

Fast inference on FPGAs

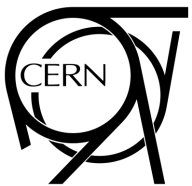
Javier Duarte, Sergio Jindariani, Ben Kreis, Ryan Rivera, Nhan Tran (Fermilab)
Jennifer Ngadiuba, Maurizio Pierini, Sioni Summers, **Vladimir Loncar** (CERN)

Edward Kreinar (Hawkeye 360)

Phil Harris, Song Han, Dylan Rankin (MIT)

Zhenbin Wu (University of Illinois at Chicago)

Giuseppe di Guglielmo (Columbia University)

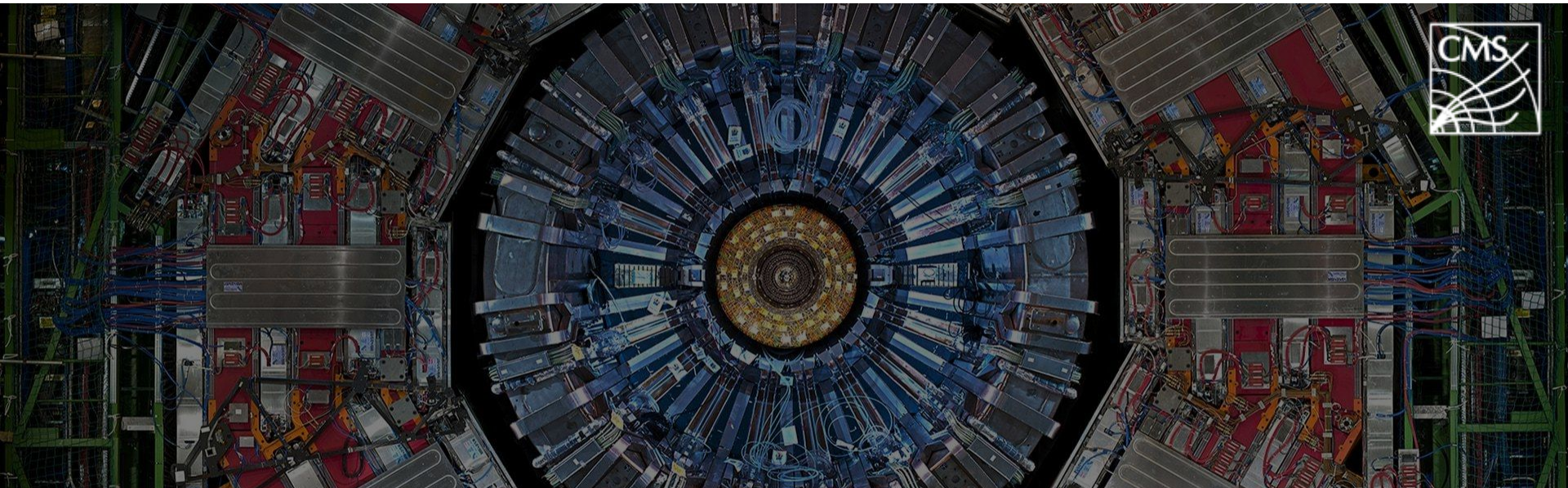


Challenges in LHC

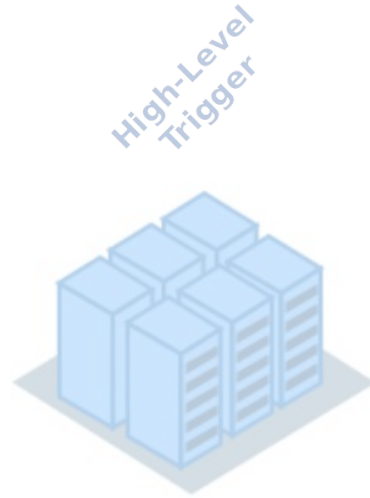
At the LHC proton beams collide at a frequency of 40 MHz

Extreme data rates of $O(100 \text{ TB/s})$

“Triggering” - Filter events to reduce data rates to manageable levels



The LHC big data problem



DATA FLOW



The LHC big data problem



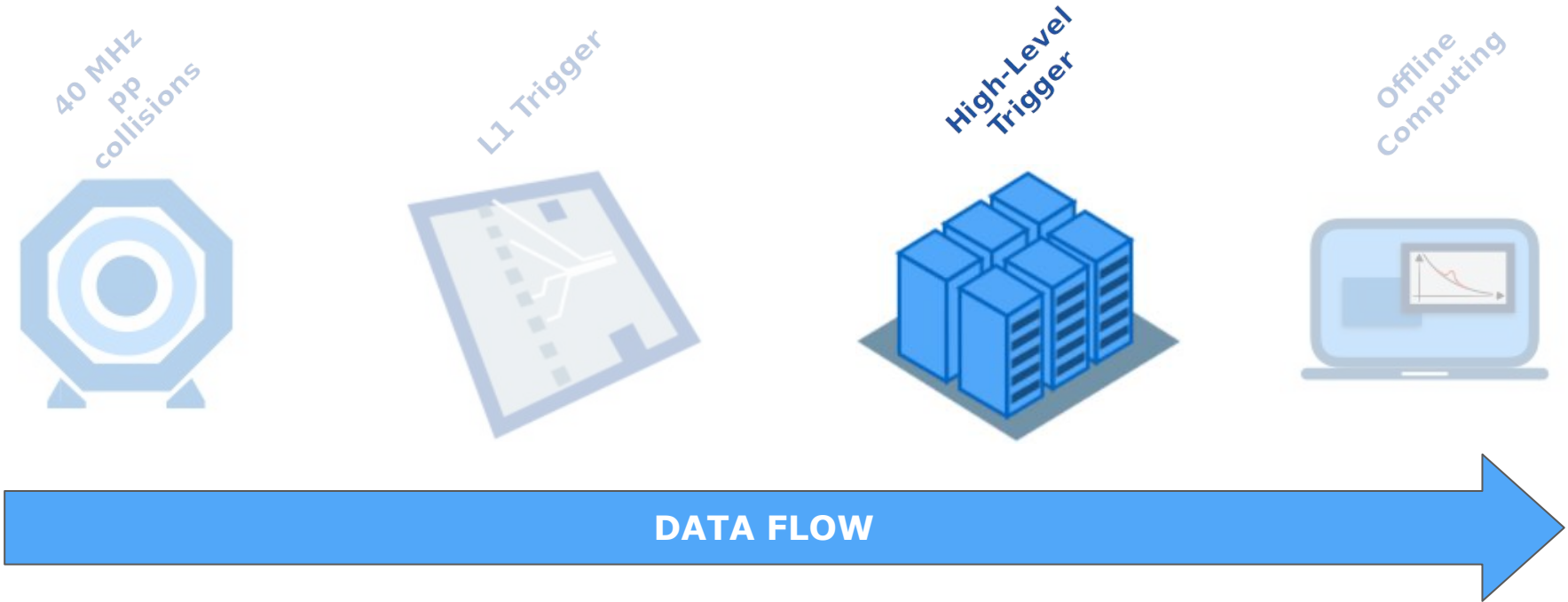
DATA FLOW

40 MHz in / 100 KHz out \Rightarrow absorbs 100s TB/s

Trigger decision to be made in $\sim 10 \mu\text{s}$

FPGAs / Hardware implemented

The LHC big data problem

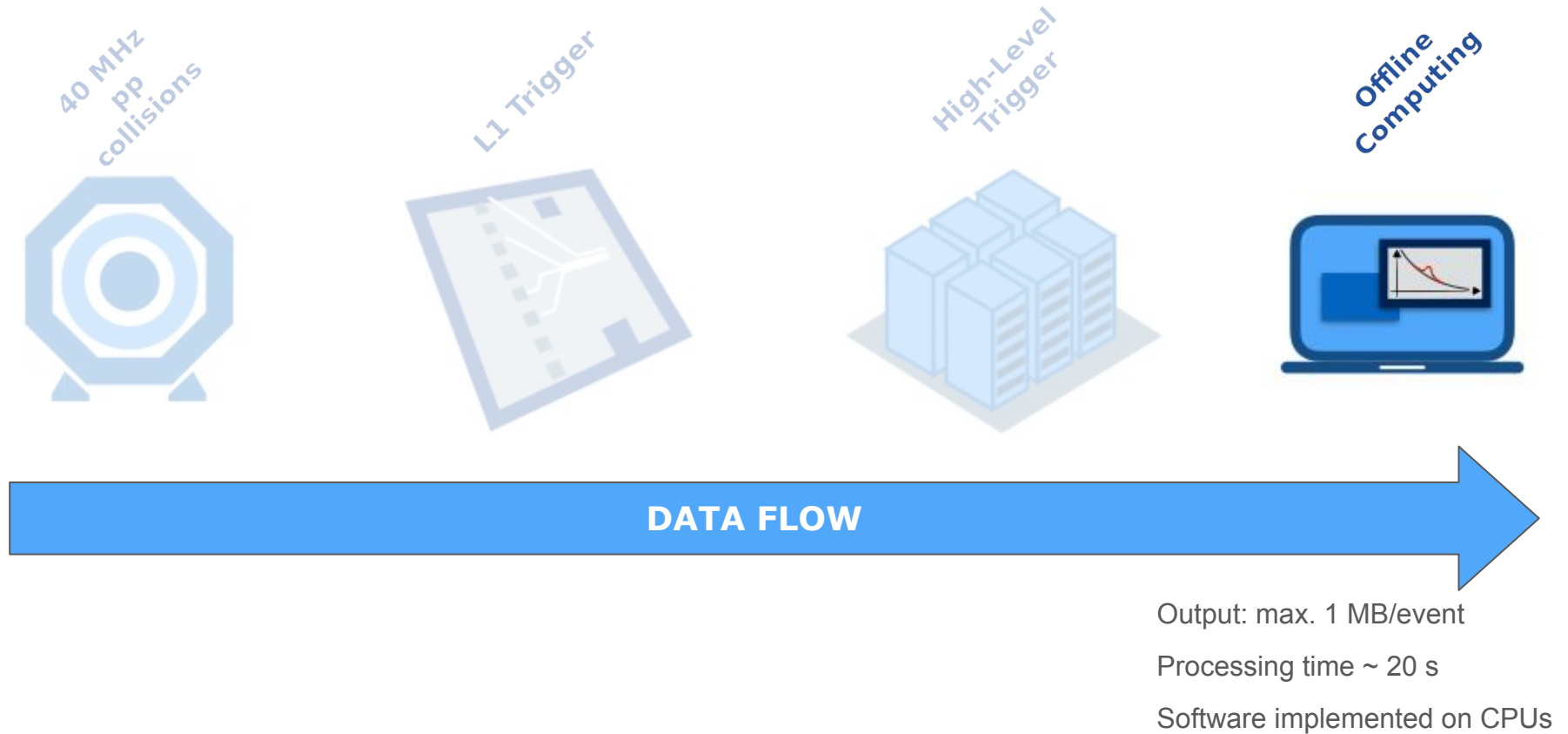


100 KHz in / 1 KHz out \Rightarrow \sim 500 KB/event

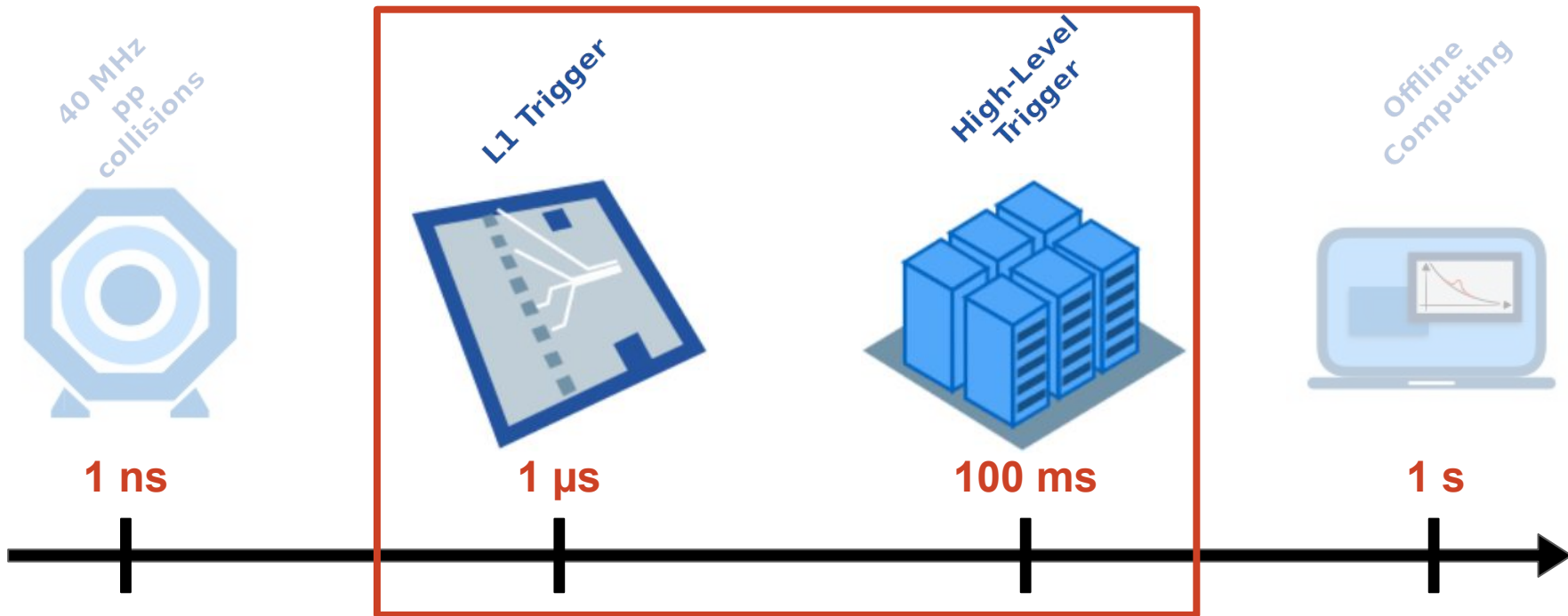
Processing time \sim 300 ms

Software implemented on CPUs

The LHC big data problem



The LHC big data problem

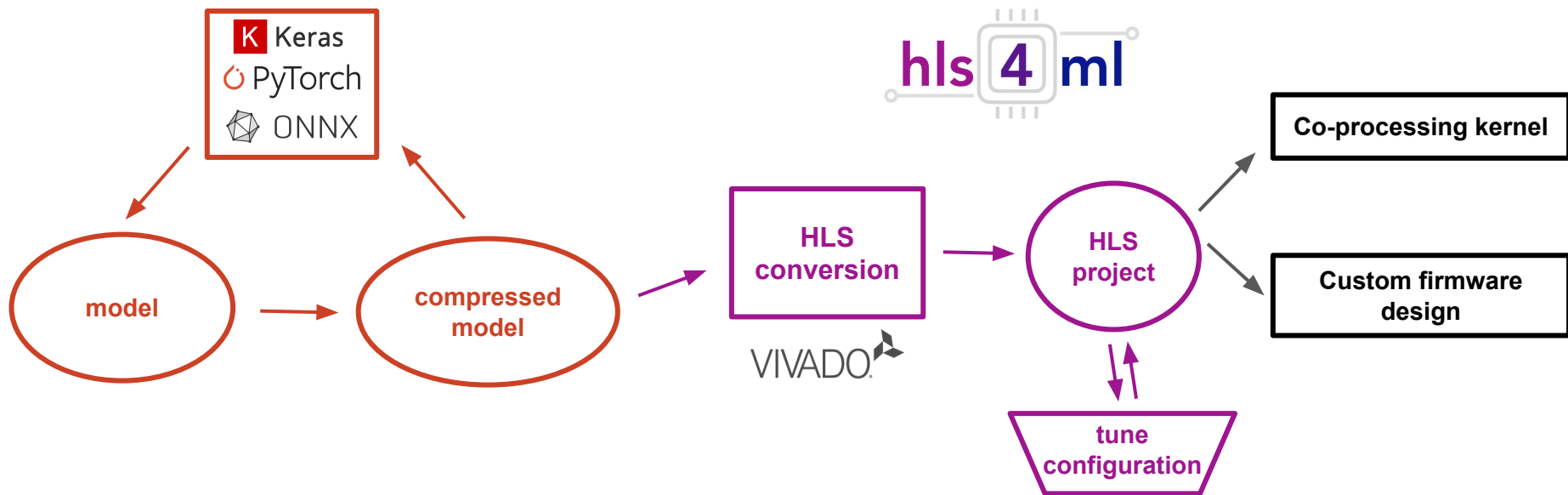


Deploy ML algorithms very early
Challenge: strict latency constraints!

high level synthesis for machine learning

User-friendly tool to automatically build and optimize DL models for FPGAs:

- Reads as input models trained with standard DL libraries
- Uses Xilinx HLS software
- Comes with implementation of common ingredients (layers, activation functions, binary NN ...)



hls4ml : features

On-chip weights

- Much faster access times
- For longer latency applications, weights storage in on-chip block memory is possible
- No loading weights from external source (e.g. DDR, PCIe)
- Not reconfigurable without reprogramming device

User controllable trade-off between resource usage and latency/throughput

- Tuned via “reuse factor”

Fully extensible through API

- Custom layers, custom HLS code, user-defined model transformations...

hls4ml : reuse factor

A handle to control resource usage and latency

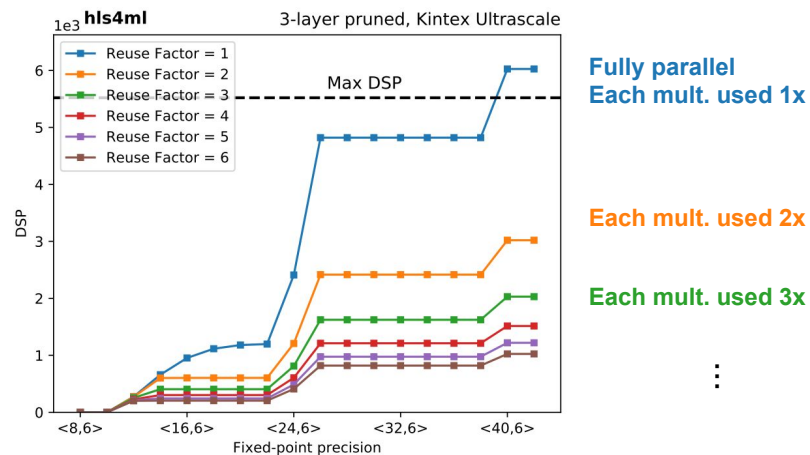
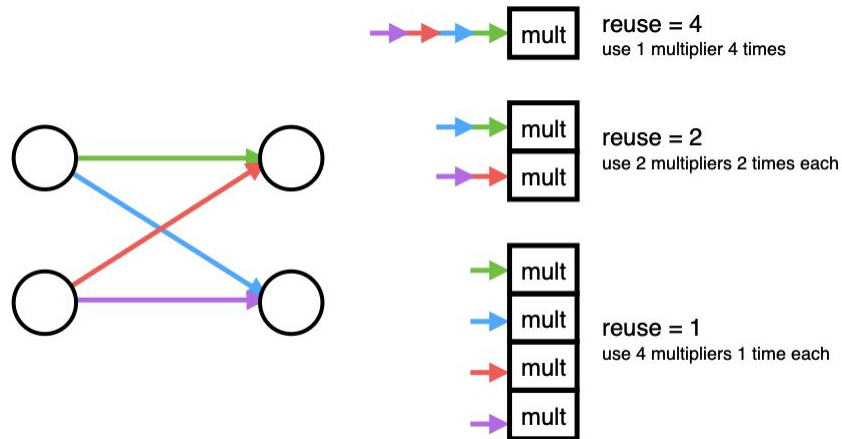
- Can be specified per-layer

Reuse = 1: Fully unroll everything

- Fastest, most resource intensive

Reuse > 1: reuse one DSP for several operations

- Increases latency, but uses less resources



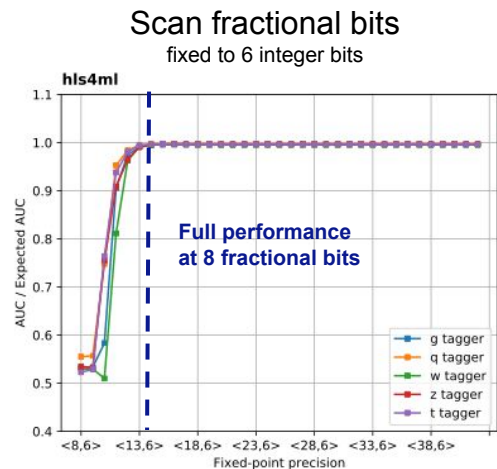
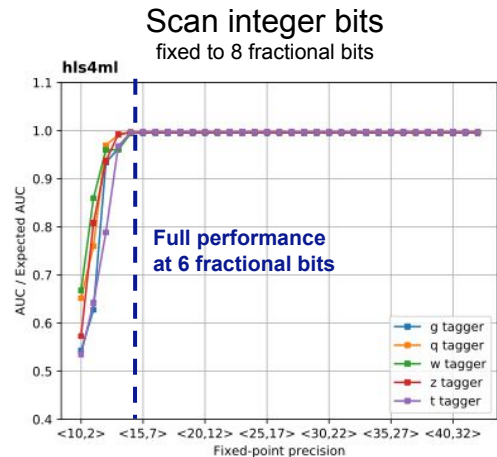
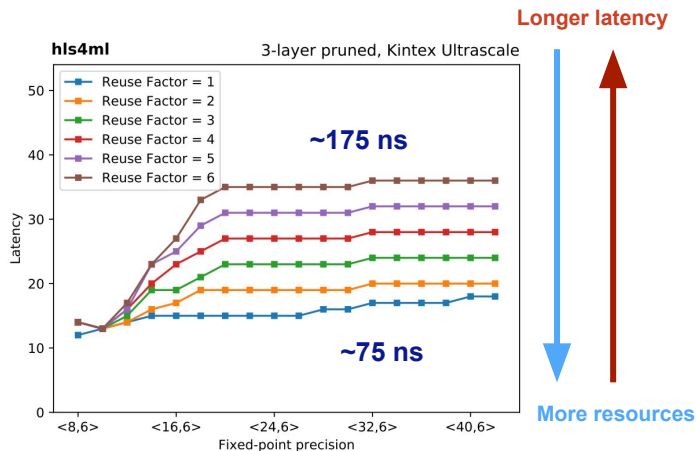
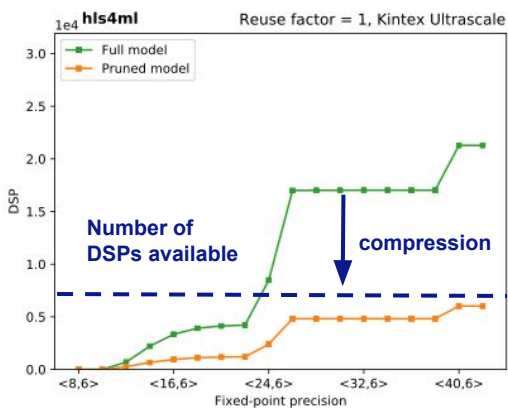
hls4ml : exploiting FPGA hardware

Parallelization (reuse): Control the inference latency versus utilization of FPGA resources

Quantization: Reduce precision of the calculations



Compression: Drop unnecessary weights (zero or close to zero) to reduce the number of DSPs used

70% compression ~ 70% fewer DSPs



hls4ml : current status

Supported architectures:

- **MLP**
 - Numerous activation functions
 - Support for very large layers 
- **Binary and Ternary MLP**
 - 1- or 2-bit precision with limited loss of performance
 - Computation without using DSPs, only LUTs
- **Convolutional NNs**
 - 1D and 2D with pooling
 - Currently limited to very small layers 
- **Other:**
 - Batch normalization
 - Merge layers (concatenation, addition, subtraction etc)

hls4ml : ongoing work (1)

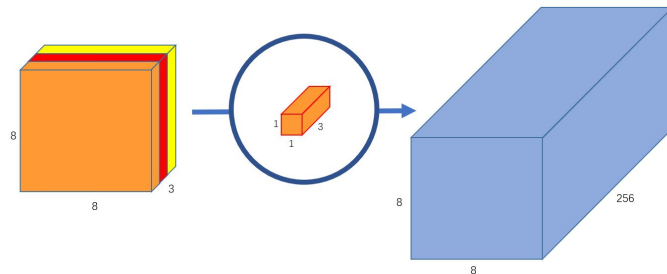
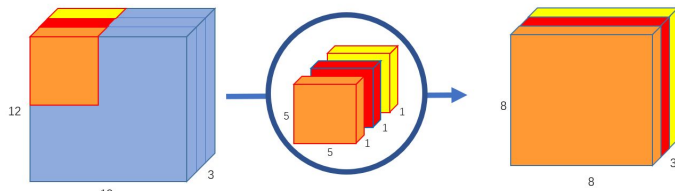
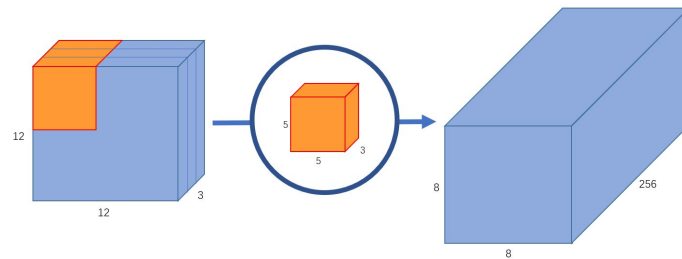
Convolutional layers

Support for “large” convolutional layers **SOON**

- Express convolution as matrix multiplication
- im2col algorithm
- Reuse “large” matrix multiplication algorithm from MLP
- Quantized (binary and ternary) weights

Depthwise separable convolution

- First step: depthwise convolution
- Second step: pointwise convolution
- For 3x3 kernels this can yield 8-9 times less multiplications



hls4ml : ongoing work (2)

Boosted decision trees

Q4 2019

- BDTs have been popular for a long time in HEP reconstruction and analysis
- Suitable for highly parallel implementation in FPGAs
- Implementation in hls4ml optimised for low latency
- No 'if/else' statement in FPGAs → evaluate all options and select the right outcome
 - Compare all features against thresholds, chain together outcomes to make the 'tree'

Test for model with 16 inputs, 5 classes, 100 trees, depth 3 on VU9P FPGA:

- 4% LUTs, 1% FFs (0 DSPs, 0 BRAMs)
- 25 ns latency with II=1

hls4ml : ongoing work (3)

Recurrent neural networks

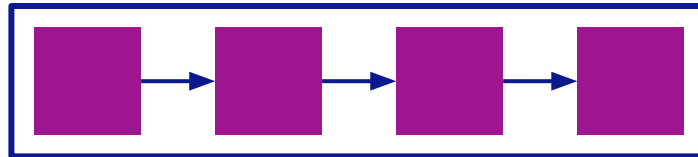
Q4 2019

- Simple RNN, LSTM, GRU

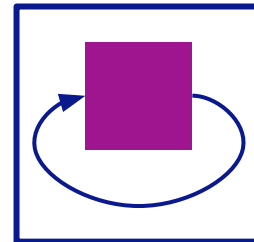
Two implementations:

- **Fully unrolled:**
 - Latency optimized with $II=1$
 - Large resource usage
- **Static:** same resources used for weights and multiplications
 - N (N =latency of layer) copies can go through at the same time
 - Latency is larger and II limited to clock time for each layer

Fully unrolled



Static

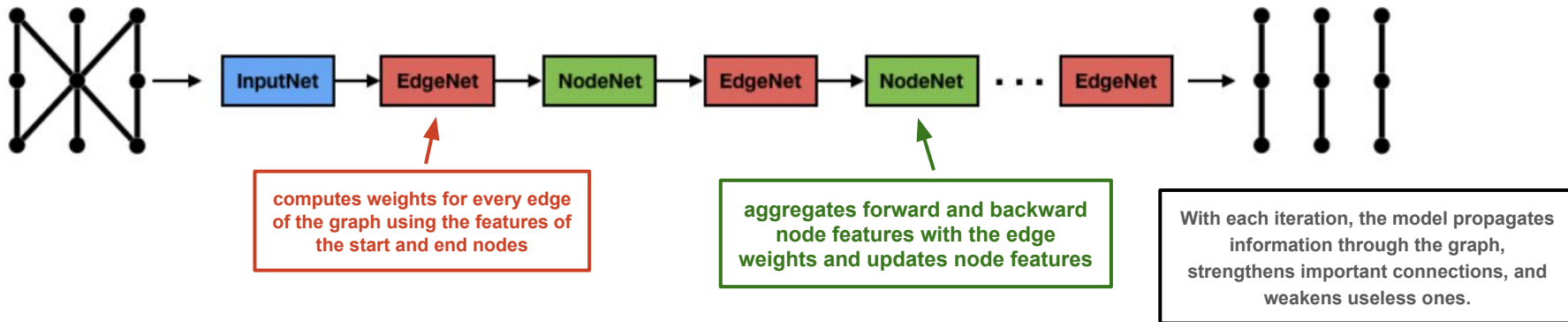


Supports small networks → scale it up using “large” matrix multiplication algorithm

hls4ml : ongoing work (4)

Graph networks H1 2020

- Natural solution for reconstructing the trajectories of charged particles



Preliminary implementation:

- Implemented as an HLS project, not supported in conversion tools
- Successfully tested a small example with 4 tracks, 4 layers
- Major effort required to scale up to larger graphs

Credit: Javier Duarte and Kazi Asif Ahmed Fuad

hls4ml : future directions (1)

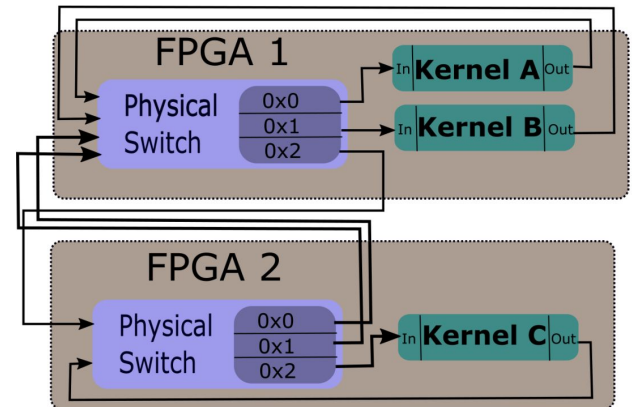
Multi-FPGA inference

H1 2020

- Main idea: place layers onto multiple FPGAs and pipeline the execution

Leverage Galapagos framework (<https://github.com/tarafdar/galapagos>)

- “...a framework for creating network FPGA clusters in a heterogeneous cloud data center.”
- Given a description of how a group of FPGA kernels are to be connected, creates a ready-to-use network device
- Possible to use MPI programming model



Credit: Naif Tarafdar, Phil Harris

hls4ml : future directions (2)

Training on FPGAs

H2 2020

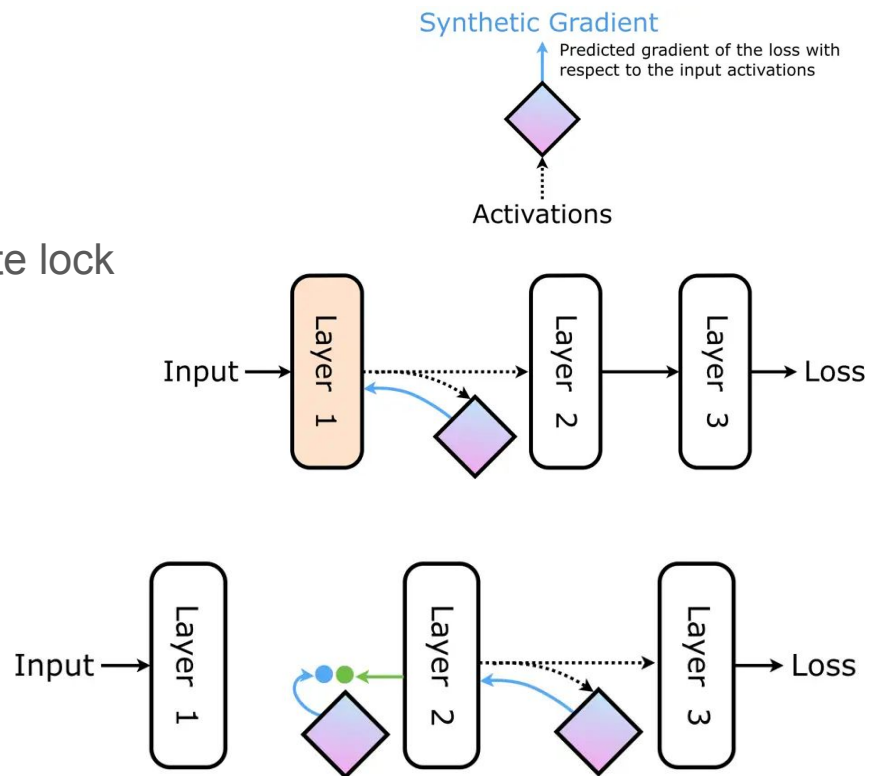
- Build on top of multi-FPGA idea

Use synthetic gradients (SG) to remove the update lock

- Individual layers to learn in isolation

Train SGs by another NN

- Each SG generator is only trained using the SGs generated from the next layer
- Only the last layer trains on the data



hls4ml : other future developments

Autoencoders

GarNet graph NN (<https://arxiv.org/abs/1902.07987>)

Alternate HLS implementations

- Intel HLS
- Mentor Catapult HLS

Inference engine for CPUs based on hls4ml

- Targeting integration into CMSSW

Probably more...

Conclusions

hls4ml - software package for translation of trained neural networks into synthesizable FPGA firmware

- Tunable resource usage latency/throughput
- Fast inference times, $O(1\mu\text{s})$ latency

More information:

- Website: <https://hls-fpga-machine-learning.github.io/hls4ml/>
- Paper: <https://arxiv.org/abs/1804.06913>
- Code: <https://github.com/hls-fpga-machine-learning/hls4ml>

