Acceleration of Scientific Deep Learning Models
On Heterogeneous Computing Platform

Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

University of Pittsburgh
BYU BRIGHAM YOUNG UNIVERSITY
Virginia Tech
UF UNIVERSITY of FLORIDA

Chao Jiang, David Ojika, Herman Lam:
*SHREC @ UF**

Sofia Vallecorsa:
CERN openlab

Thorsten Kurth, Prabhat:
*NERSC Berkeley Lab***

Bhavesh Patel:
Dell EMC

* SHREC: NSF Center for Space, High-Performance, and Resilient Computing, University of Florida
** NERSC: National Energy Research Scientific Computing Center, Lawrence Berkeley National Lab

Outline

- Hetergeneous computing for deep learning
  - Data pre-processing; model training; model inference
  - Focus on FPGA-acceleration of inference stage
- Experimental platforms & tools
  - Intel PAC10 card; OpenVINO; DLA* design suite
- Case studies
  - HepCNN & CosmoGAN:
    - Review of ISC19 IXPUG workshop results
    - Improved results *(new)*
  - 3DGAN *(new)*:
    - Initial results
- Conclusions & going forward

Heterogeneous Computing[1] for Deep Learning

Motivation

- Deep learning becoming *pervasive* for mission-critical computing
  - *Heterogeneous computing** offers unique capabilities to accelerate DNNs[2]

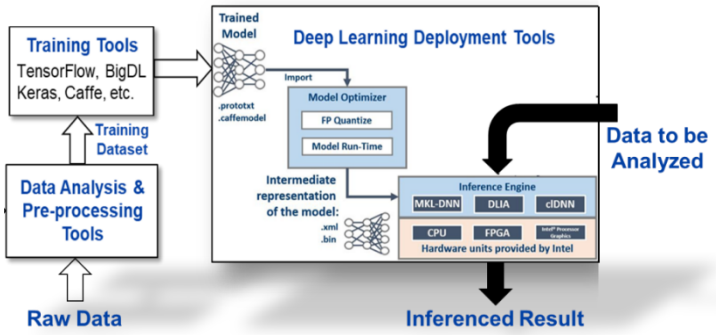
Goal

Perform design-space exploration:

- Of *emerging HGC[1] archs/tools* and *DNN models*
- For *acceleration* of selected mission-critical apps

Approach

Focus on use of *FPGAs* to *accelerate inference stage* of the HGC workflow



Stages of HGC workflow

- Data analysis & pre-processing
- Model training
- DNN inference

Collaborating partners

- ***NERSC**:*** HepCNN, CosmoGAN model support
- ***CERN openlab:*** 3D GAN model support
- ***Dell:*** SHREC membership support, equipment
- ***Intel:*** Deep-learning tools; engineering support

DNN Models *from NERSC & CERN Openlab*

- HEP-CNN
- CosmoGAN
- 3D GAN

*HGC: Heterogeneous Computing (CPU+GPU+FPGA)

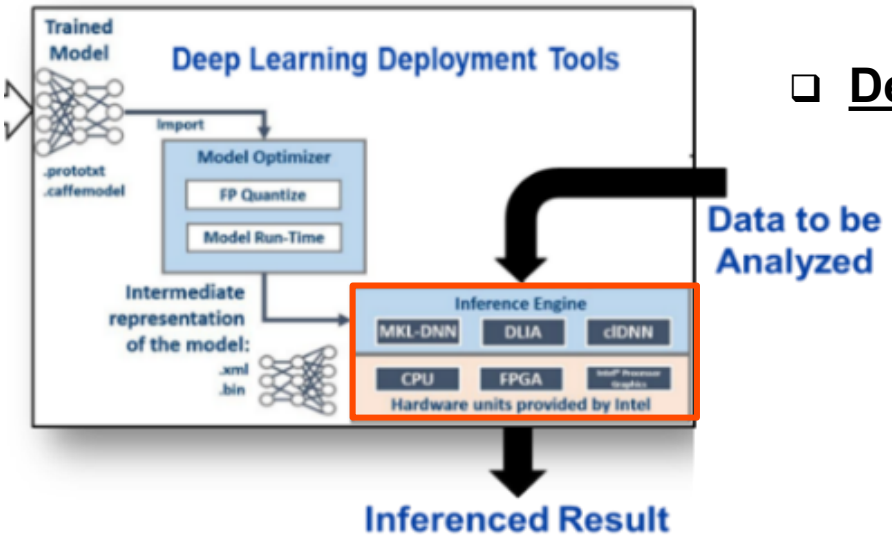
** NERSC: National Energy Research Scientific Computing Center, Lawrence Berkeley National Lab

FPGA Acceleration for DNN Inference

Experimental Setup & Tools

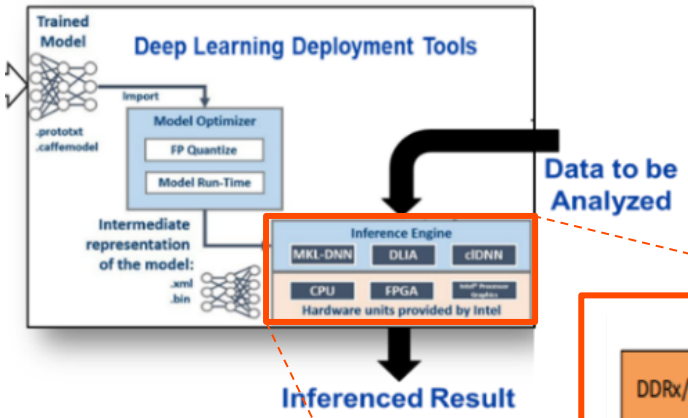
- **Experimental platforms**
  - **Dell EMC server:** 2x Intel Xeon Gold 6130 CPU
  - **Intel PAC:** Arria 10 GX FPGA

- **Intel OpenVINO Toolkit**
  - Model Optimizer
    - Convert mainstream deep learning framework model *(TensorFlow, Caffe, etc.)* into unified *intermediate representations (IR)*
  - Inference Engine
    - API library for mapping IR onto *Intel hardware platforms (CPU, GPU, FPGA, etc)*
    - Integrated with *Deep Learning Accelerator suite* for *FPGA acceleration*



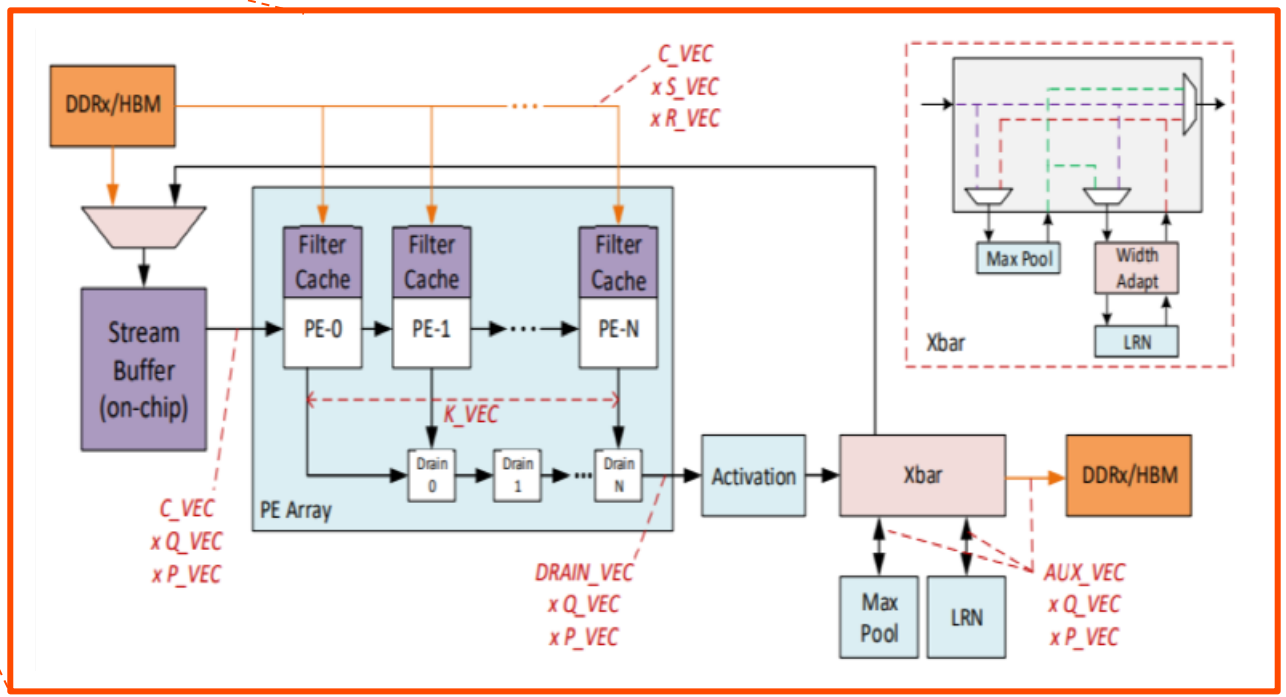
- **Deep Learning Accelerator suite (DLA)**
  - *OpenCL-based* implementation of DNN inferencing hardware architecture
  - Source code acquired through NDA with Intel to be *optimized for various applications*

Deep Learning Accelerator Suite (DLA [1])



- *OpenCL-based* implementation of DNN inferencing hardware architecture
- Source code acquired through NDA with Intel to be *optimized for various applications*

- DDR/HBM
- Stream Buffer
- PEs: processing elements
- Activation module
- Xbar
- Max Pool module
- LRN: normalization

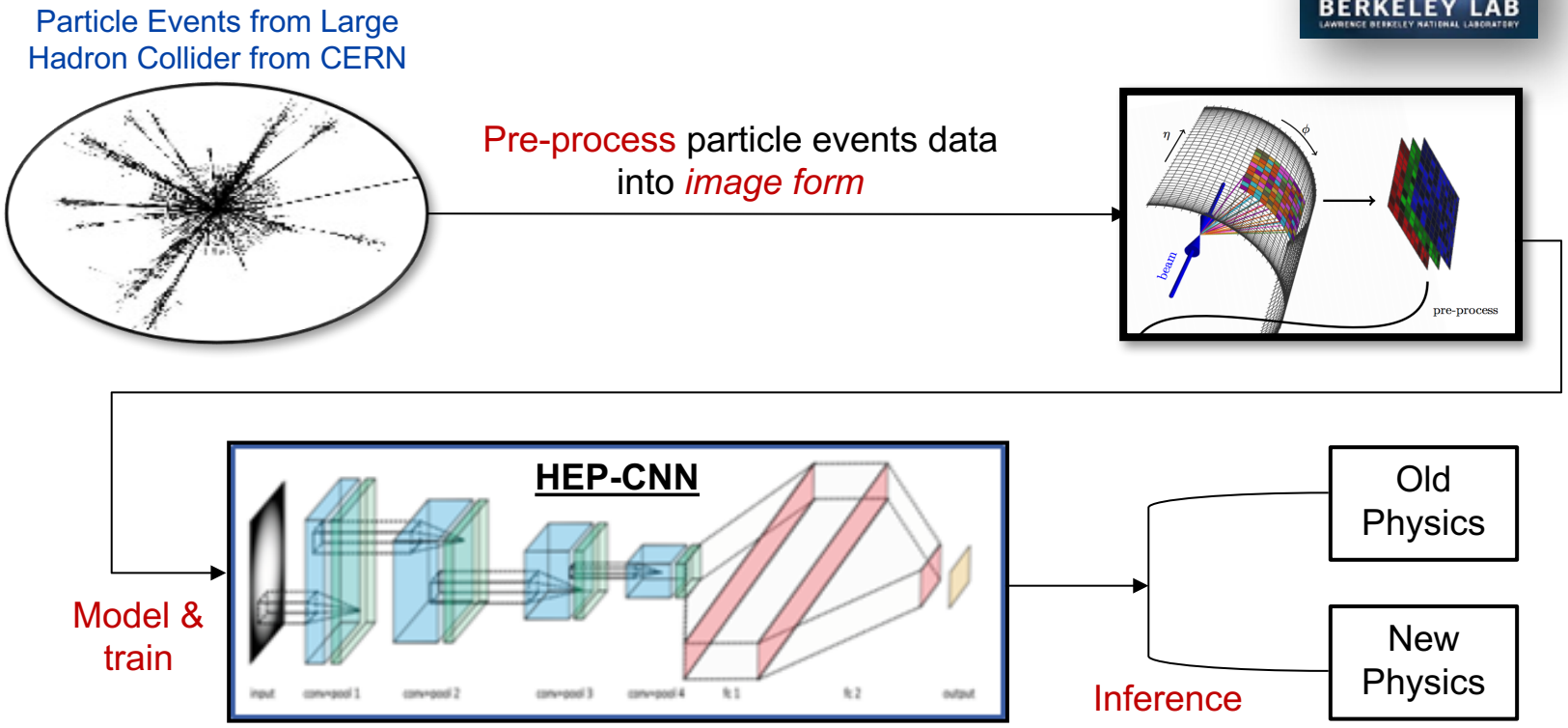


[1] Abdelfattah, M.S. et al., DLA: Compiler and FPGA Overlay for Neural Network Inference Acceleration. arXiv e-prints arXiv:1807.06434 (Jul 2018)

Outline

- Hetergeneous computing for deep learning
  - Data pre-processing; model training; model inference
  - Focus on FPGA-acceleration of inference stage
- Experimental platforms & tools
  - Intel PAC10 card; OpenVINO; DLA* design suite
- Case studies
  - HepCNN & CosmoGAN:
    - Review of ISC19 IXPUG workshop results
    - Improved results *(new)*
  - 3DGAN *(new)*:
    - Initial results
- Conclusions & going forward

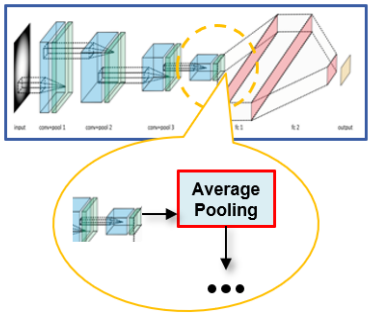
Case Study: HEP-CNN Model [2] from



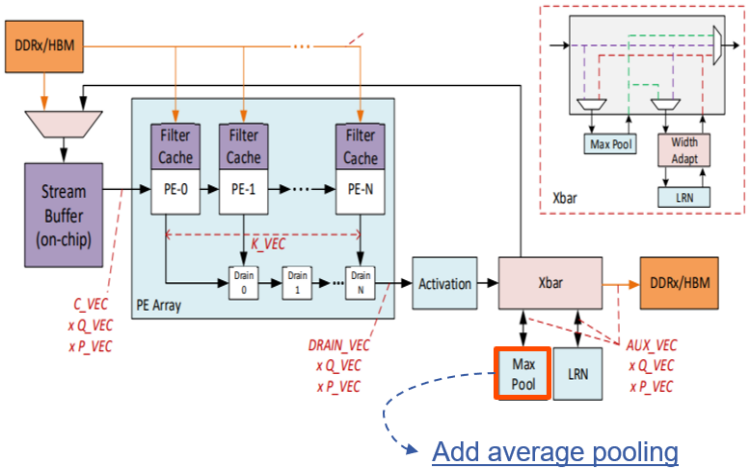
- Classifies *particle events* between
  - "ones which *can be fully described by standard model physics*"
  - "ones which *contain new physics*".
- Developed and trained by *NERSC at Lawrence Berkeley Lab* using CNN (Convolutional Neural Network) topology

[2] T. Kurth et al. 2017. Deep Learning at 15PF: Supervised and Semi-supervised Classification for Scientific Data. High Performance Computing, Networking, Storage and Analysis (SC '17). 7:1–7:11.

HEP-CNN: ISC19 IXPUG Workshop Results



HEP-CNN model structure

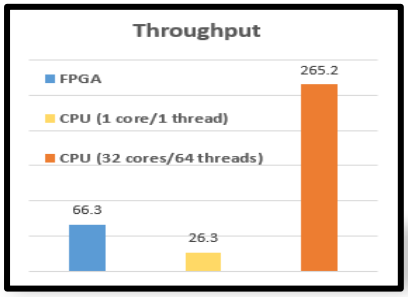


Problem:

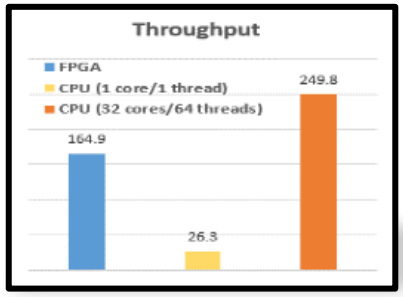
- Native DLA architecture does not have FPGA primitive to process *Average Pooling* layer

Solution:

- Customize DLA source code to add Average Pooling FPGA primitive in Max Pool module



Initial results (2.52X)



Improved results (6.27X)

- All processing performed in FPGA
- Improved from *2.52x* to *6.27x* vs 1-core/1-thread Intel Xeon CPU

HEP-CNN: Further Improved Results (w/ DLA 2019)

FPGA vs. CPU performance *DLA 2018 vs 2019* Implementation

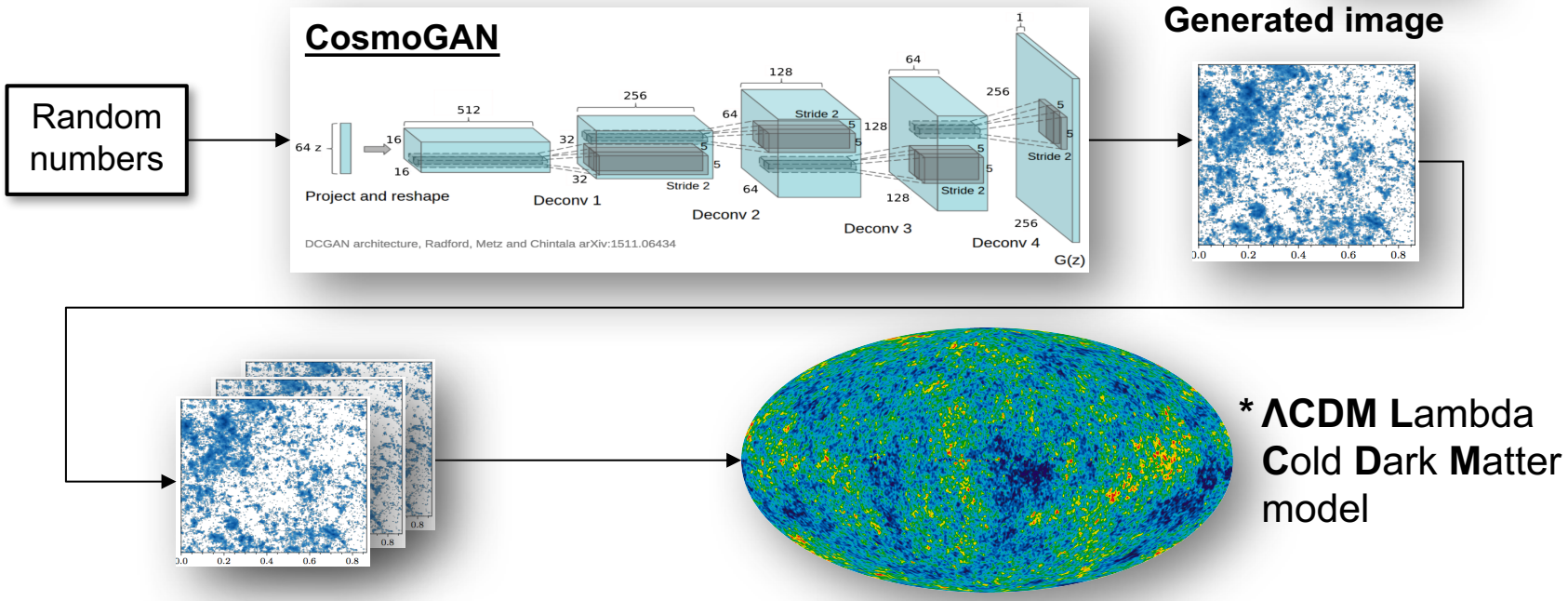
	Bit precision	FPGA * Throughput	CPU (1 core/1 thread) ** Throughput	CPU (32 core/64 thread) ** Throughput	FPGA speedup vs. 1 core CPU
DLA 2018	FP16	**164.9 imgs/sec**	**26.3 imgs/sec**	**265 imgs/sec**	**6.27x**
DLA 2019	FP16	***252.1 imgs/sec***	**26.3 imgs/sec**	**265 imgs/sec**	***9.58x***

* Arria 10 at 20 nm process
** Intel Xeon Gold 6130 CPU at 14 nm process

Observation:

- FPGA inference performance for *HEP-CNN* improved ***~53%*** with *DLA 2019*
  - DLA 2019 *discards Winograd transformation* for convolution layer to save computation resource for *more data parallelism*

Case Study: CosmoGAN[3] Model from



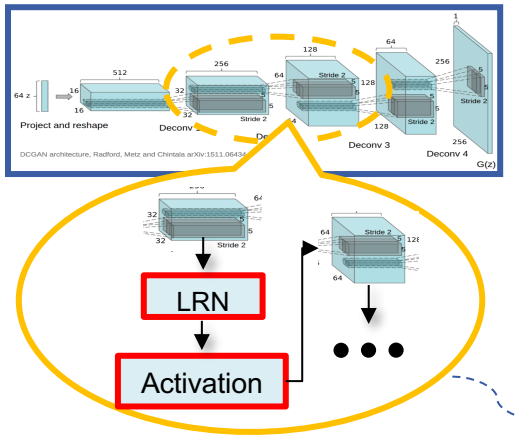
* **ΛCDM** Lambda **C**old **D**ark **M**atter model

- Generates ΛCDM* weak lensing convergence maps for cosmological studies
  - Each output image is a *measure of the density of the universe* observed from a particular direction
- Developed and trained by NERSC at Lawrence Berkeley Lab using *DCGAN (Deep Convolutional Generative Adversarial Network)* topology
- **Objective:** study how to improve FPGA acceleration for complex scientific DNNs

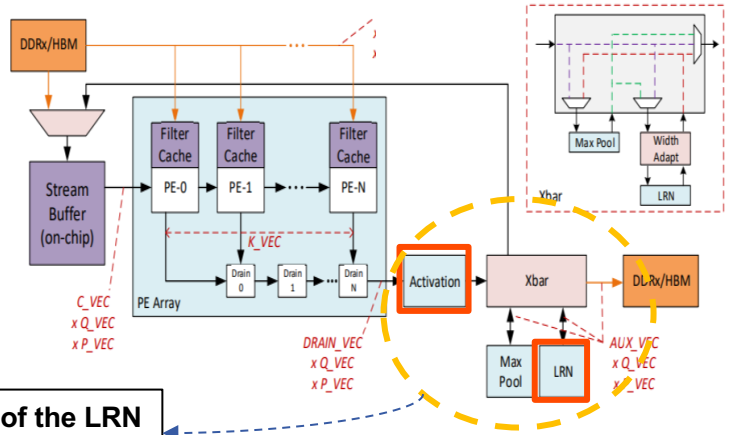
[3] M. Mustafa, D. Bard, W. Bhimji, R. Al-Rfou, and Z. Lukić,.Creating Virtual Universes Using Generative Adversarial Networks. 2017

CosmoGAN: ISC19 IXPUG Workshop Results

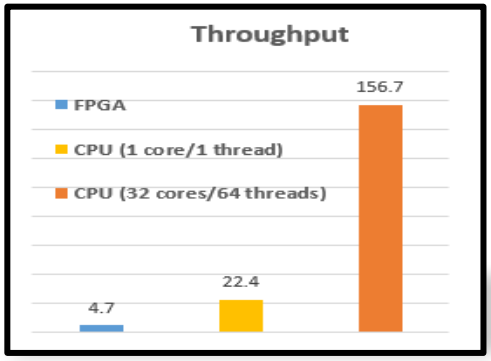
CosmoGAN model architecture



DLA system-level diagram



Execution order of the LRN and Activation is reversed



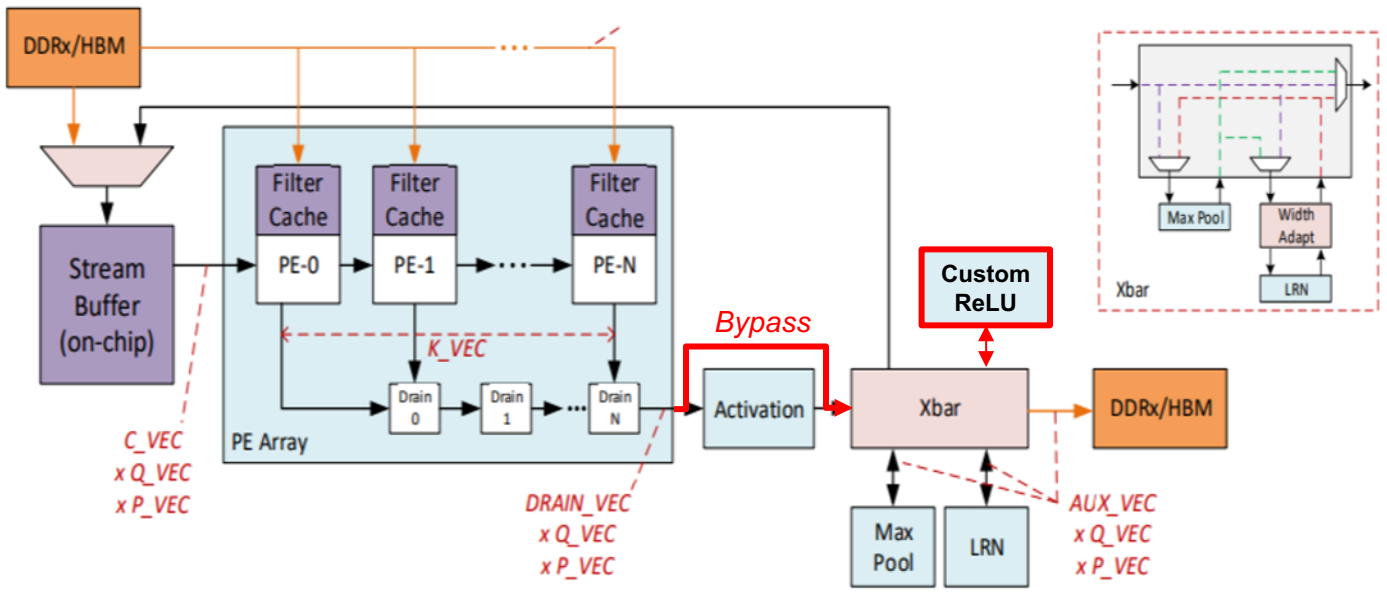
Observations:

Inefficiency of DLA architecture to process the normalization and activation layers caused *FPGA* performance to be very *poor*

Predictions:

By *reversing* the LRN & Activation layers in the CosmoGAN, the predicted optimal performance can be up to *3x speedup vs 1 core CPU*

CosmoGAN: Improved Results with Custom DLA



DLA w/ "custom ReLU" module

Solution:

- Bypass the *original* "Activation" module
- Add a *new* custom ReLU activation module as a custom primitive attached to the "Xbar"

CosmoGAN: Improved Results with Custom DLA

FPGA vs. CPU performance *for CosmoGAN*

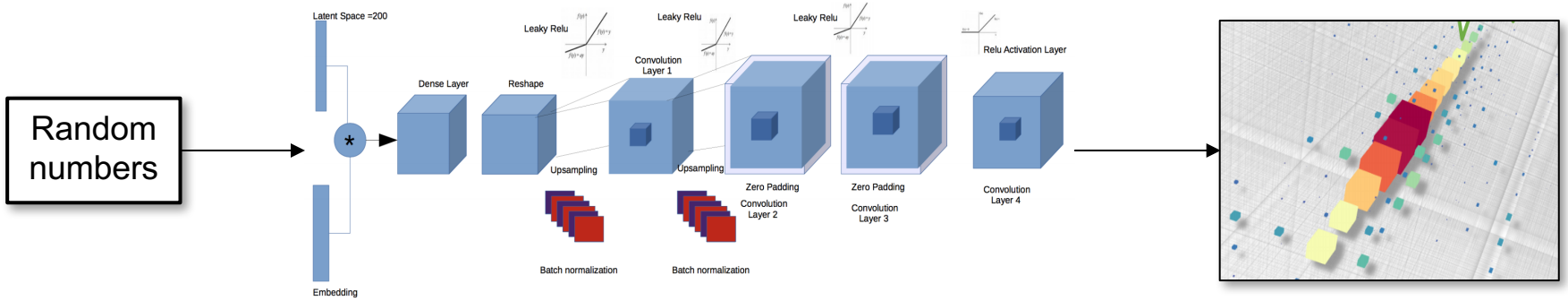
	Bit precision	FPGA * Throughput	CPU (1 core/1 thread) ** Throughput	CPU (32 core/64 thread) ** Throughput	FPGA speedup vs. 1 core CPU
Native DLA 2018	FP16	**4.7 imgs/sec**	**22.4 imgs/sec**	**156 imgs/sec**	**0.21x**
↓		↓			↓
Custom DLA 2018	FP16	***55.2 imgs/sec***	**22.4 imgs/sec**	**156 imgs/sec**	***2.46x***

* Arria 10 at 20 nm process
** Intel Xeon Gold 6130 CPU at 14 nm process

Conclusion:

- Native DLA speedup: *0.21x* vs 1-core/1-thread Intel Xeon CPU
  - Slow due to inefficient architecture
- Custom DLA speedup: *2.46x* vs 1-core/1-thread Intel Xeon CPU
  - By processing LRN before Activation, only one single *iteration is required* for each deconvolutional layer
  - Using custom DLA *12x* faster than native DLA

Case Study: 3DGAN[1] Model from CERN openlab



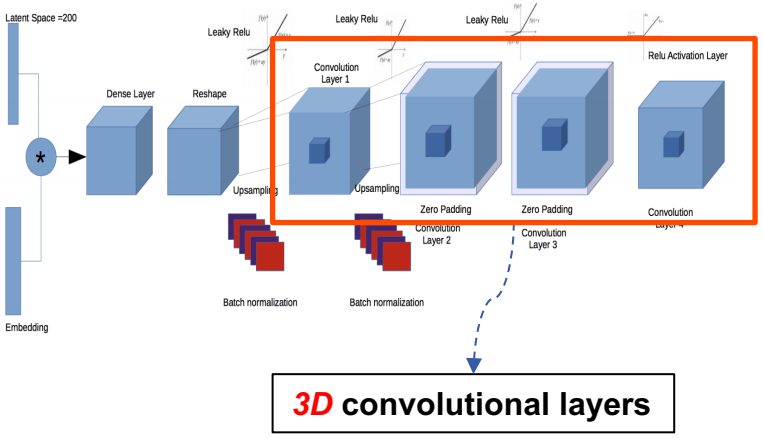
Particle shower simulation

- Generate a 3D simulation of a particle detector in HEP experiments
- Developed and trained by openlab at CERN using *3DGAN* topology with upsampling + 3D convolutional layers

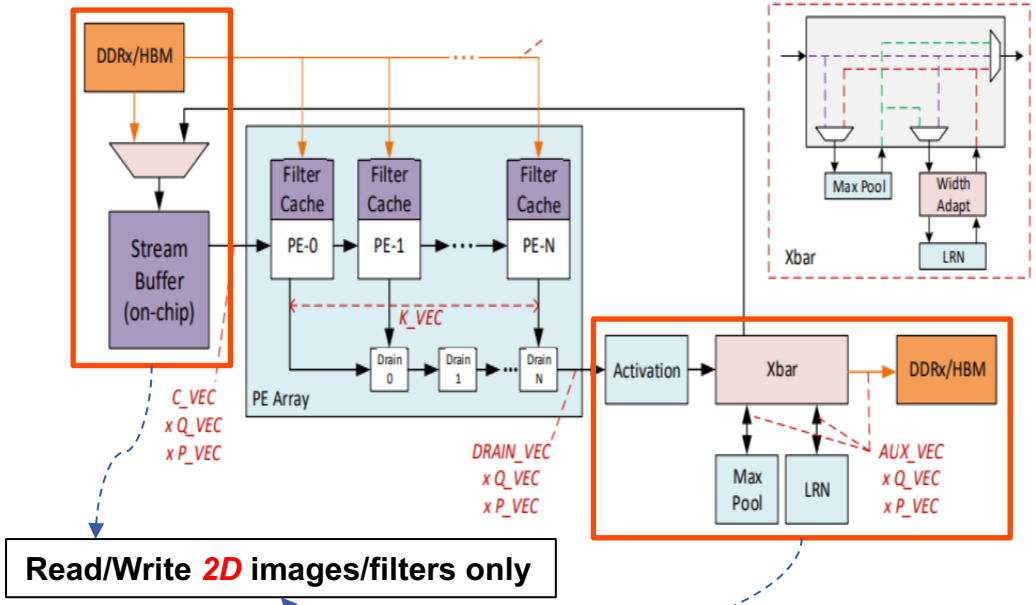
[1]

FPGA Primitive to Support 3D Convolutional Layer

3DGAN model structure



System-level architecture of DLA



Problem:

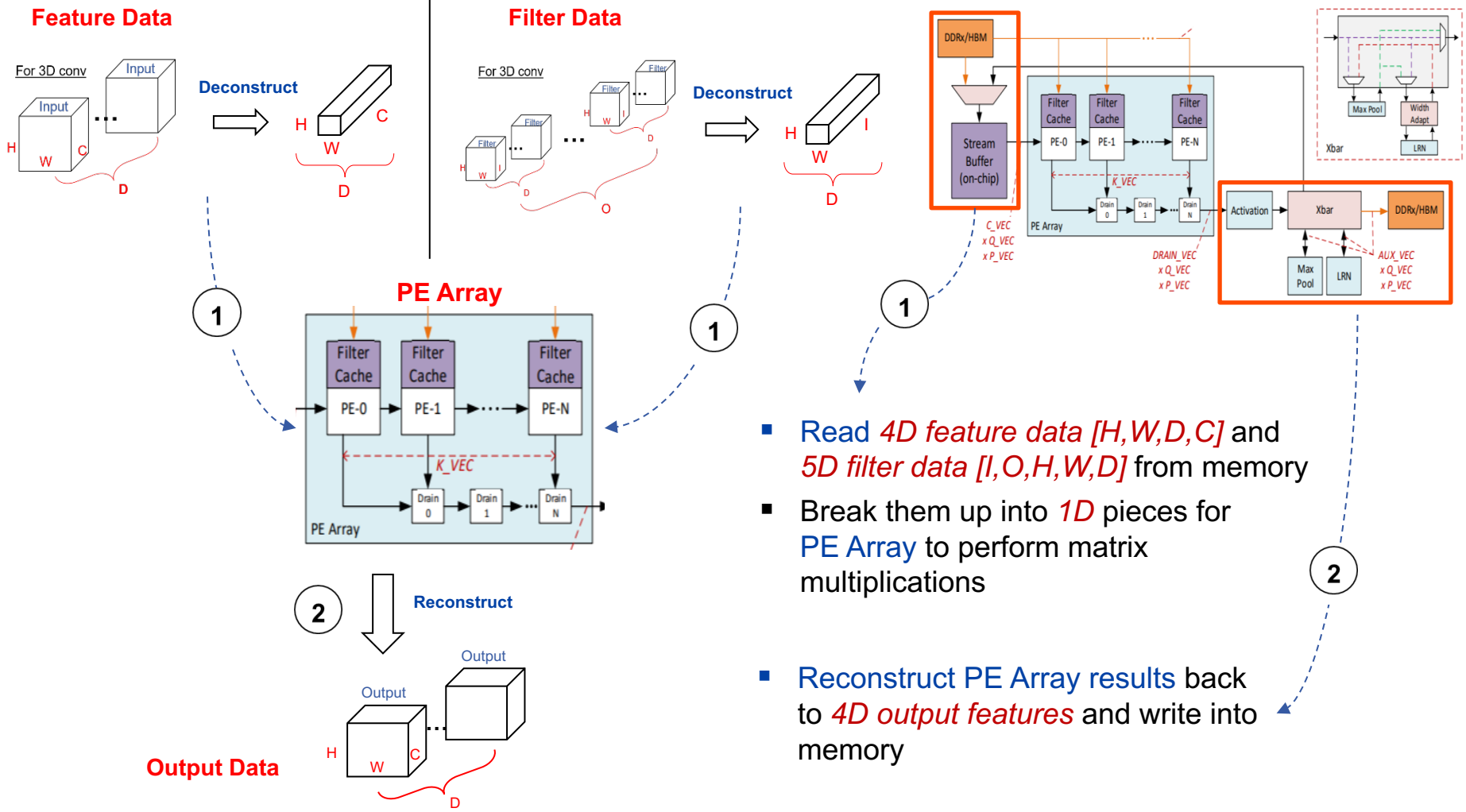
- DLA can not inference *3D convolutional layers on FPGA* natively

Solution:

- *Customize* DLA to implement *3D convolutional primitive*

[1] Abdelfattah, M.S. et al., DLA: Compiler and FPGA Overlay for Neural Network Inference Acceleration arXiv e-prints arXiv:1807.06434 (Jul 2018)

FPGA Primitive to Support 3D Convolutional Layer



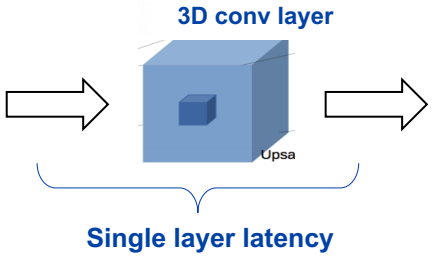
- Read *4D feature data [H,W,D,C]* and *5D filter data [I,O,H,W,D]* from memory
- Break them up into *1D* pieces for PE Array to perform matrix multiplications
- Reconstruct PE Array results back to *4D output features* and write into memory

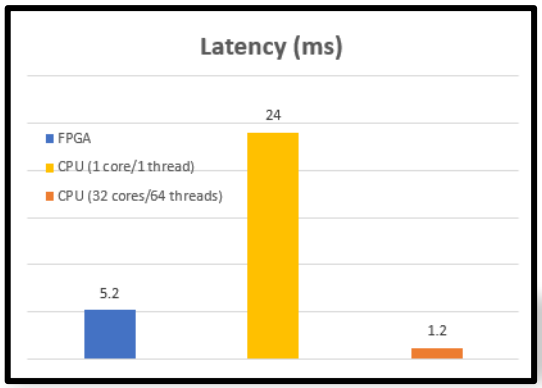
[1] Abdelfattah, M.S. et al., DLA: Compiler and FPGA Overlay for Neural Network Inference Acceleration, arXiv e-prints arXiv:1807.06434 (Jul 2018)

3DGAN: Initial Results (Single Conv3d Layer)

FPGA vs. CPU performance *for single 3D convolutional layer*

Bit precision	FPGA * Latency	CPU (1 core/1 thread) ** Latency	CPU (32 core/64 thread) ** Latency	FPGA speedup vs. 1 core CPU
FP16	***5.2 ms/layer***	**24 ms/layer**	**1.2 ms/layer**	***4.6x***





* Arria 10 at 20 nm process
** Intel Xeon Gold 6130 CPU at 14 nm process

Conclusion:

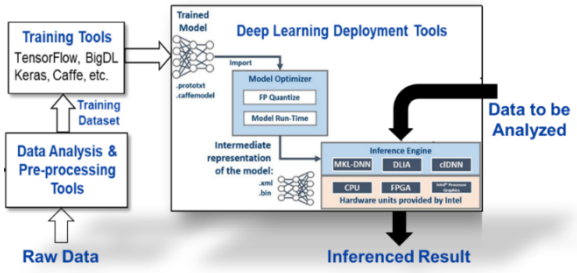
- Successfully inference *3D conv layer* on FPGA with customized DLA

Going forward:

- *Integrate* the customized DLA OpenCL code with *Intel OpenVINO toolkit* to performance complete 3DGAN inference
- Perform *design space exploration* for further optimisation

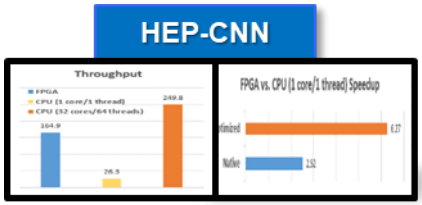
Summary & Conclusions

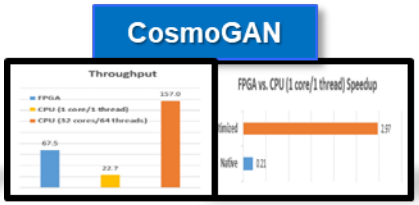
- Heterogeneous computing for deep learning

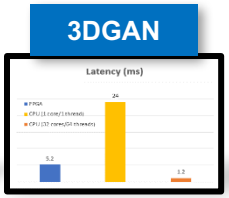


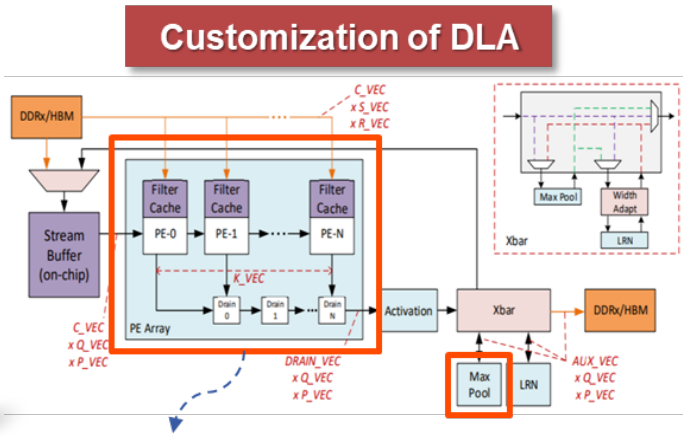
- Collaboration with CERN openlab & NERSC on scientifically relevant DNN
- SHREC: Focused on FPGA-acceleration of inference stage

- Exploration of FPGA-based platforms & tools
  - Intel PAC Arria 10 card; OpenVINO; DLA design suite
  - Explore use and improvement of state-of-art tools









Conclusions & Going Forward

- Continue to explore & improve FPGA-based DNN platforms & tools
  - Scale up multiple FPGAs for faster inference
  - Explore FPGA+ for efficient DNN model training
- Appropriate use of FPGA-based DNN platforms
  - Compare FPGA-based platform vs. CPU, GPU, & other emerging devices (energy, size, weight, cost, etc.)
  - Determine appropriate missions for FPGA-based systems

QUESTIONS