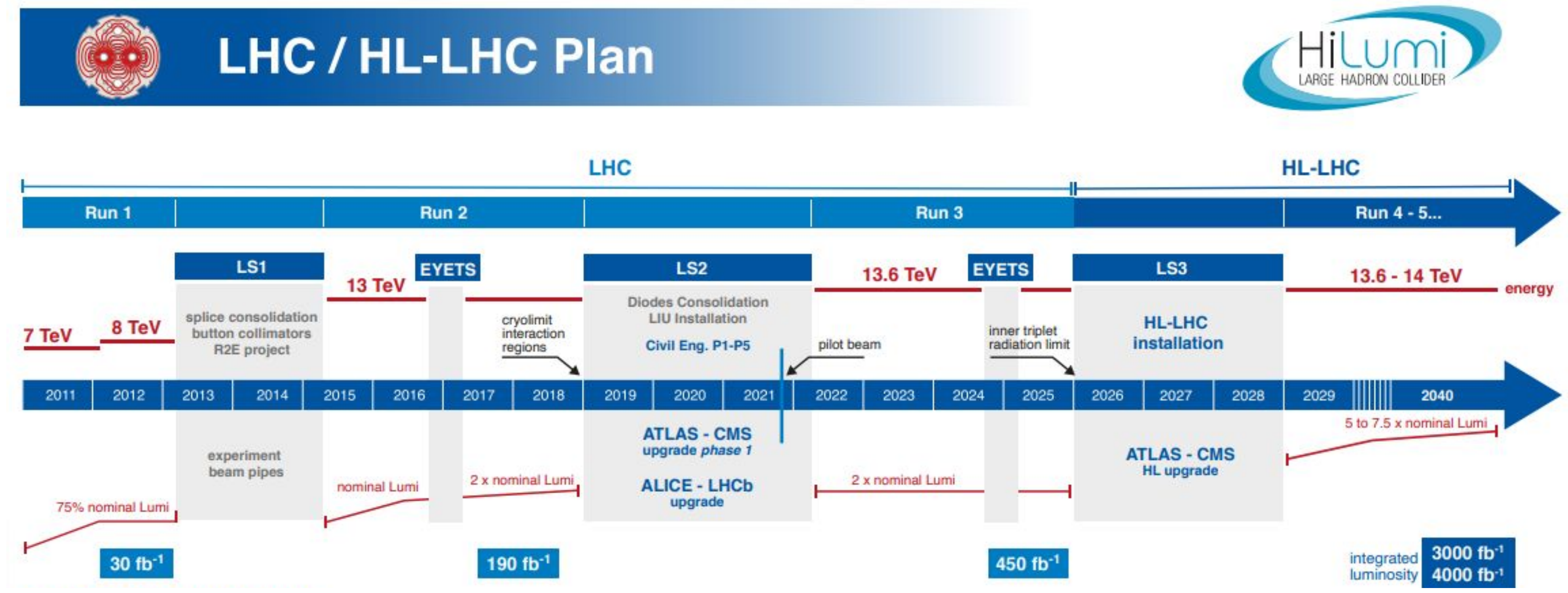
Embedded Recurrent Neural Networks on FPGAs for Real-Time Computation of the Energy Deposited in the ATLAS Liquid Argon Calorimeter

CEPC - Marseille
09/04/2024

Georges Aad
CPPM, Aix-Marseille Université, CNRS/IN2P3, Marseille

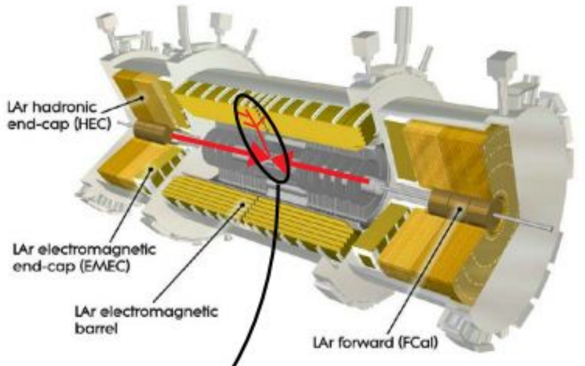
Introduction

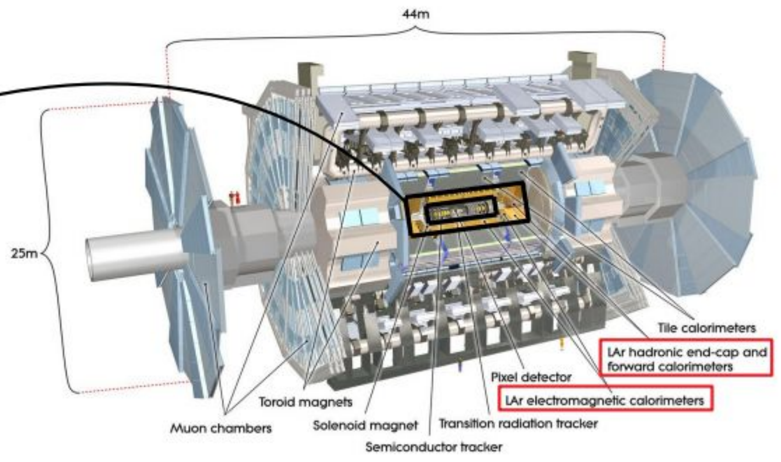
- LHC upgrade during the long shutdown starting 2026 leading to the HL-LHC
  - Increase the instantaneous luminosity by a factor 5 to 7 with respect to the LHC design value
  - 140 to 200 simultaneous proton-proton collisions (pileup)
- ATLAS will be upgraded to cope with the HL-LHC conditions
  - Increase the level 1 trigger frequency from 100 kHz to 1 MHz
  - ➢ New readout electronics for the liquid argon calorimeter



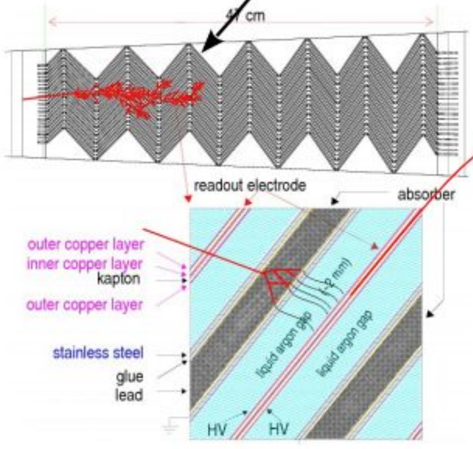
The ATLAS Liquid Argon Calorimeter

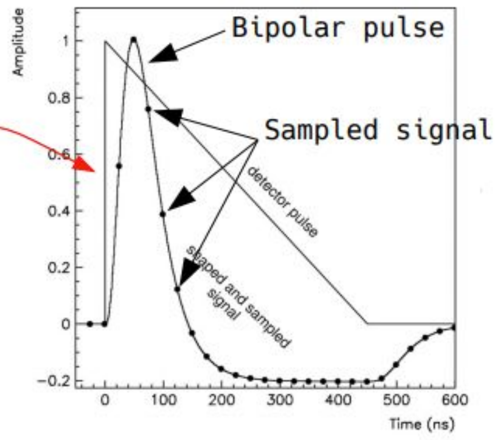
- Measures the energy of electromagnetically interacting particles mainly electrons and photons
- Trigger capabilities at the first level of triggering (implemented in hardware)
  - Fast processing of the data needed (at 40 MHz)


LAr hadronic end-cap (HEC)
LAr electromagnetic end-cap (EMEC)
LAr electromagnetic barrel
LAr forward (FCal)


44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

180000 channels

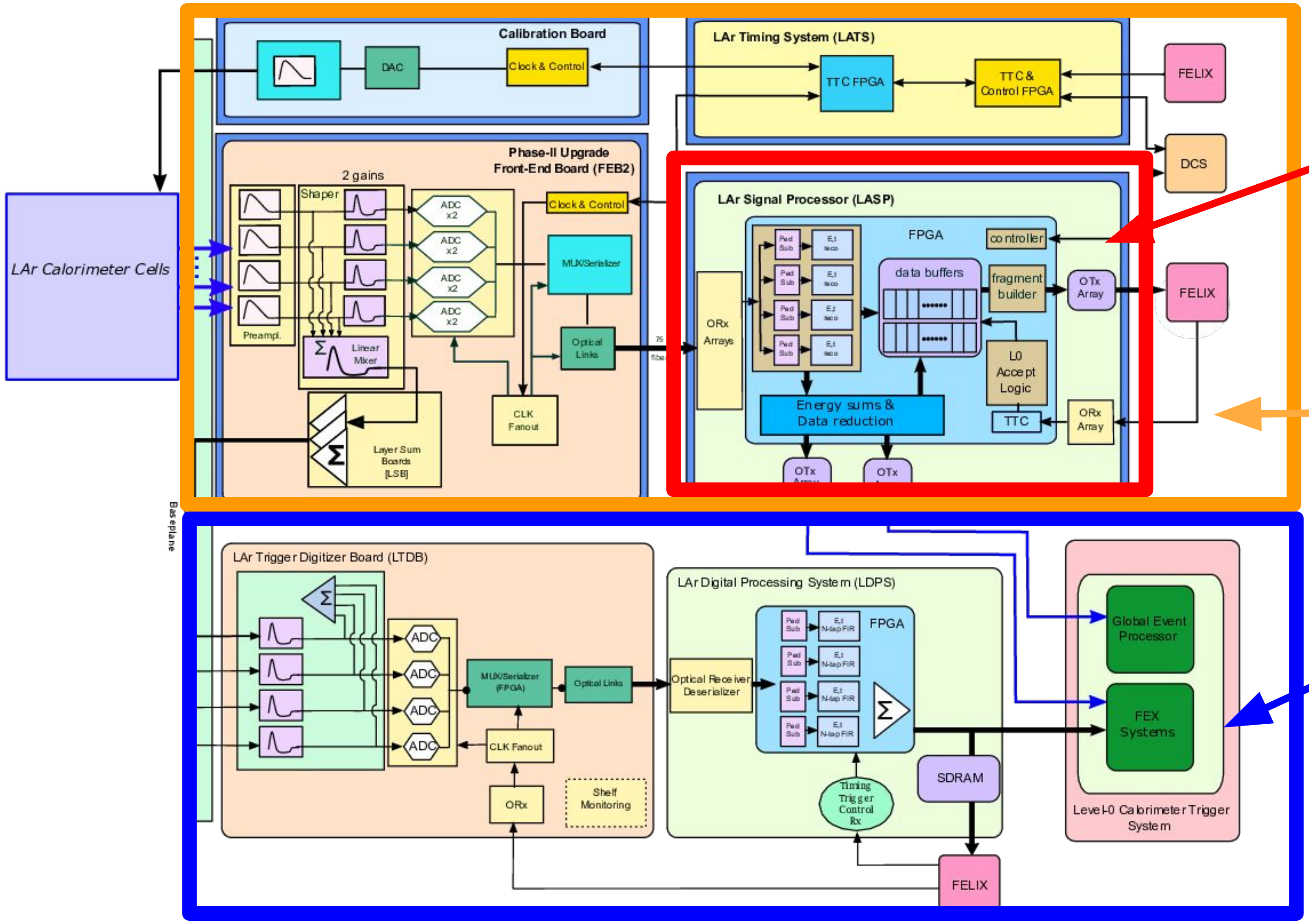




- Electronic signal amplitude proportional to the deposited energy in the calorimeter
- Shaped and sampled at 40 MHz

LAr Phase-II Upgrade

- Full electronics of the readout path will be exchanged
  - New on-detector electronics that will digitize the signal at 40 MHz and send it to the backend
  - New off-detector electronics to compute the energy at 40 MHz



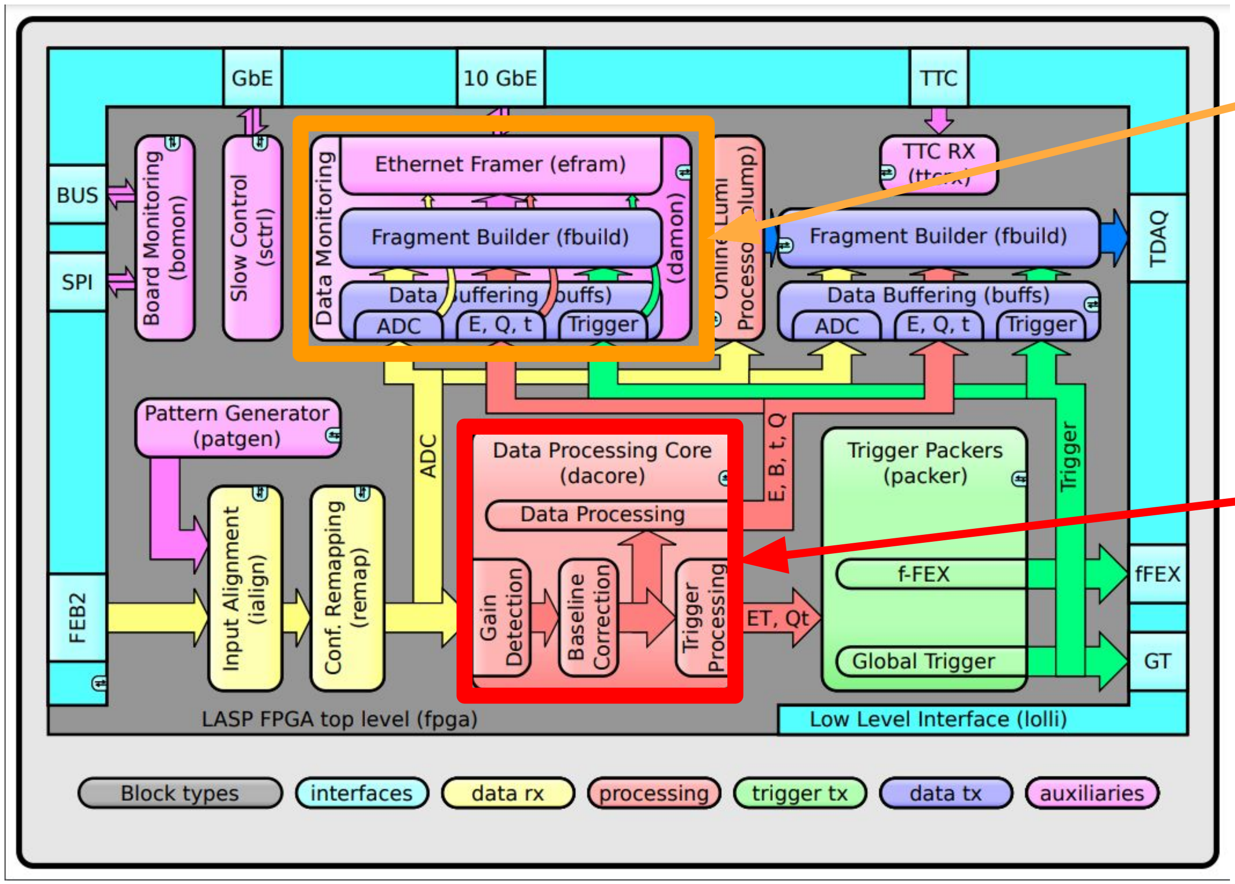
LASP board
Off-detector board responsible of energy computation

Phase-II electronics
Trigger path
+
readout path

Phase-I electronics
Trigger path

LASP Firmware

- LASP board containing 2 processing units based on INTEL FPGAs
  - Demonstrator board available with Stratix 10 FPGAs
    - Baseline for the firmware development shown in this talk
  - Final board will be equipped with Agilex FPGAs
- One FPGA should process 384 channels
  - About 125 ns allocated latency for energy computation

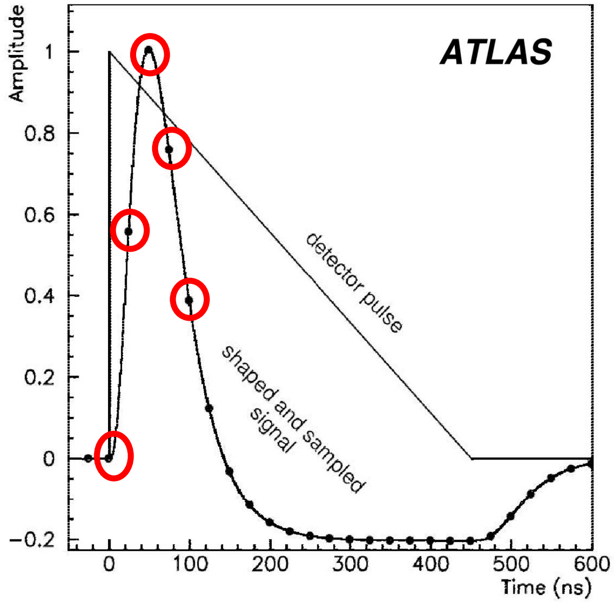


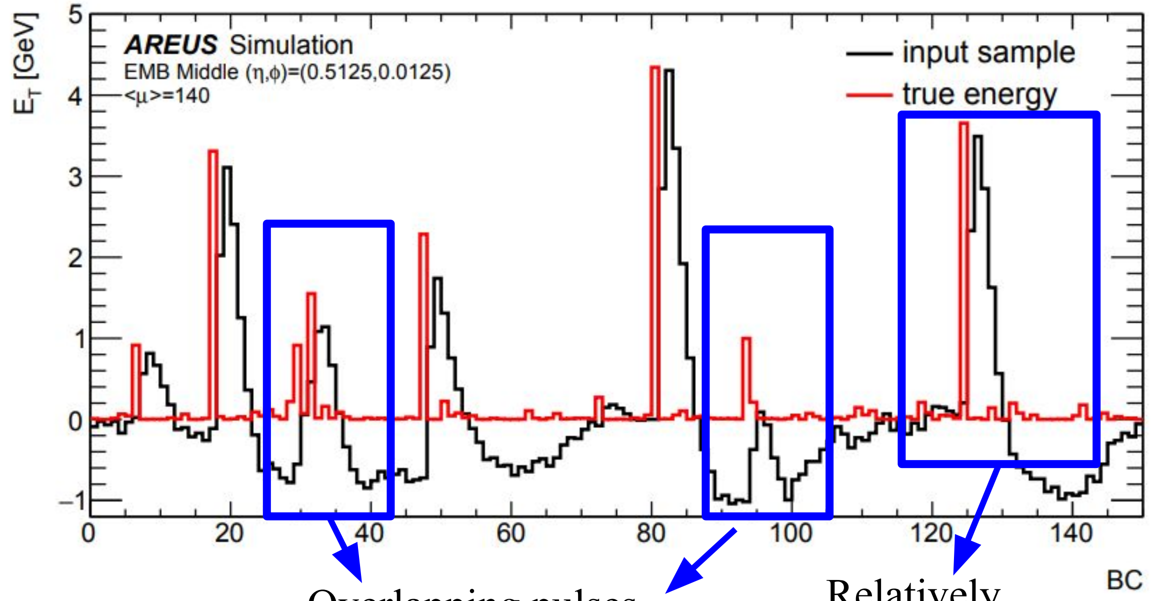
Buffering, waiting for L1A

Compute energy at 40 MHz
Assign the energy to the correct bunch crossing (collision time)

Energy reconstruction

- Legacy energy reconstruction using an optimal filtering algorithm with maximum finder (OFMax)
  - Optimal filtering to reconstruct pulse amplitude
  - Max finder to determine the correct time
- Not robust for distorted shapes due to pileup
- Neural networks promising candidate to recover performance in high pileup conditions
  - CNNs and RNNs considered
  - Only RNNs covered in this talk





Overlapping pulses
Distorted pulse shape

Relatively isolated pulse

Energy from Optimal-Filter (OF)

n = 5 in this talk

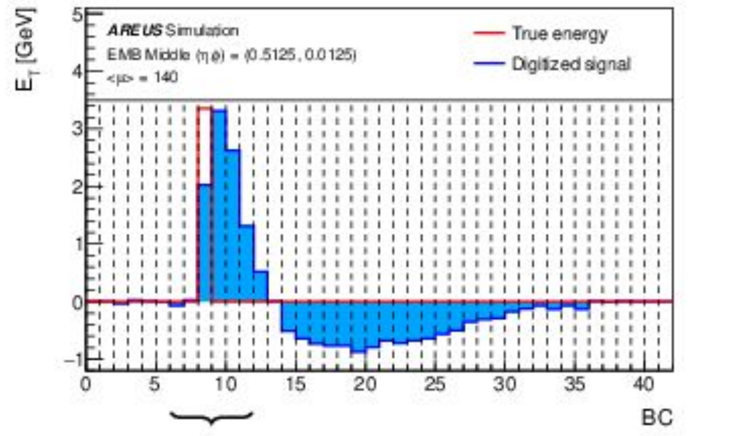
$$E(t) = \sum_{i=t}^{t+n} a_i \cdot s_i$$

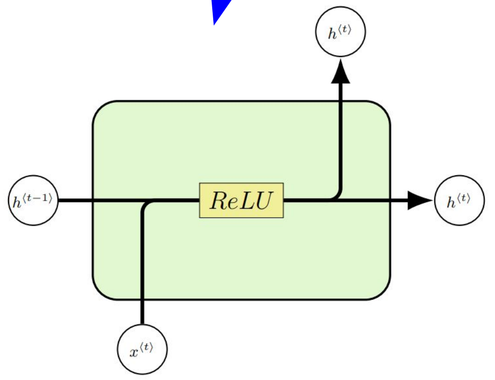
Pulse Samples

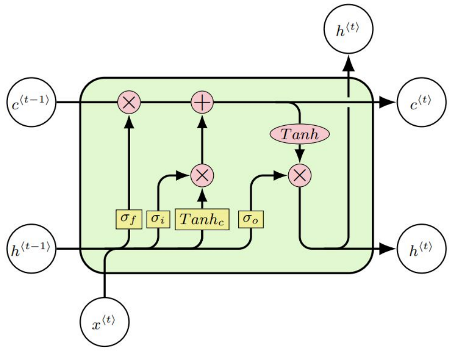
Pre-set coefficients (fit of the peak)

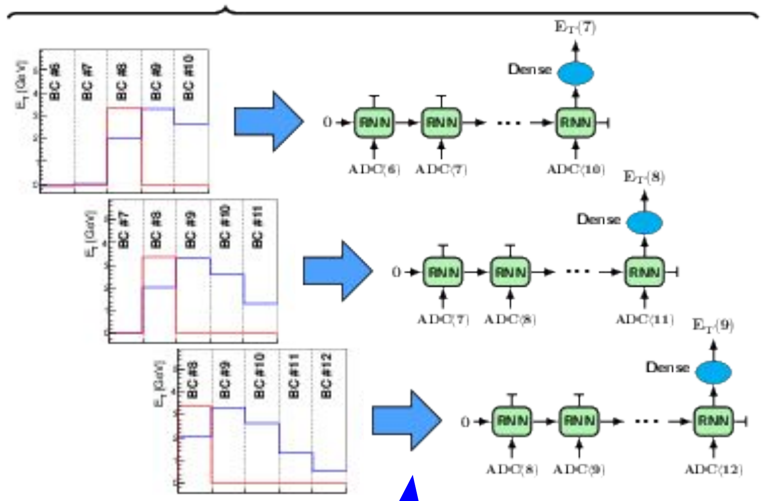
RNN structure

- Sequence of RNN cells each taking as input an ADC sample at a given BCID
  - 4 samples on the pulse
  - N samples prior to the pulse to correct for pileup
- Two general parameters control network size
  - Sequence length (number of samples)
  - NN units (internal dimension of the NN cells)
- Several cell structures tested
  - Vanilla RNN, GRU, LSTM









<u>Sliding windows architecture</u>

Computation on a moving slice (fixed intervals)

Takes into account a limited set in the past

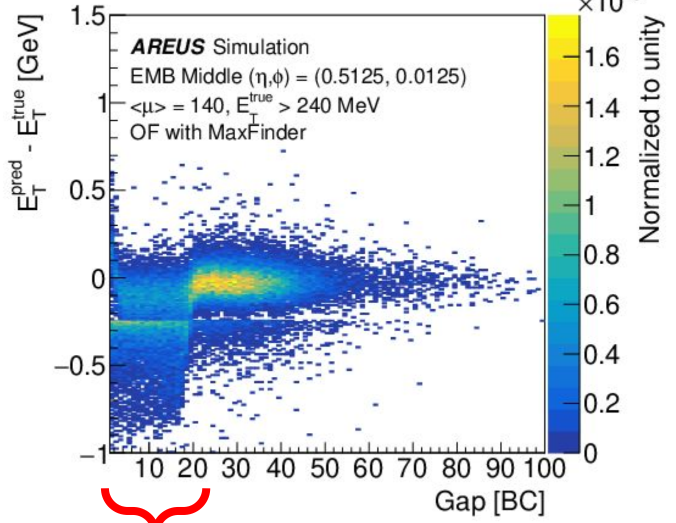
(1 sample in the past for this example)

Performance as Function of Time Gap

Lauri Laatu, PhD thesis (2023)

- Energy resolution as function of the time gap between two pulses to isolate pileup effects
- Clear drop in OFMax performance when pulses overlap
  - Time gap of less than ~ 20 BC
- Neural networks recover the performance in this region
  - Strongly dependent on the number of samples used in the past (prior to the energy deposit)
- Vanilla RNNs chosen due to their simplicity
  - Easier to fit into FPGAs

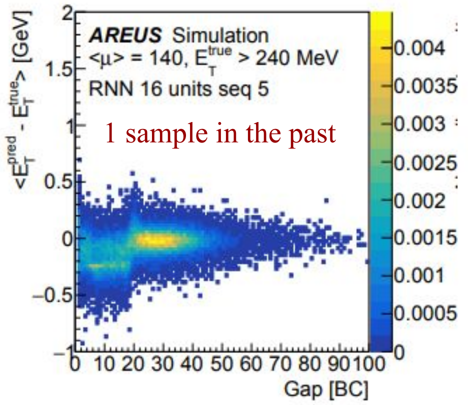
OFMAX

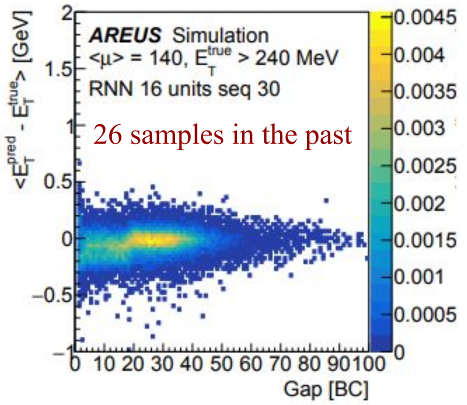


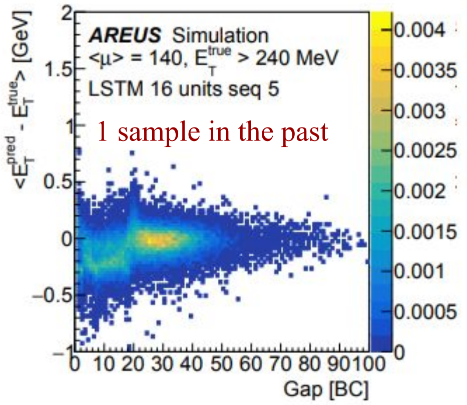
Overlapping signals region

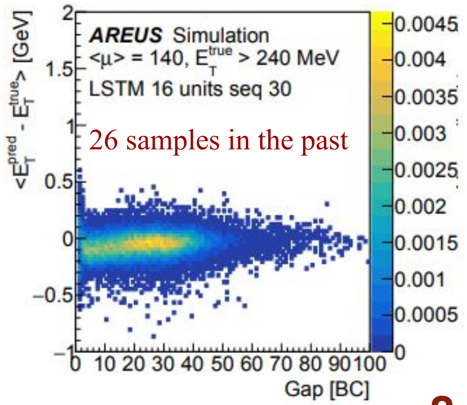
Vanilla RNN

LSTM





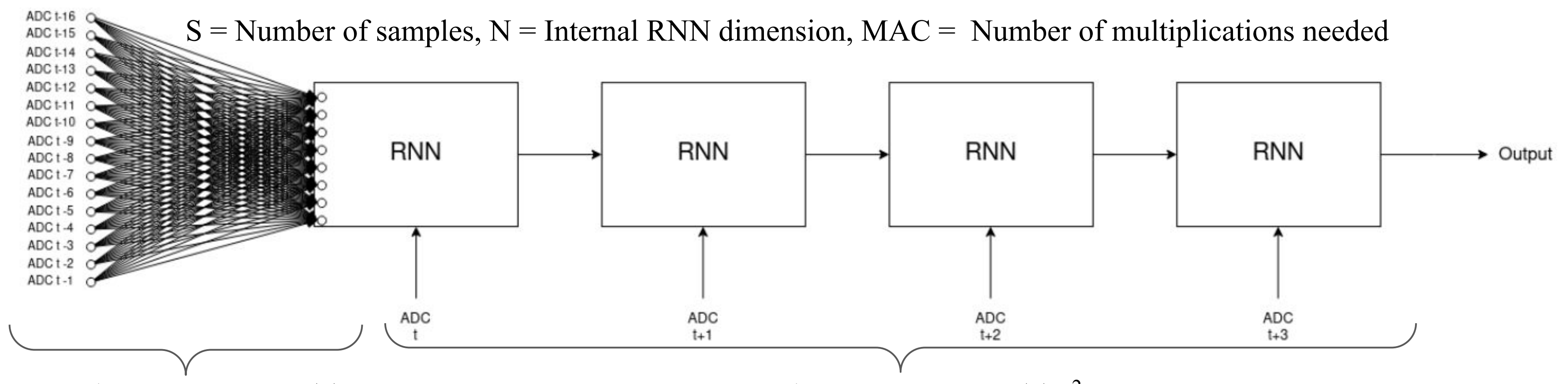




Lauri Laatu, PhD thesis (2023)

Optimisation of computational resources

S = Number of samples, N = Internal RNN dimension, MAC = Number of multiplications needed

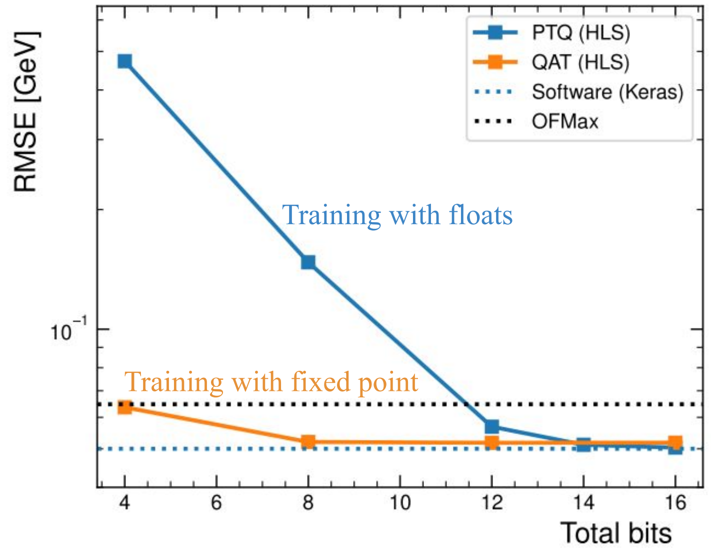


Dense layer : MAC $\propto$ S$\times$N

RNN layers : MAC $\propto$ S$\times$N^2

- Long sequences needed to efficiently correct for pileup
  - Significant computational resources for RNN cells
- Replace RNN cells in the past by a dense layer
  - Dense to correct to pileup, RNN to compute the amplitude
  - Reduce the number of needed multiplications by a factor 4
    - For a network with dimension 30 and sequence length 20
  - No effect on performance
- Reduce number of bits needed for arithmetic computation
  - Replace floating point with fixed point operation
  - Train the network directly with fixed point (QAT)
  - Quantization aware training (QAT) can reduce the number of needed bits by a factor 2

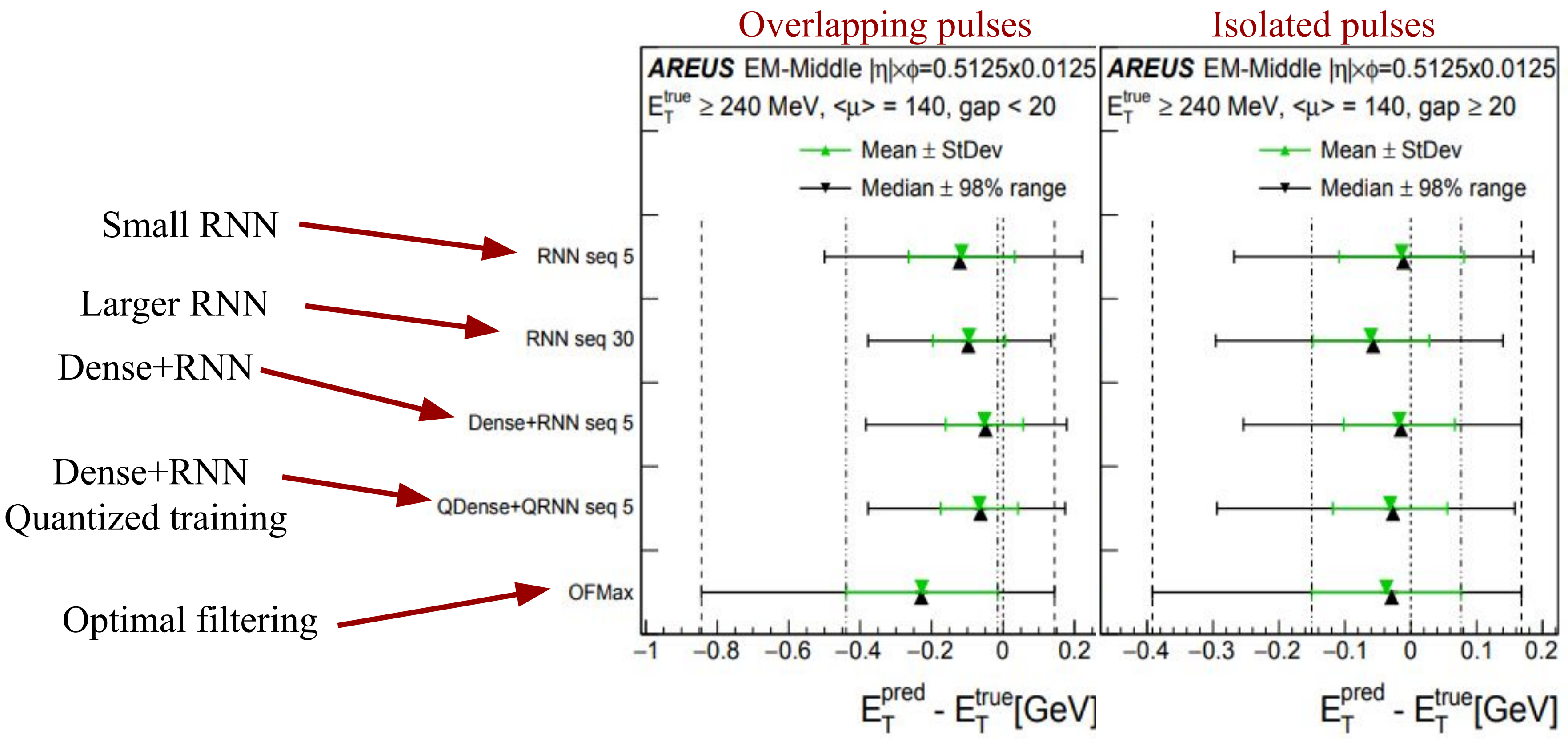
Simulation of the energie resolution in firmware as function of the number of bits



RNN performance (summary)

Lauri Laatu, PhD thesis (2023)

- Small RNNs (sequence length of 5 samples) can outperform OFMax overall
  - But not in all regions
  - Larger networks needed
- Several optimisation carried out to improve the performance
  - Keeping the network suitable for FPGA processing



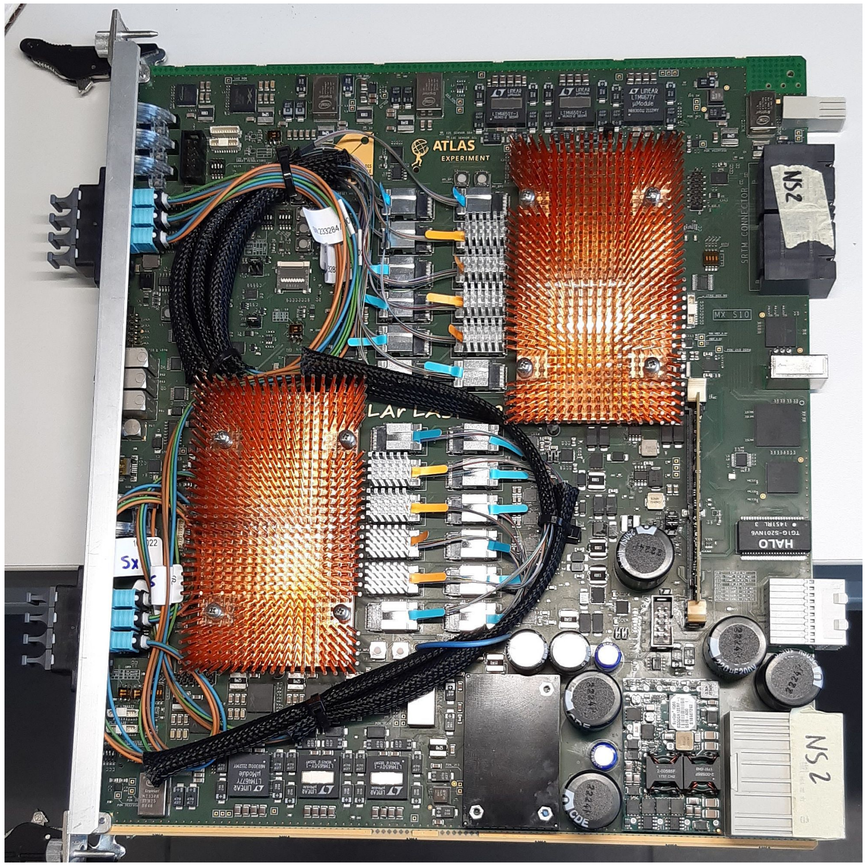
Firmware Implementation

- Implemented on Stratix 10 FPGA
  - Reference 1SG280HU1F50E2VG
  - Implementation on Agilex ongoing

- Challenges:
  - 384 channels per FPGA
  - 125 ns latency

- Preliminary implementation in HLS shows that LSTM is too large to fit
- Focus on Vanilla RNN
- Start with small RNN
  - 8 units and sequence length of 5
  - 89 parameters
  - 368 multiplications/accumulations (MAC) needed

LASP demonstrator board
Produced at CPPM



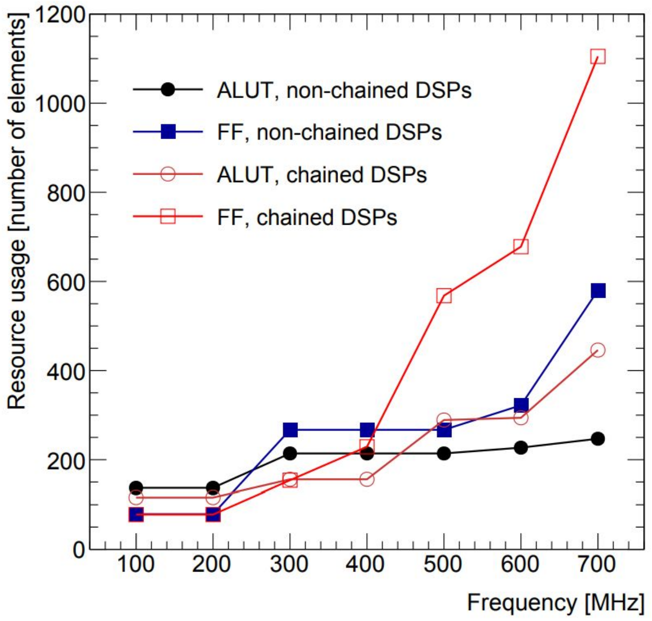
Implementation in HLS

JINST **18** (2023) P05017
Etienne Fortin, PhD thesis (2022)

- Optimisation needed to fit RNNs within resource and latency limitations
  - Impossible to fit 384 NNs in the FPGAs
  - Need to serialize (time multiplexing)
  - Need to go to high frequency
- Optimisation of vector/matrix multiplications
  - Most elementary operation inside neural networks
  - Naive C++: let HLS do it all
  - ACC37: Accumulate (sum) in DSPs by chaining them
  - ACC19: Accumulate in general logic elements (ALUT)

$$A.B = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_8 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_8 \end{bmatrix} = \sum_{i=0}^{7} a_i.b_i$$

	Implementation	ALUTs	FF	DSP
@100 MHz	C++ style	709	222	8
	ACC37	116	79	4
	ACC19	137	78	4



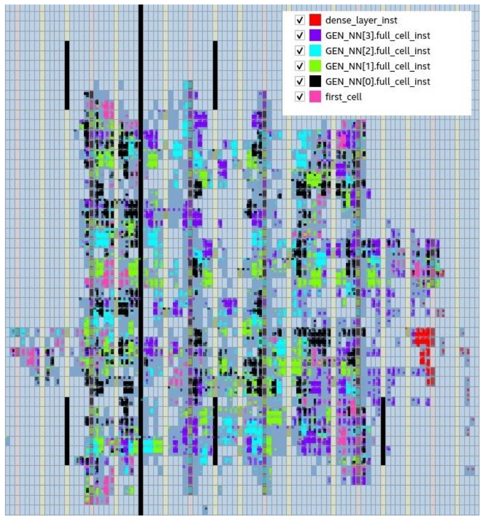
- Best strategy depends on frequency
  - Accumulate in DSP at low frequency
  - Accumulate in ALUT at high frequency
- Chaining DSPs at high frequency needs more logic than what is gained by performing sums inside DSPs

Implementation in VHDL

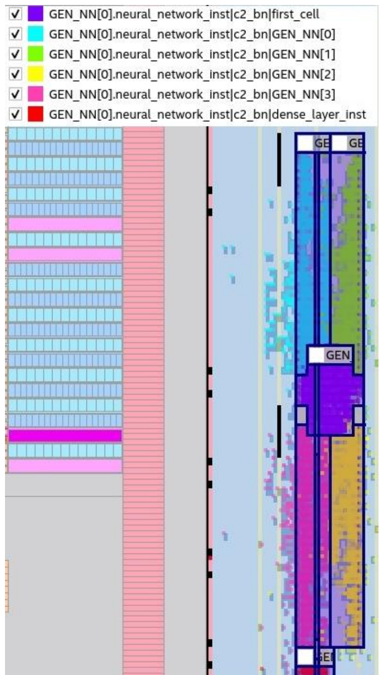
JINST **18** (2023) P05017
Etienne Fortin, PhD thesis (2022)

- HLS does not allow to reach the target frequency and resource usage
  - Increase of the needed logic (per network) and the latency as we add networks to the FPGA
- Move to VHDL for the final fine tuning
- Force placement of the RNN components
  - Allow to better tackle timing violations and improve the maximum reachable latency (FMax)
- Use incremental compilation
  - Freeze networks with no timing violations and recompile only the rest

HLS placement



VHDL forced placement

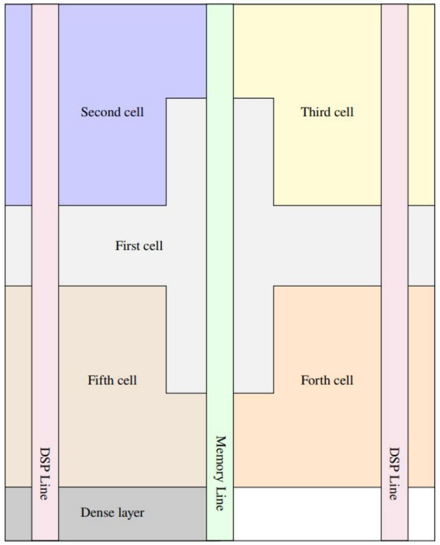


GEN_NN[0].neural_network_inst|c2_bn|first_cell
GEN_NN[0].neural_network_inst|c2_bn|GEN_NN[0]
GEN_NN[0].neural_network_inst|c2_bn|GEN_NN[1]
GEN_NN[0].neural_network_inst|c2_bn|GEN_NN[2]
GEN_NN[0].neural_network_inst|c2_bn|GEN_NN[3]
GEN_NN[0].neural_network_inst|c2_bn|dense_layer_inst
GEN

Optimized placement of RNN cells

First cells in the middle and connected to all cells (common computations done only in first cell)

Dense layer next to last cell



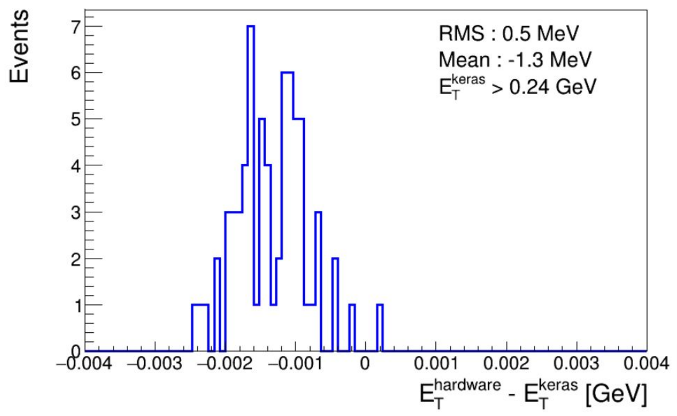
RNN firmware results

JINST **18** (2023) P05017
Nemer Chiedde, PhD thesis (2023)

*based on experience with the phase-I upgrade

	N networks x multiplexing	ALM	DSP	FMax	latency
Target	**384 channels**	**30%***	**70%***	**Multiplexing x 40 MHz**	**125 ns**
“Naive” HLS	384x1	226%	529%	-	322 ns
HLS optimized	37x10	90%	100%	393 MHz	277 ns
VHDL optimized	28x14	18%	66%	561 MHz	116 ns

- HLS allows fast development and optimisation
  - However less control on hardware specific implementation
- VHDL is needed to fine tune the design and fit the LAr requirements
- Vanilla RNN firmware produced and fits the requirements
  - Better performance expected with the Agilex FPGA
- Firmware tested on the hardware (Stratix 10 DevKit)
  - Extracted results match bit-by-bit the firmware simulation
  - Firmware resolution < 0.1% as expected from simulation



Conclusions

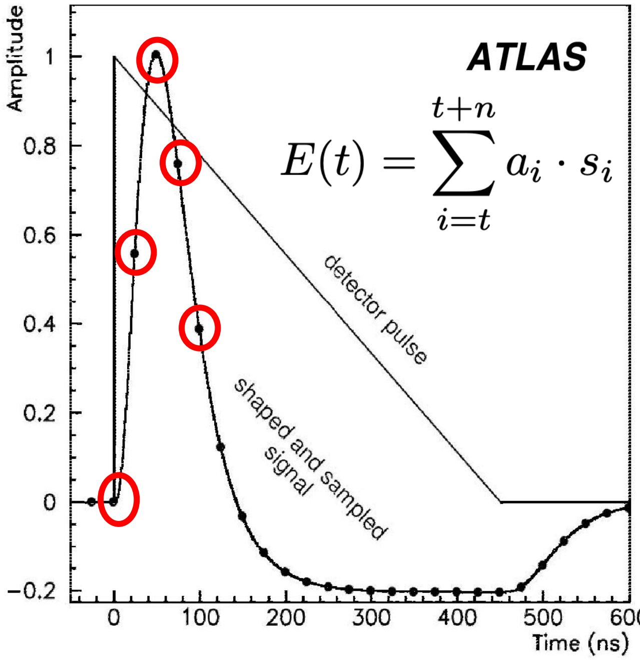
- Neural networks can outperform the optimal filtering algorithm for the energy reconstruction in the ATLAS LAr Calorimeter
  - Particularly in the region with overlap between multiple pulses
- Several optimisations carried out to improve the RNN performance while keeping minimal resource usage
  - The improvement on object reconstruction (electrons, photons) is ongoing

- Small Vanilla RNN implemented on Stratix 10 FPGAs
- HLS implementation allows very fast prototyping
  - Added support for both Vanilla RNNs and LSTMs on INTEL FPGAs to HLS4ML
  - HLS design did not fit the stringent resource and latency requirements
- Final implementation done in VHDL
  - Fits requirements and successfully tested on hardware
- Next steps is to implement larger networks in Agilex FPGAs

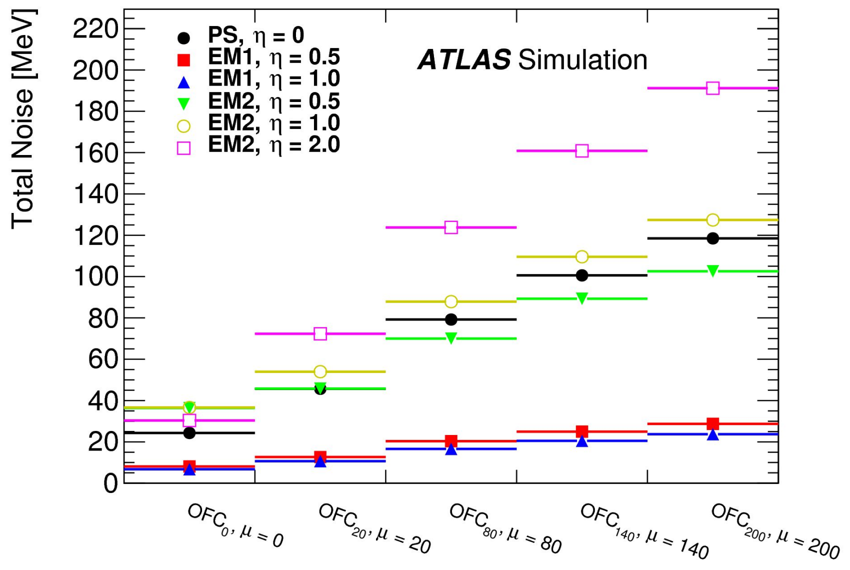
Backup

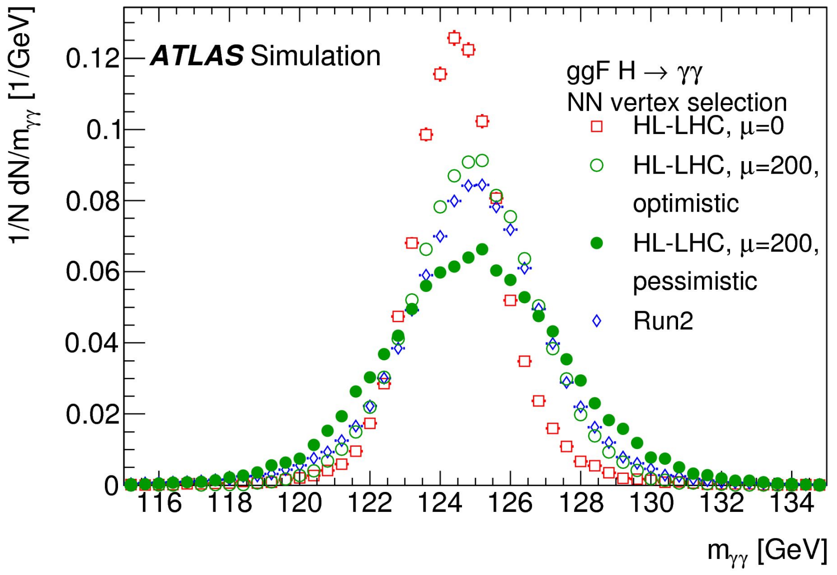
Energy reconstruction at the HL-LHC

- Energy reconstruction using optimal filtering
  - Weighted sum of sampled pulse amplitudes
- Increased noise due to increased pileup
  - Up to a factor of 2 with respect to Run 3
- About 30% degradation in $m_{\gamma\gamma}$ resolution
  - Better energy reconstruction algorithms needed
  - Neural networks are obvious candidates



ATL-COM-LARG-2017-030

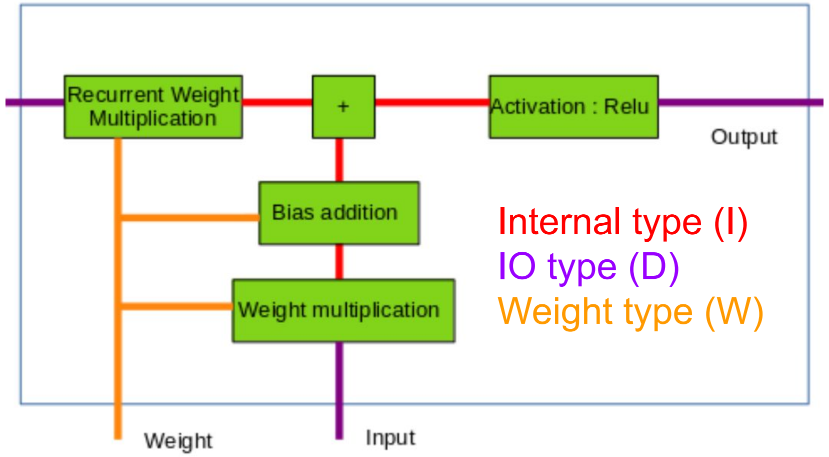


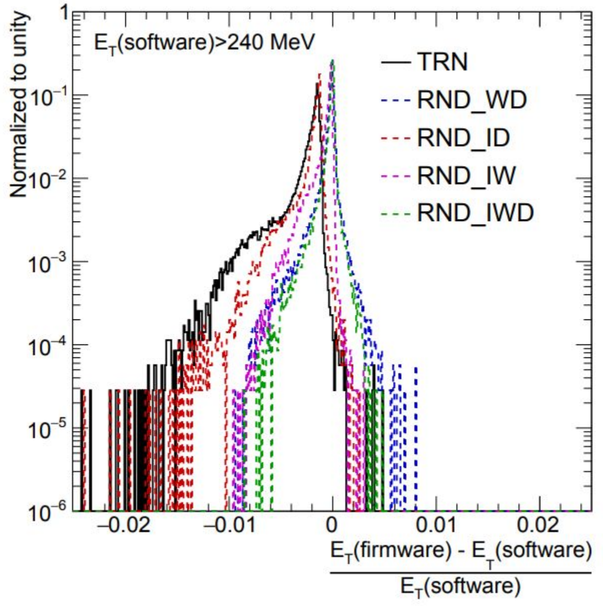


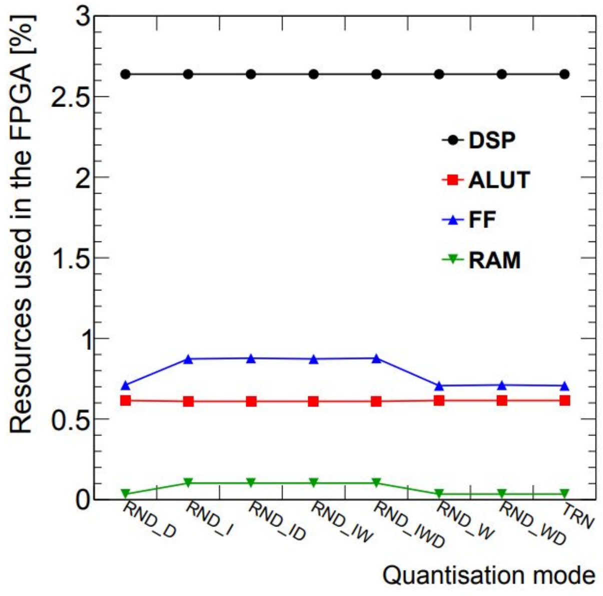
Rounding vs Truncation

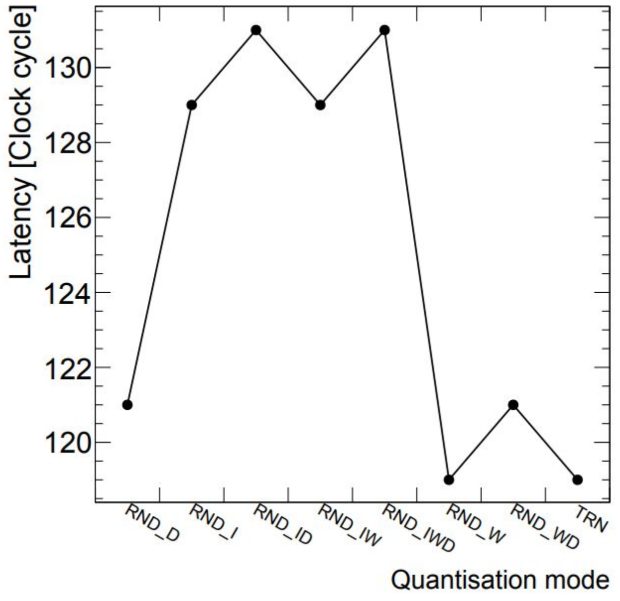
JINST **18** (2023) P05017
Etienne Fortin, PhD thesis (2022)

- 18 bits fixed point representation to match INTEL's DSP design
- Split neural network operations to 3 types
- Compromise between resolution and resource usage and latency
  - Truncation of IO and Internal types has small impact on energy resolution
  - Weight type rounded in software before loading
- Use truncation in the firmware


Recurrent Weight Multiplication
+
Activation : Relu
Output
Bias addition
Weight multiplication
Weight
Input
Internal type (I)
IO type (D)
Weight type (W)


Normalized to unity
E_T(software)>240 MeV
TRN
RND_WD
RND_ID
RND_IW
RND_IWD
E_T(firmware) - E_T(software) / E_T(software)


Resources used in the FPGA [%]
DSP
ALUT
FF
RAM
RND_D
RND_I
RND_ID
RND_IW
RND_IWD
RND_W
RND_WD
TRN
Quantisation mode


Latency [Clock cycle]
RND_D
RND_I
RND_ID
RND_IW
RND_IWD
RND_W
RND_WD
TRN
Quantisation mode