Machine Learning on FPGAs for Real-Time Processing for the ATLAS Liquid Argon Calorimeter

Nemer CHIEDDE
on behalf of the ATLAS Liquid Argon Calorimeter Group

DIS2023: XXX International Workshop on Deep-Inelastic Scattering and Related Subjects on 28 March 2023

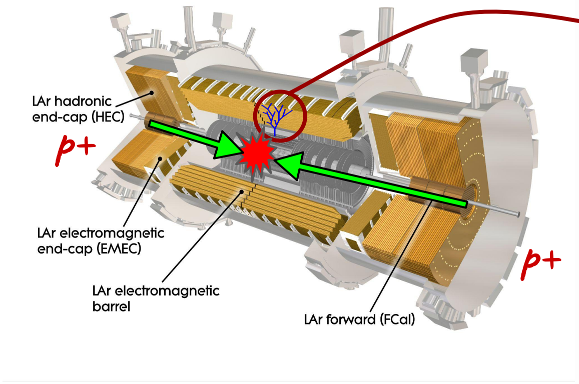
Content

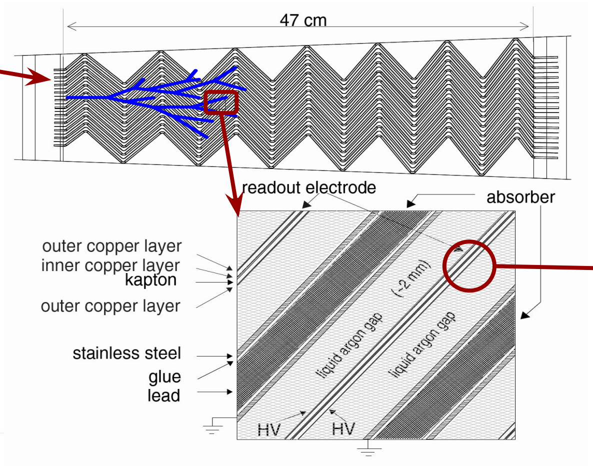
ENERGY RECONSTRUCTION IN THE LAR CALORIMETER

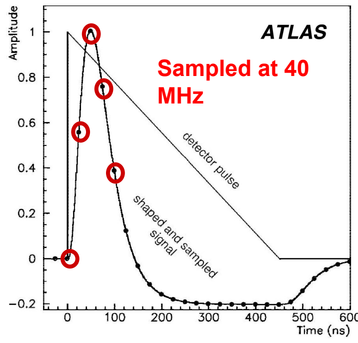
- The ATLAS liquid argon calorimeter (LAr) exploits the ionization signal to measure the energy of electrons and photons
  - Calorimeter with ~182,000 cells

- Bipolar pulse shape (total length up to 750 ns, 30 bunch crossings)
  - Sampled and digitized at 40 MHz

- LAr processing uses optimal filtering algorithms to compute the deposited energy
  - Maximum finder used to identify the deposit time (OFMax)
  - Use of five samples around the peak of the pulse


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



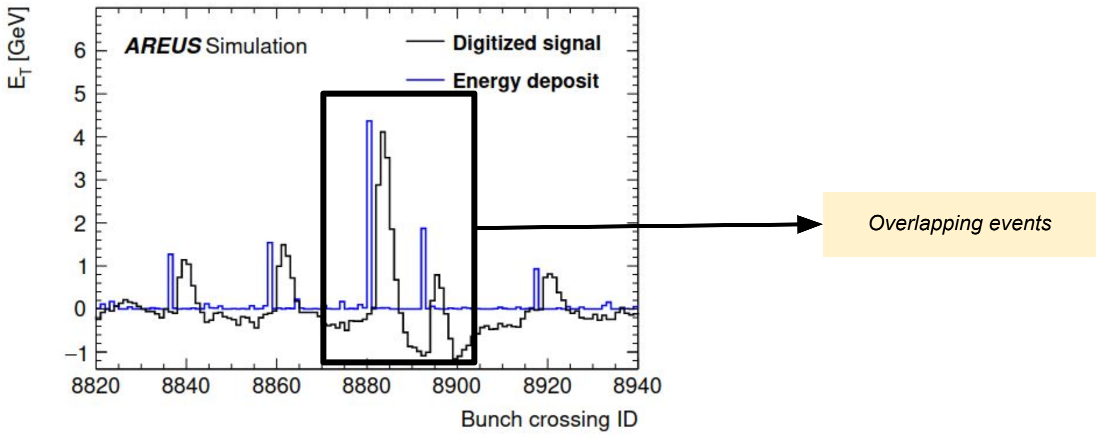


Energy reconstruction under HL-LHC conditions

- As part of the HL-LHC upgrade:
  - The luminosity will be increased to better understand rare processes
  - More p-p collisions per bunch crossing (pileup): ~200 compared to the current ~40

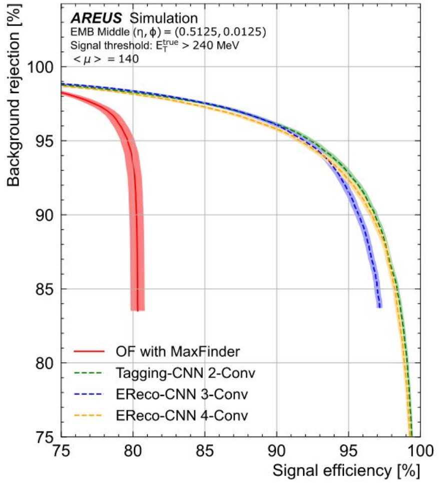
- Pulses might overlap due to high pileup which distorts the bipolar pulse profile of successive pulses
  - The current model (OFMax) was not developed to work under these conditions

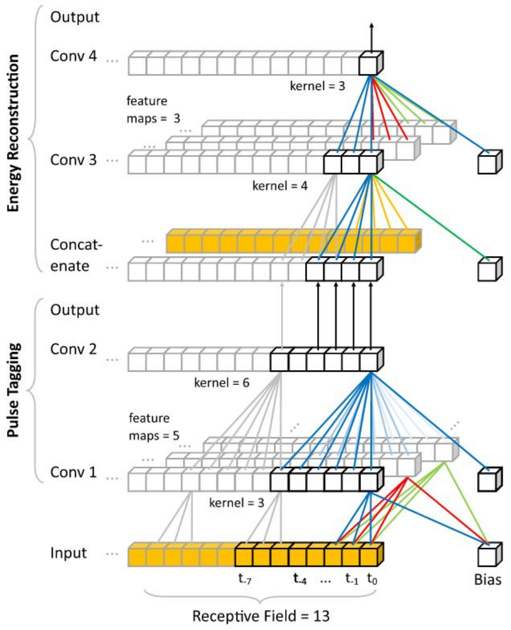
- Phase II electronics will have higher computational capacity
  - Possibility to implement a neural network based algorithm in FPGAs to reconstruct the energies deposited in the LAr calorimeter



CNN ARCHITECTURE AND NETWORK SIZE

- 1D Convolution Neural Networks (CNN) designed to process time series regression
- CNN architecture divided into two sub-networks:
  - Identification of the energy above 3σ of the electronic noise, corresponding to 240 MeV.
  - Reconstruction of the energy deposited in each cell of the calorimeter.
- 3-Conv has 5 samples in the peak, 23 in the past with 3 total layers
- 4-Conv has 5 samples in the peak, 8 in the past with 4 total layers
- The maximum finder achieves a maximum signal efficiency of about 80%, while the tagging CNN reaches efficiencies well above 90%





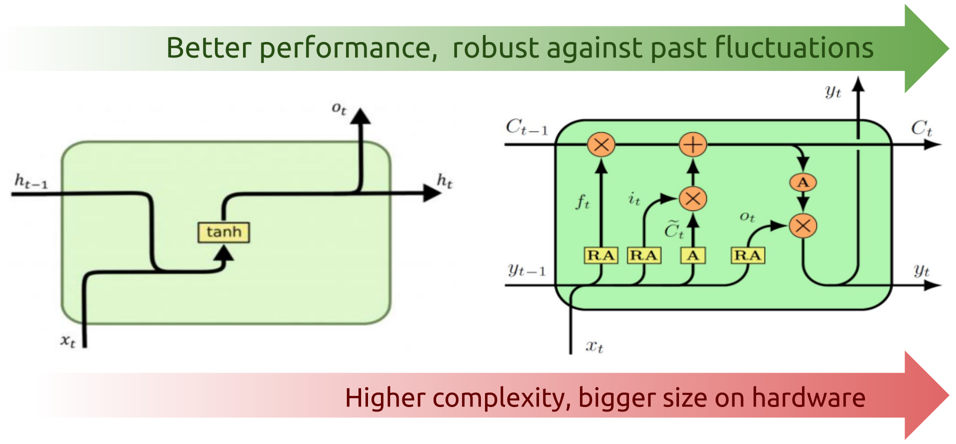
RNN ARCHITECTURE AND NETWORK SIZE

- Recurrent Neural Networks (RNN) are designed to process time series data
  - RNNs consist of neural network layers that combine new temporal input with previously processed state

- Long short-term memory (LSTM) and Vanilla-RNN are the RNNs selected to verify the feasibility of the hardware implementation

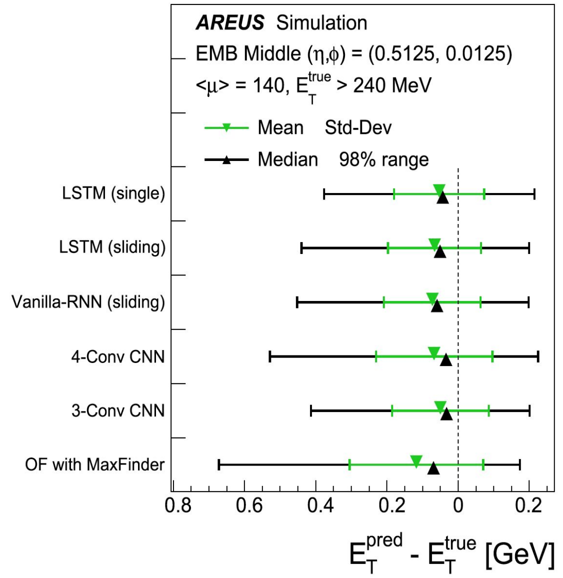
  - The LSTM cell is the most complex RNN, and, consequently, has the **highest accuracy.** It needs two activation functions (hyperbolic tangent and sigmoid). **Uses a lot of FPGA resources**

  - The Vanilla-RNN or Simple-RNN cell is the most straightforward.It consists of a single activation (ReLU) and **requires less resources than LSTM. Less effective than LSTM**, but **still acceptable for our purposes**



NN PERFORMANCE AND NETWORK SIZE

- All neural networks have better energy resolution and mean closer to zero compared to the actual model (**OFMax**)
- LSTM network has the best performance among all evaluated neural networks, however it is too large to fit on an FPGA
- CNNs and Vanilla RNNs have less parameters and perform adequately



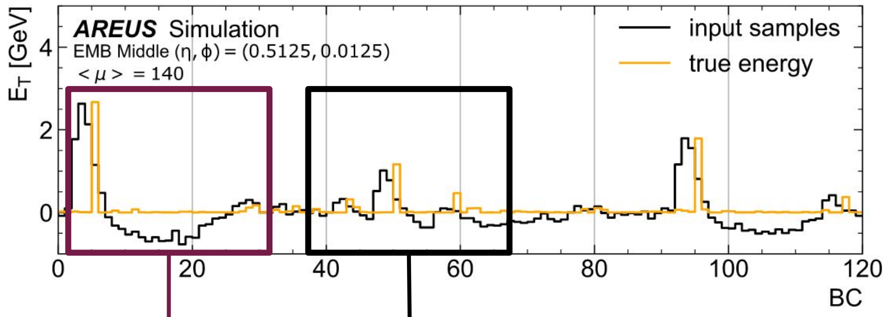
Algorithm	**LSTM** (single)	**LSTM** (sliding)	**Vanilla** (sliding)	**CNN** (3-conv)	**CNN** (4-conv)	**Optimal Filtering**
Number of parameters	491	491	89	94	88	5
MAC units	480	2360	368	87	78	5

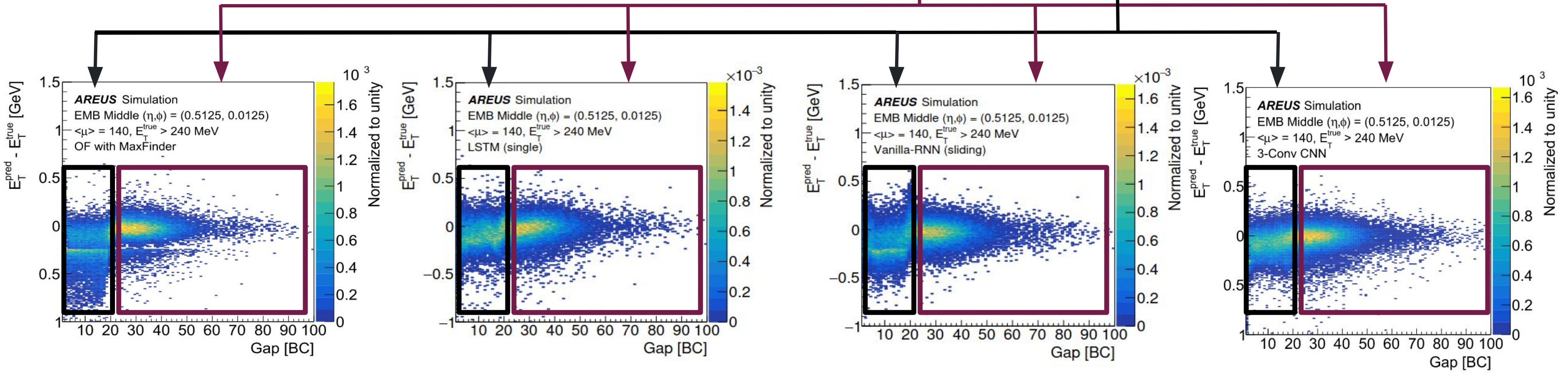
RNN resolution as a function of gap to previous energy deposit in BCs

- Overlapping pulses decrease the performance of the energy reconstruction by **OFMax**

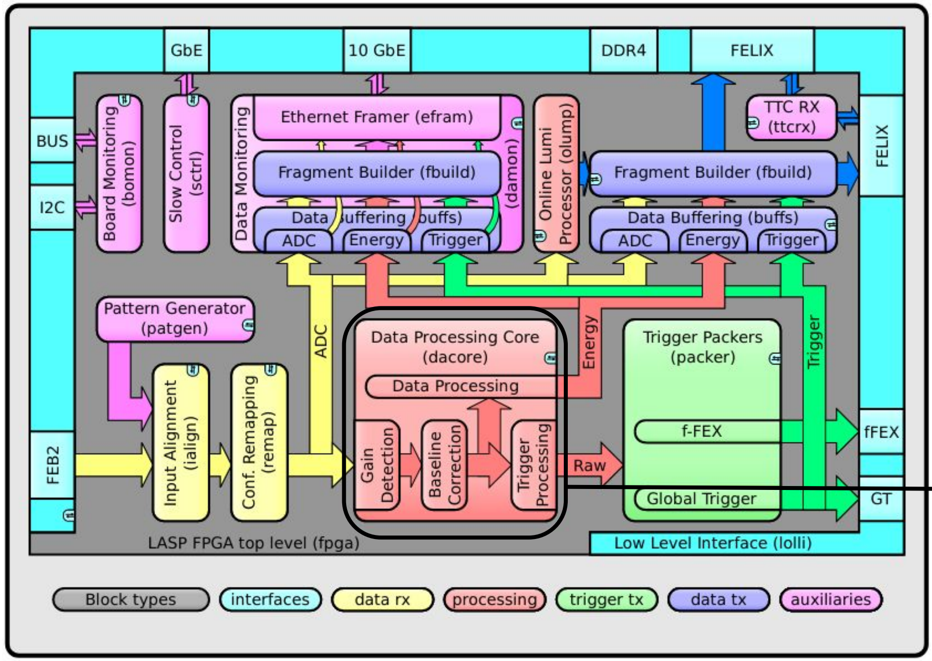
- Performance in the overlap region depends on the number of samples that are used from past events

- All NNs tested are more efficient with overlapping pulses
  - Published results (**link**)



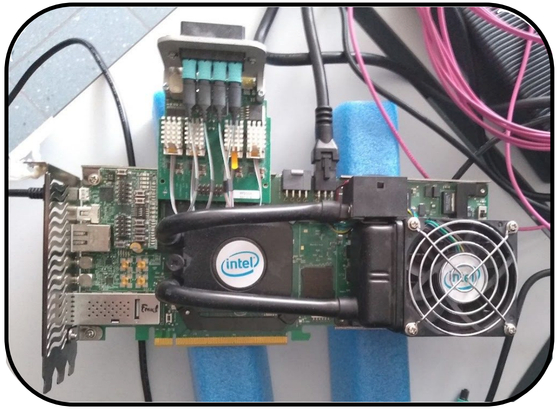


Firmware implementation: LASP board



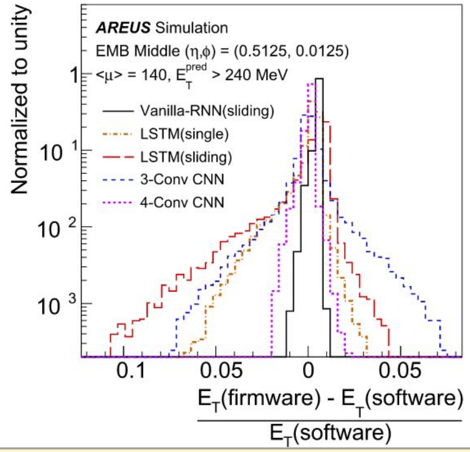
- The LAr Signal Processor (LASP) board will contain two latest generation INTEL Agilex FPGAs
- **3** front-end boards must be processed per LASP
  - **384** channels
  - The latency must be less than 125 ns.
- Each channel operates at 40 MHz
- All channels must be processed simultaneously by the NNs

Replace OFMax by the implementation of neural networks



- The tests are performed on a Stratix 10 development kit as the Agilex development kit is not available yet
- In order to implement in hardware, it is necessary to use specific tool such as HLS or VHDL
- RNN are developed in HLS and then VHDL, while CNN in VHDL

FIRMWARE IMPLEMENTATION: FPGA



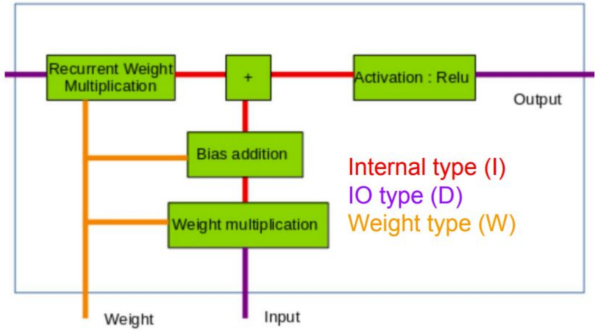
- Resolution between software and firmware output is O(1%)
- Necessary to use multiplexing to compute several networks simultaneously
- Using multiplexing, we are able almost meet the requirements for one instance of the network per fpga
  - Some optimizations still needed
- Maximum clock frequency and channels:
  - Vanilla RNN: 512 channels and 600 MHz for 15x multiplexing
  - CNN: 384 channels and 480 MHz for 12x multiplexing
- These implementations are expected to reach even less resource usage, shorter latency, and higher clocking frequency

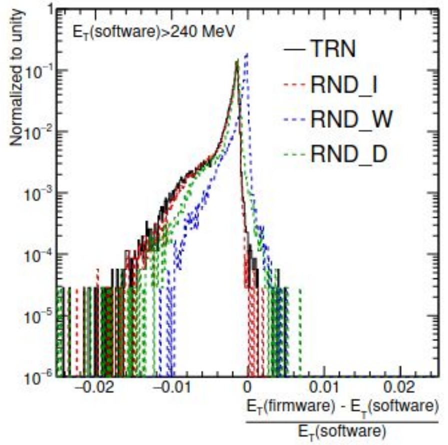
One instance per FPGA

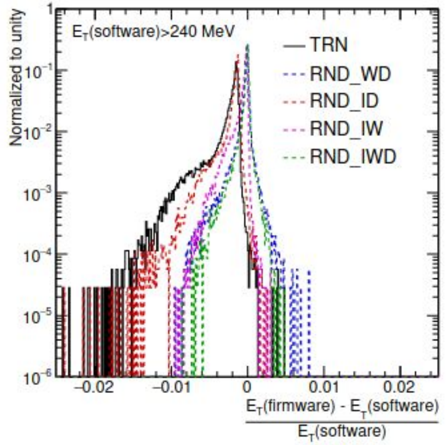
	Language	Number of channel	ALM [%]	DSPs [%]	Latency [ns]	Max. Frequency [MHz]	Multiplexing
Not Multiplexed	3-conv CNN	-	0.6	0.8	62	493	-
	4-conv CNN	-	0.6	0.7	58	480	-
	Vanilla RNN	-	1.4	0.6	206	641	-
	LSTM (sliding)	-	7.5	12.8	363	517	-
Multiplexed	3-conv CNN	516	2.3	0.8	125	487	12
	4-conv CNN	660	1.8	0.7	150	423	12
	Vanilla RNN	576	0.6	2.6	120	640	15

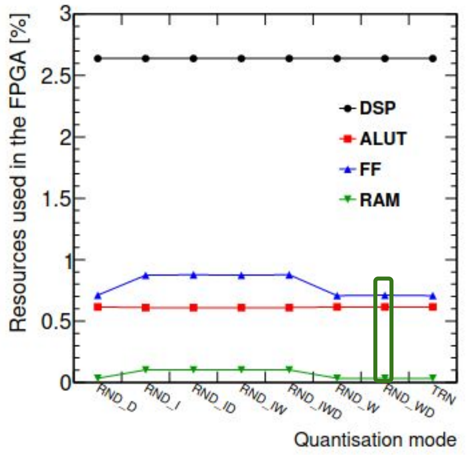
Optimization of arithmetic operations

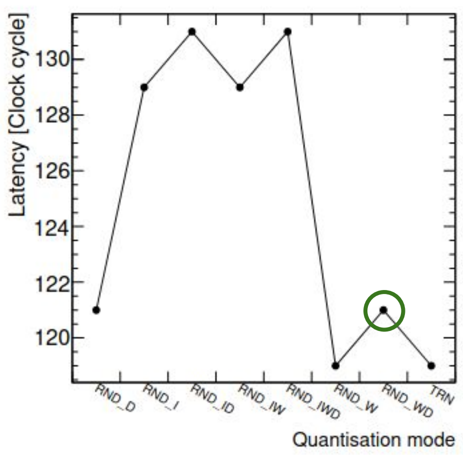
- To optimize the arithmetic operations, a mix of quantization procedures are used for different data categories:
  - Internal, Input/Output and Weights
- Rounding (RND) of the weights does not require any additional resources in the FPGA
  - Rounded weights can be loaded into the FPGA
- Truncation of I/O, internal and weights types leads to a significant loss in resolution
  - RND_IWD (0.07%)
  - **RND_WD (0.09%)**
  - RND_W (0.12%)
  - Truncation (TRN) (0.2%)











VHDL implementation of the Vanilla RNN

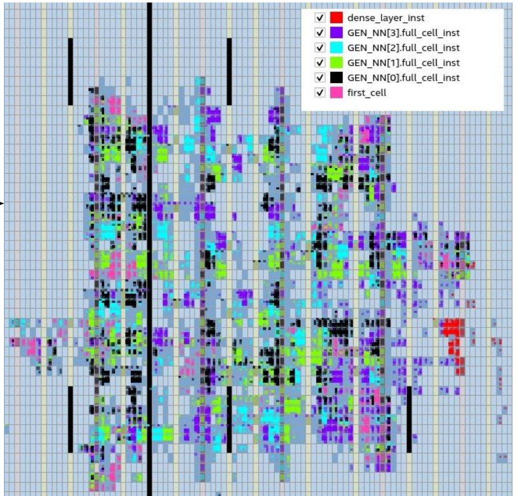
- HLS does not achieve the target frequency and resource utilization when several instances of the NN are implemented in one FPGA
    - Increased Adaptive Logic Module (ALM) resources and reduced the maximum frequency (FMax) when we increase the number of network instances

- Force the placement of RNN components
    - Allow better handling of timing violations and improve FMax
    - Defining a placement area in HLS is very complex
    - Change to VHDL for final setup

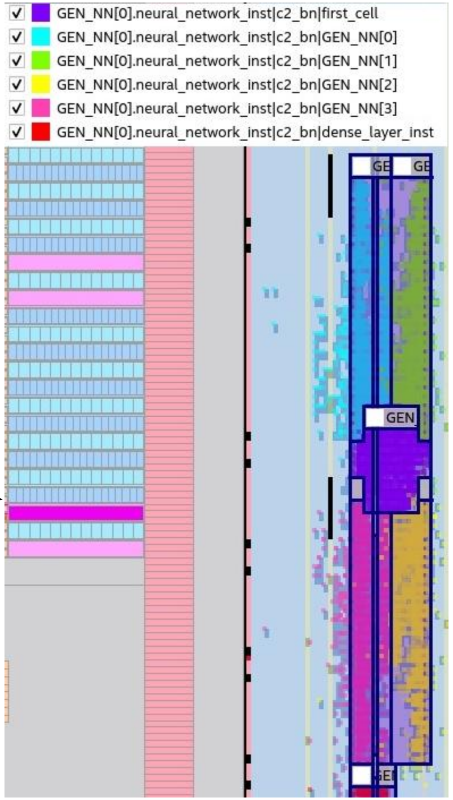
- Use incremental compilation

    Keep the network instances that do not present timing violations and recompile only the rest

HLS placement


✓ 🟥 dense_layer_inst
✓ 🟪 GEN_NN[3].full_cell_inst
✓ 🟦 GEN_NN[2].full_cell_inst
✓ 🟩 GEN_NN[1].full_cell_inst
✓ ⬛ GEN_NN[0].full_cell_inst
✓ 🟪 first_cell

VHDL 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

FPGA firmware simulation results with Vanilla-RNN

- VHDL is needed to refine the design and meet the requirements of LAr
- The produced Vanilla-RNN firmware fits the resource limitations estimated by the LAr collaboration
  - Hard to meet strict specifications with HLS
- Incremental compilation with forced placement helps to avoid timing issues, increase the frequency

Stratix10 1SG280HU2F50E2VG	Language	Number of channel	*ALM [%]	** DSPs [%]	Latency [ns]	Max. Frequency [MHz]	Multiplexing
Results	HLS optimized	370	23%	100%	302	414	10
	VHDL optimized	392	18%	66%	121	561	14
Specification	-	384	max 30%	max 70%	max 125		

* Adaptive Logic Module (ALM)
** Digital Signal Processing (DSP)

HLS4ML: High Level Synthesis for Machine Learning

- HLS4ML is an open source software designed to facilitate the implementation of AI algorithms on FPGAs

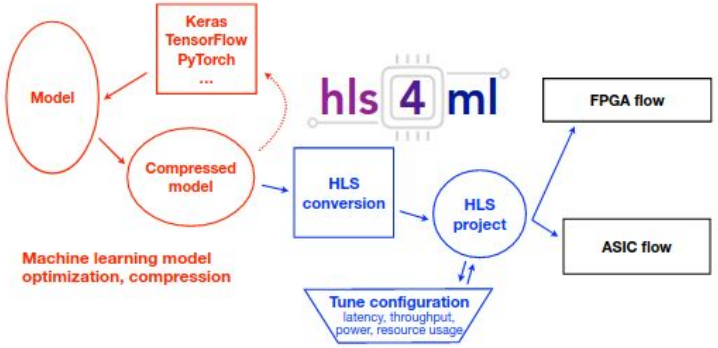
- Automatically performs the task of translating a trained NN, specified by the model architecture, weights and bias, into firmware for a specific hardware

- Includes implementation of common elements (layers, activation functions, binary NN, ...)

- RNNs in quartus HLS are now implemented in HLS4ML
    - Supports many of the optimizations that were done for the RNNs

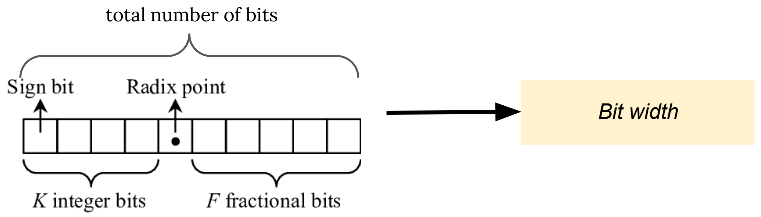
- **RNN** for Quartus github (link)

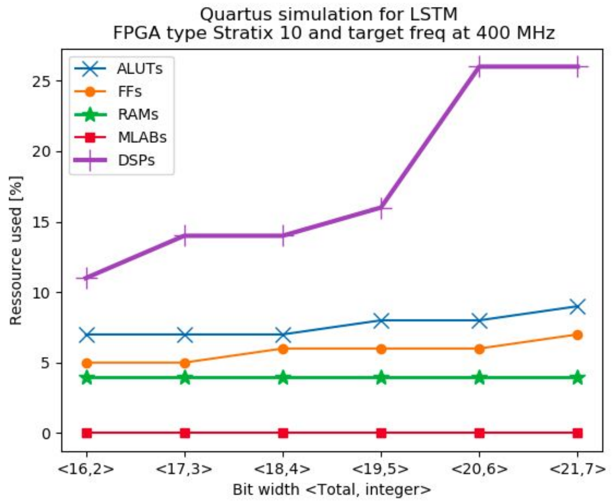
HLS4ML can be used to easily adapt NNs and optimize parameters for implementation on FPGAs

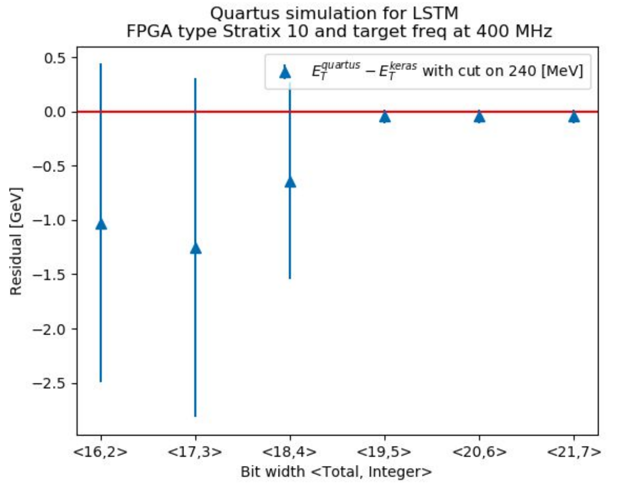


HLS4ML ADAPTABILITY

- RNNs in quartus HLS are now implemented and validated in HLS4ML
    - Supports many of the optimizations that were done for the RNNs
- Provides a number of configurable parameters which can help the user explore and customize, for example:
    - FPGA type
    - Look-up table (LUT) size
    - Clock period
    - Backend (Vivado, Vivado Accelerator and Quartus)
- Used to optimize the parameters for implementation on FPGAs
    - Bit width can be optimized per data type selected
    - Different activation functions can be applied per layer
    - LUT precision can be fixed independently
    - Utilise symmetry properties for softsign, sigmoid, and tanh to give higher precision for the same resource usage







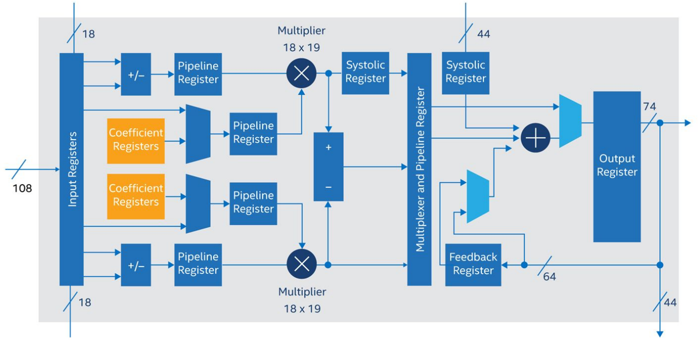
General status and perspectives

- RNNs and CNNs outperform the optimal filtering algorithm for energy reconstruction in the ATLAS LAr calorimeter
  - Especially in the overlapping region between multiple pulses
- All networks are designed to minimize resource usage while maintaining performance
- Vanilla-RNN is a strong candidate that can satisfy the strict requirements of the LASP firmware
- HLS is a powerful tool for fast prototyping while implementation in VHDL is needed to refine the firmware in case of stringent requirements
- HLS4ML can be used to easily adapt the NN and optimize the parameters for implementation on FPGAs
  - LSTM and RNN are implemented in HLS4ML for quartus
- Hardware tests (INTEL DevKits) have started and show good results
- Paper published/submitted:
  - Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters (link)
  - Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment (link)

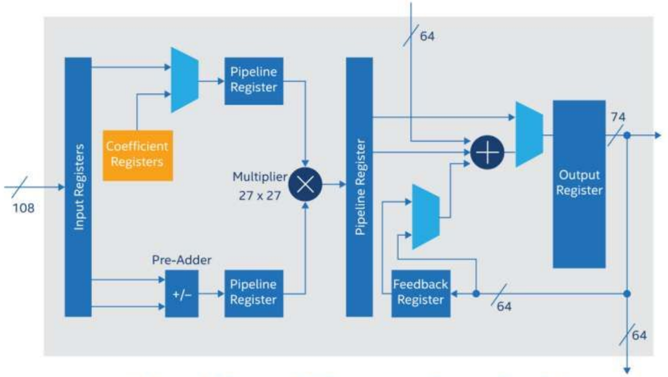
Backup

DSP modes and multiplication resource impact

- The fixed point representation can directly affects the resource usage in the FPGA.
- Dedicated component (Digital Signal Processing) inside the FPGA perform the scalar multiplications
- The DSP can work in 3 differents mode for Stratix 10 and Agilex:
  - One block DSP for 32×32 bits multiplication in the floating-point representation
  - One block DSP for 27×27 bits multiplication in the fixed-point representation
  - One block DSP do **two simultaneous 19×18 bits multiplications** in the fixed-point representation
- The third mode allows doubling the available dedicated multiplication resources on the FPGA
  - Using 16 bits for weights and 19 bits for the rest optimizes FPGA DSP resources while maintaining firmware calculation resolution (<0.1%)



Intel® Stratix® 10 Device DSP Block: Standard-Precision Fixed Point



Intel® Stratix® 10 Device DSP Block: High-Precision Fixed Point

Matrix multiplication with and without chained DSPs

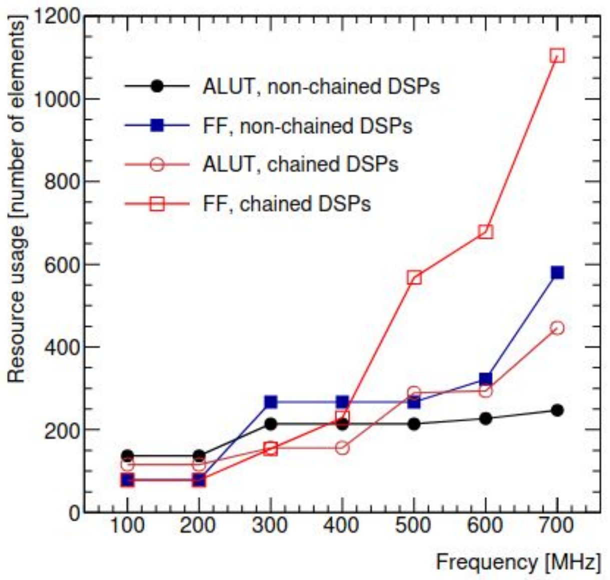
- DSP can sum the result of two multiplications internally and has a component with an external input adder

  The first DSP's output can be used in the second DSP's additional adder to sum their outputs

  - Needs to synchronize the DSPs
  - Extra registers are required to delay the results

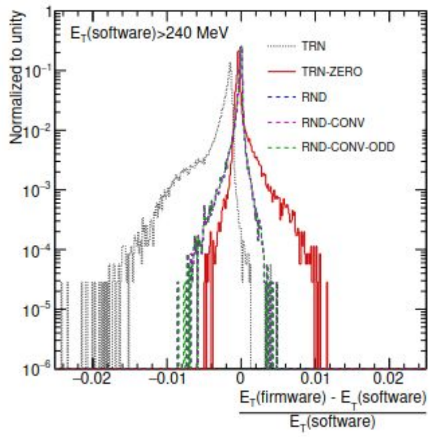
- The chained mode is advantageous below 450 MHz

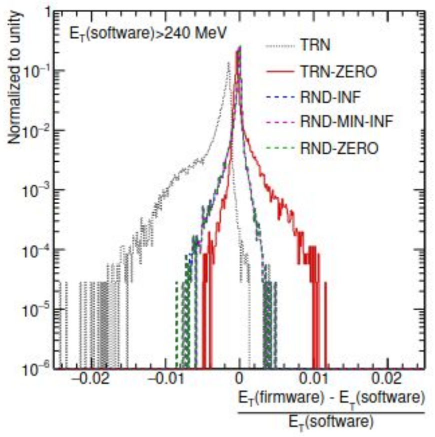
  - MLAB frequency is limited to 450 MHz in read-write mode required for register (FIFO) implementation.

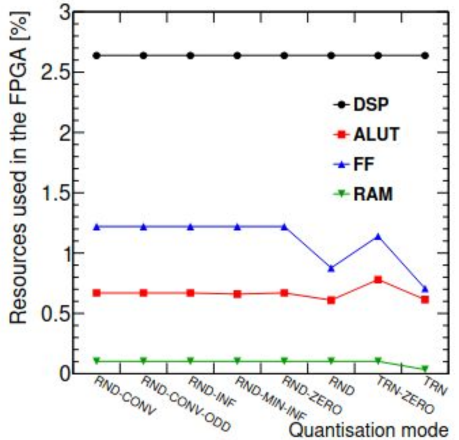


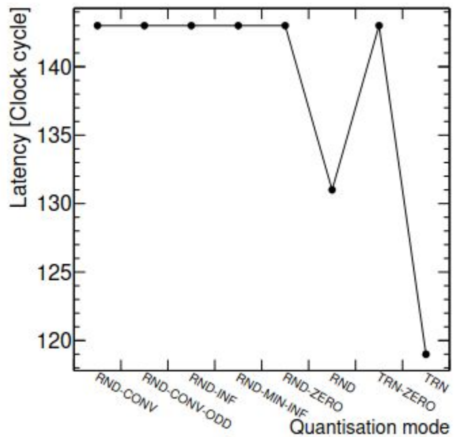
Arithmetic operations effects

- Two types of quantization are implemented in the Intel HLS compilation:
  - Truncation (TRN)
  - Rounding (RND)
- TRN quantization modes have the worst effect on the calculated transverse energy
- RND mode gives a good compromise among resource usage, latency, and resolution.


E_T(software)>240 MeV
TRN
TRN-ZERO
RND
RND-CONV
RND-CONV-ODD
Normalized to unity
$\frac{E_T(\text{firmware}) - E_T(\text{software})}{E_T(\text{software})}$


E_T(software)>240 MeV
TRN
TRN-ZERO
RND-INF
RND-MIN-INF
RND-ZERO
Normalized to unity
$\frac{E_T(\text{firmware}) - E_T(\text{software})}{E_T(\text{software})}$


Resources used in the FPGA [%]
DSP
ALUT
FF
RAM
RND-CONV
RND-CONV-ODD
RND-INF
RND-MIN-INF
RND-ZERO
RND
TRN-ZERO
TRN
Quantisation mode


Latency [Clock cycle]
RND-CONV
RND-CONV-ODD
RND-INF
RND-MIN-INF
RND-ZERO
RND
TRN-ZERO
TRN
Quantisation mode