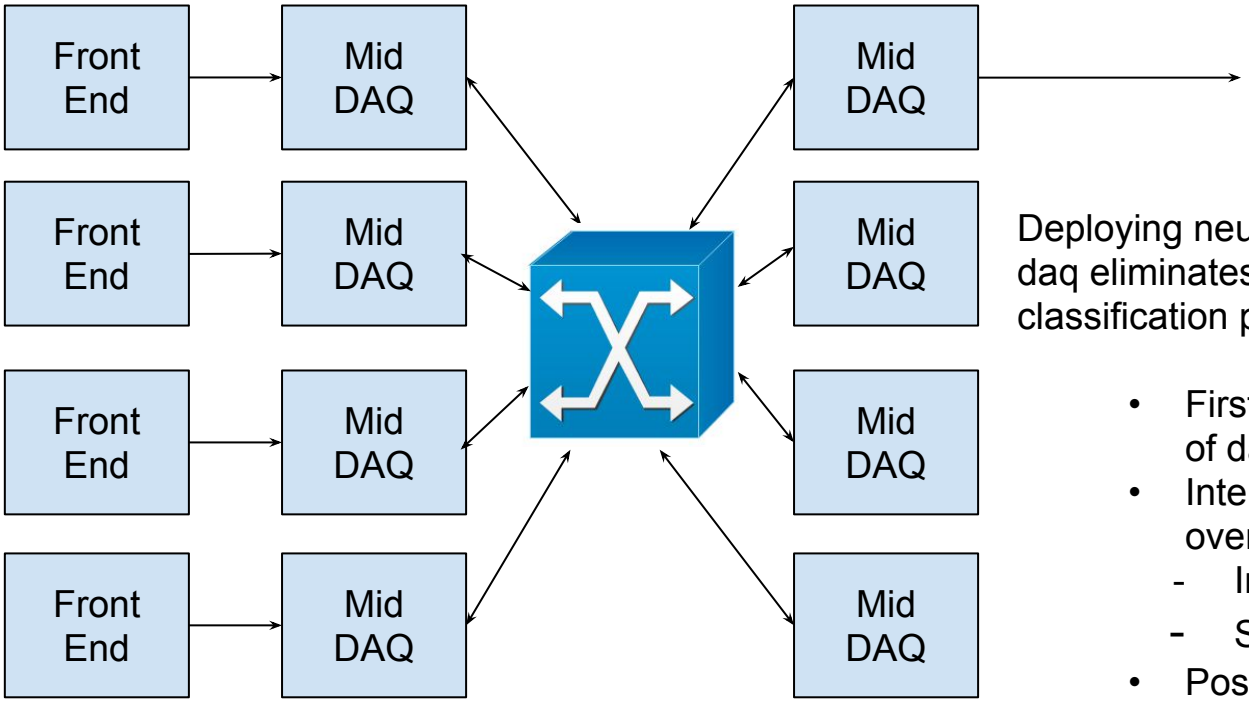




Deploying Machine Learning In Hardware
Distributing The Layers Without Event Building

SLAC

- Simple to deploy FPGA firmware for classification
  - Each layer is pipelined, allowing higher frame rate
  - Layers are flexible, can exist in different FPGAs
    - Take advantage of 8-bit quantization for DSP density

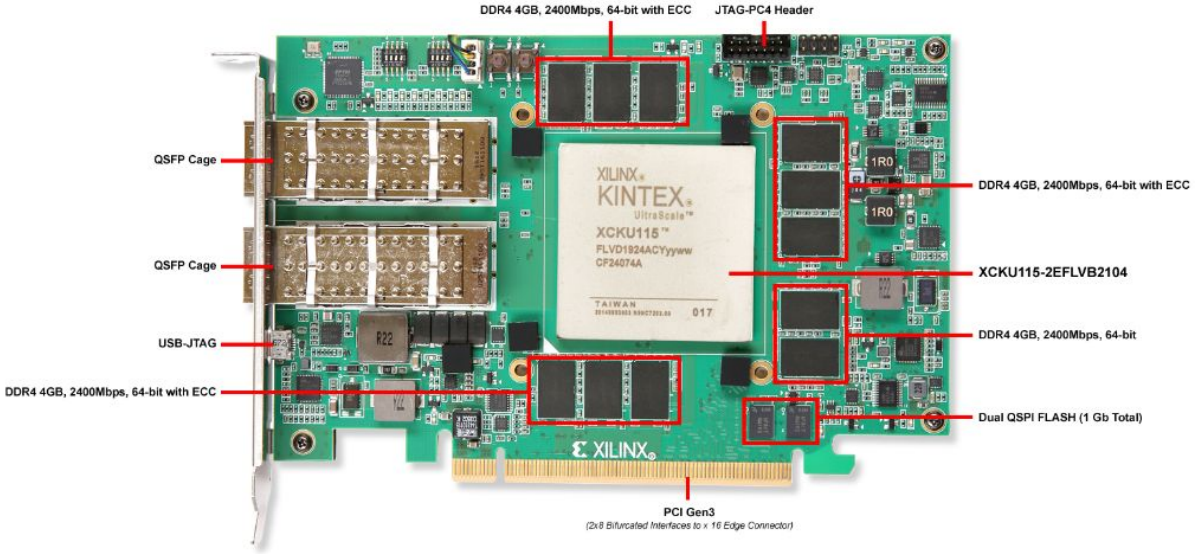


Deploying neural network structures in mid-level daq eliminates need to event build for classification purposes

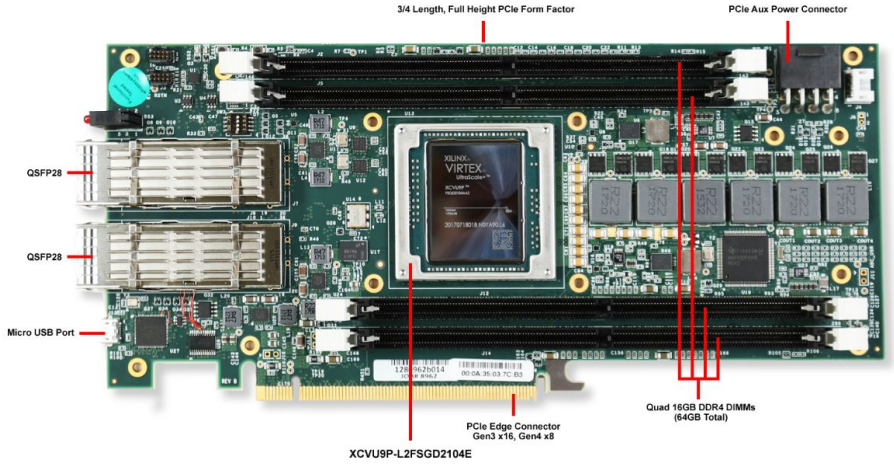
- First few layers have minimal overlap of data (convolution)
- Inter-FPGA connections transport overlap data
  - Inter-firmware RUDP
  - Shared hybrid memory cube (HMC)
- Possible to design networks which minimize or eliminate overlaps

Commercial FPGA Hardware Examples

Xilinx KCU1500 co-processor
- XCKU115 FPGA
- 2 QSFP optical modules
- 16GB DDR
- Amazon AWS



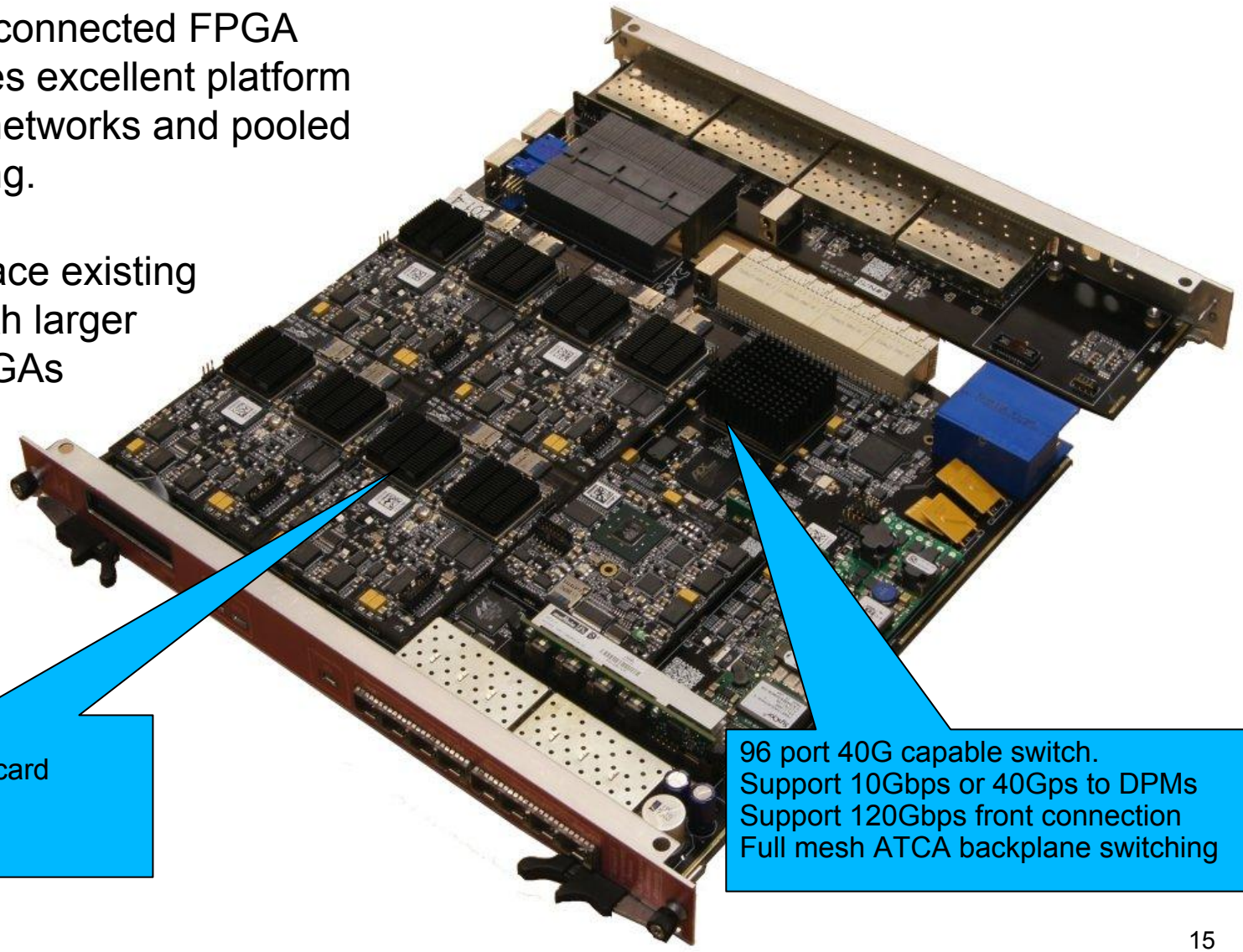
Xilinx Virtex UltraScale+ VCU1525
- XCVU9P FPGA
- 2 QSFP optical modules
- 64GB DDR



Custom Hardware
SLAC RCE Platform

Networked interconnected FPGA modules provides excellent platform for both neural networks and pooled FPGA processing.

Possible to replace existing Zynq FPGAs with larger Ultrascale + FPGAs

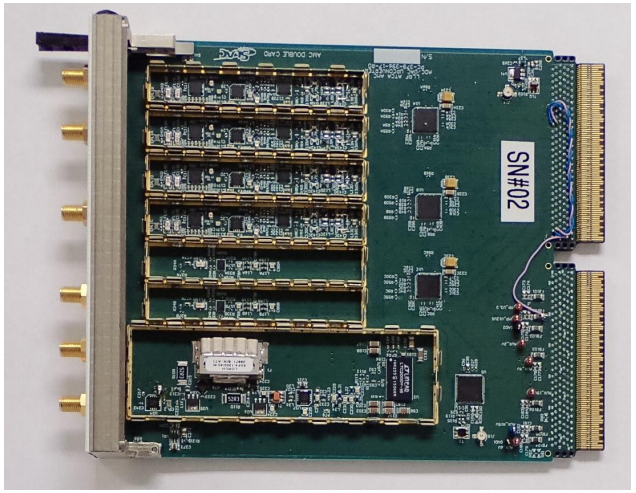


8 Zynq7045 FPGAs
2 per DPM daughter card
1 Zynq 7030 FPGA
DTM daughter card

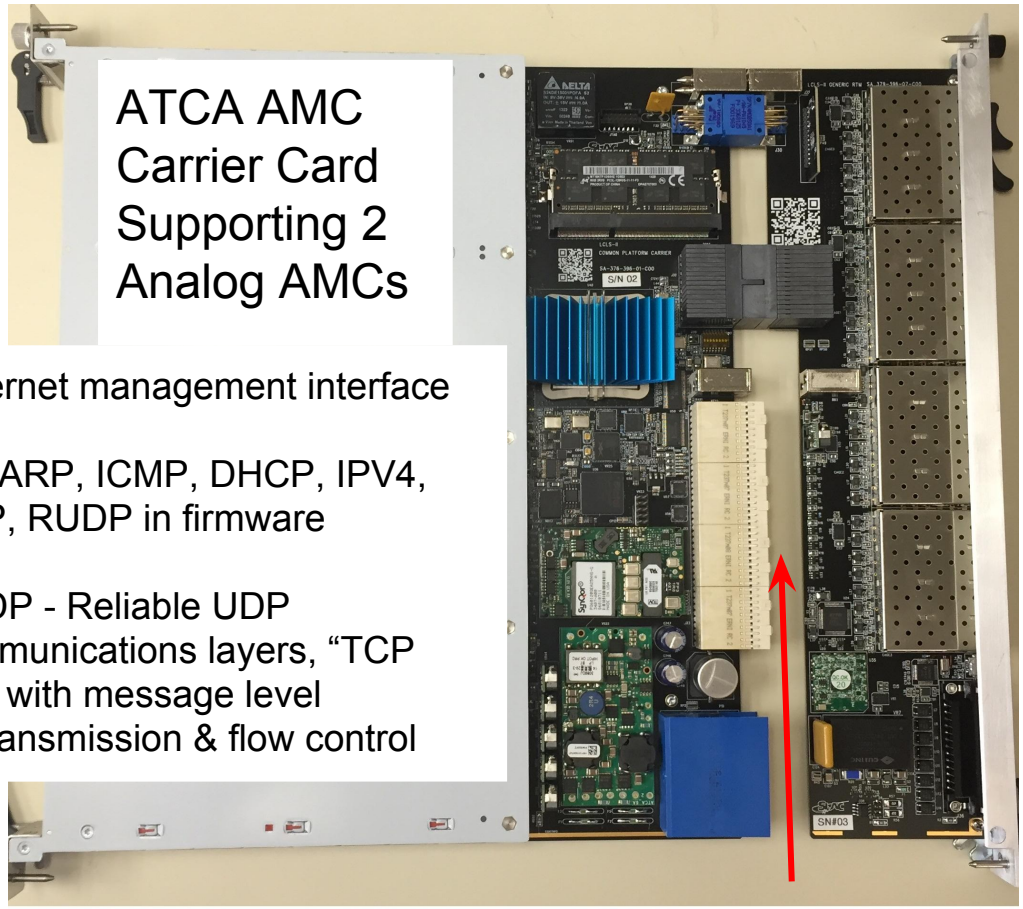
96 port 40G capable switch.
Support 10Gbps or 40Gps to DPMs
Support 120Gbps front connection
Full mesh ATCA backplane switching

Custom Hardware
SLAC AMC Common Platform

LCLS-1 LLRF Down Convert



ATCA provides the space, power & cooling required for LCLS-2!



- Ethernet management interface
- Full ARP, ICMP, DHCP, IPV4, UDP, RUDP in firmware
- RUDP - Reliable UDP communications layers, “TCP like” with message level re-transmission & flow control

Large FPGA coupled with analog application cards

Basis for LCLS2 High Performance Systems (HPS) Controls

Base platform for SMURF TES sensor RF readout

10/40Gbps Ethernet Backplane

The End