A **Quartus** backend for **hls4ml**

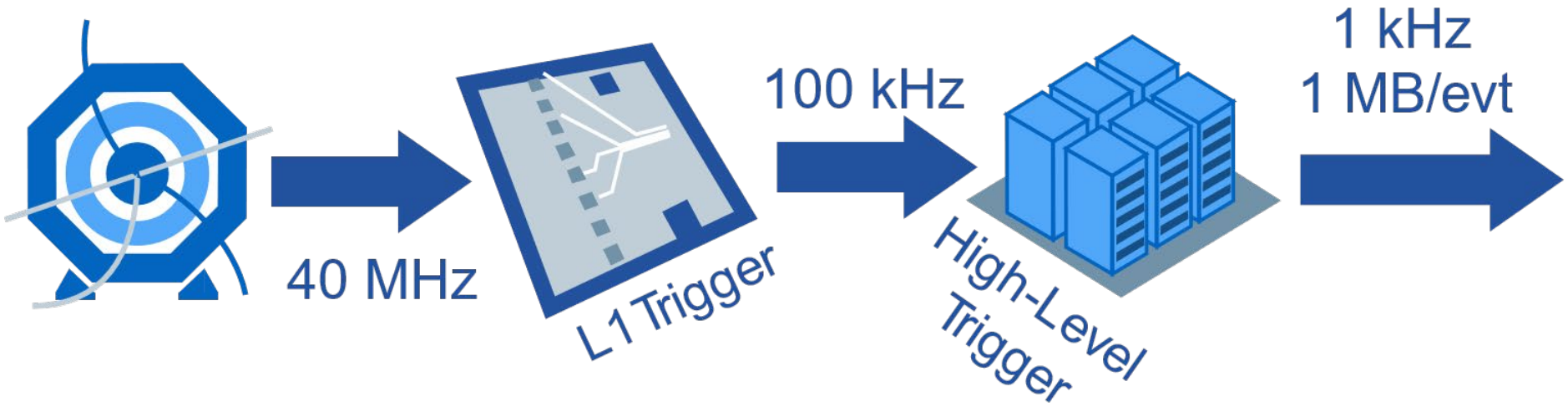
Deploying low-latency Neural Networks on Intel FPGAs

Hamza Javed

Fast Machine Learning for Science Workshop – November 30, 2020

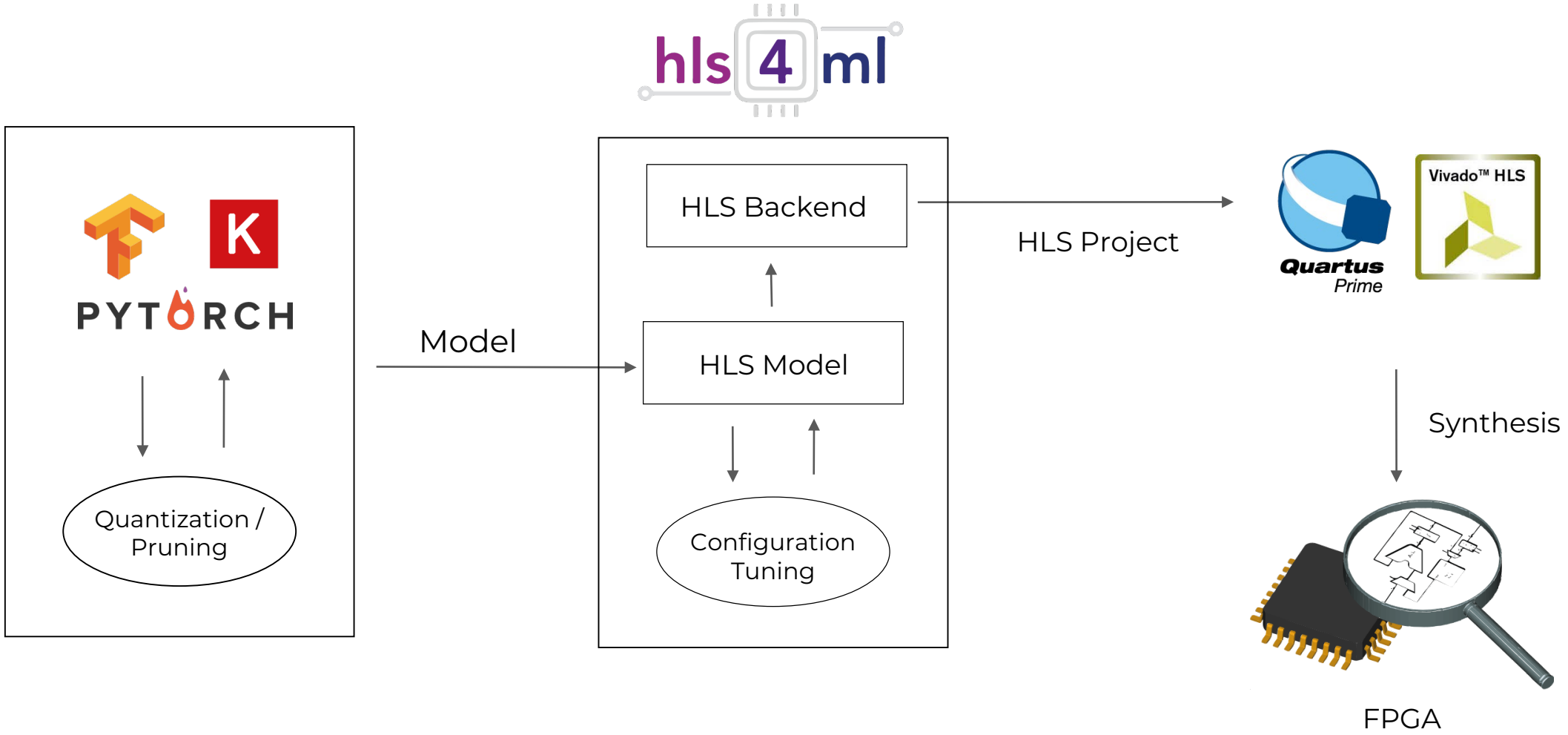
erc CERN mPP hls4ml

Problem



- Ultra low latency.
- High throughput

Approach


hls 4 ml
PYTORCH
K
Quantization / Pruning
Model
HLS Backend
HLS Model
Configuration Tuning
HLS Project
Quartus Prime
Vivado™ HLS
Synthesis
FPGA

Quartus Backend

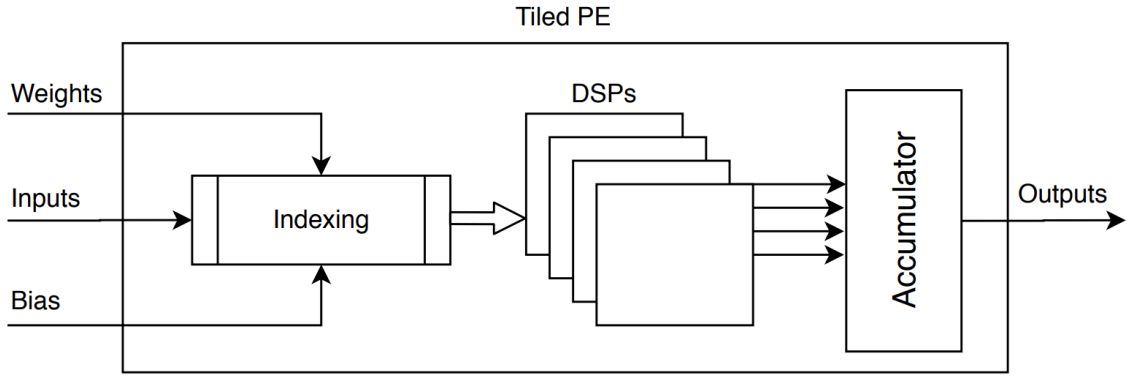
- Adopts layer architectures from the Xilinx backend.
- Parallelism is controlled via reuse factor while Quantization is handled through internal fixed-point datatypes.

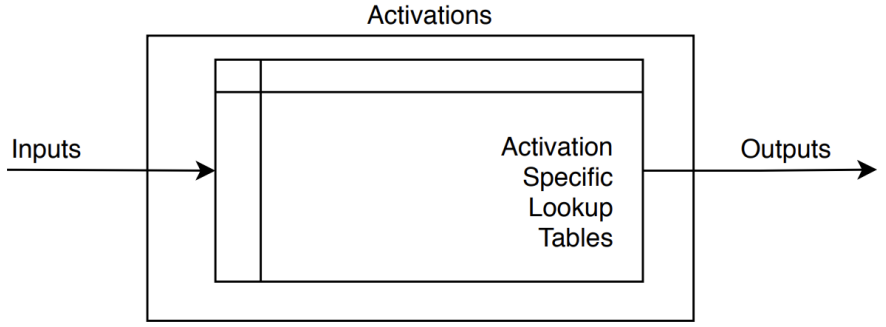
Supported Layers

Layer	Vivado	Quartus
Dense	✓	✓
Sparse Dense	✓	✓
RNN	✓	✓
GRU	✓	✓
LSTM	✓	✓
CONV1D	✓	✗
CONV2D	✓	✗
POOL	✓	✗

Base-Arch (Accelerator Templates)

- A tiled-PE is used to perform matrix multiplication operations within layers.
- Lookup tables are used to approximate mathematical functions.





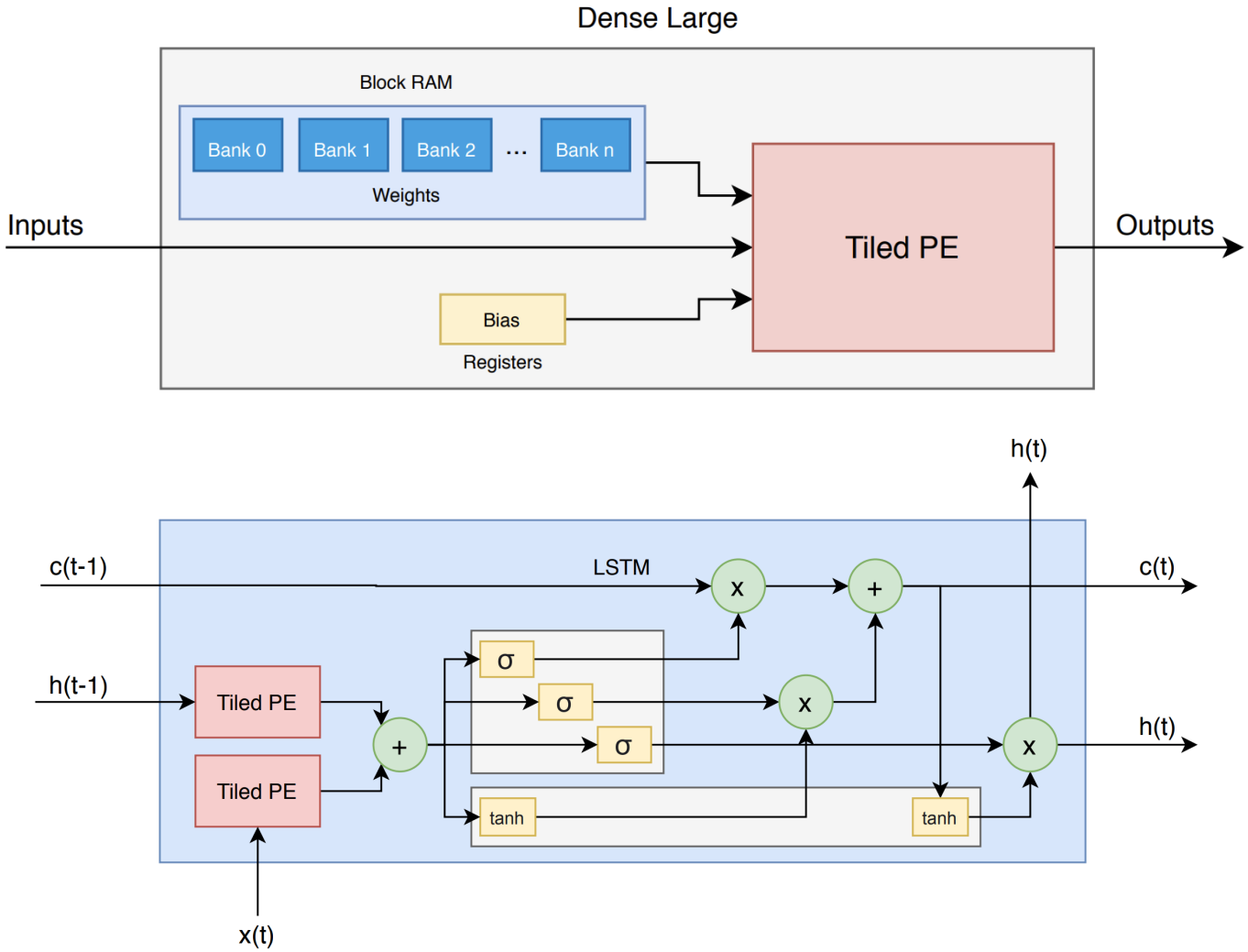
DL Layers

Layer	Weights storage	Composition
Dense Latency	Registers	Tiled-PE
Dense Large	Block RAM	Tiled-PE
Sparse Dense*	Registers (Sparsified with COO)	Tiled-PE with online index computations
RNN	Block RAM	Tiled-PE , Activation Block
GRU	Block RAM	Tiled-PE , Activation Blocks
LSTM	Block RAM	Tiled-PE , Activation Blocks

*partially supported for narrow networks.

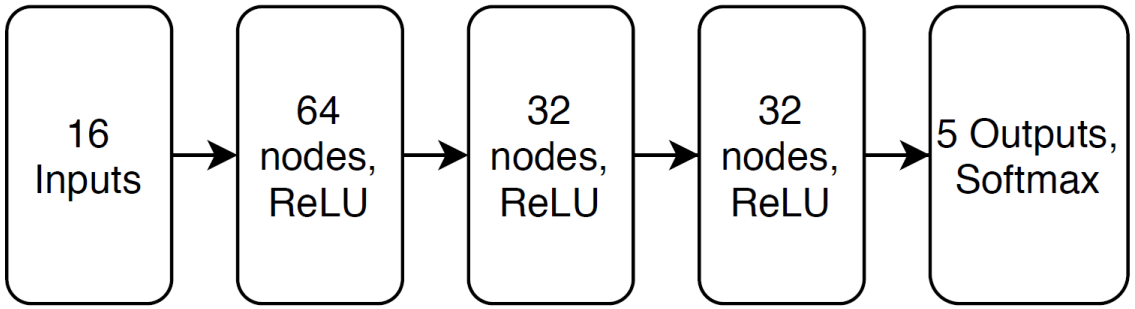
All of the Quartus layers include optimizations for binary/ternary computations.

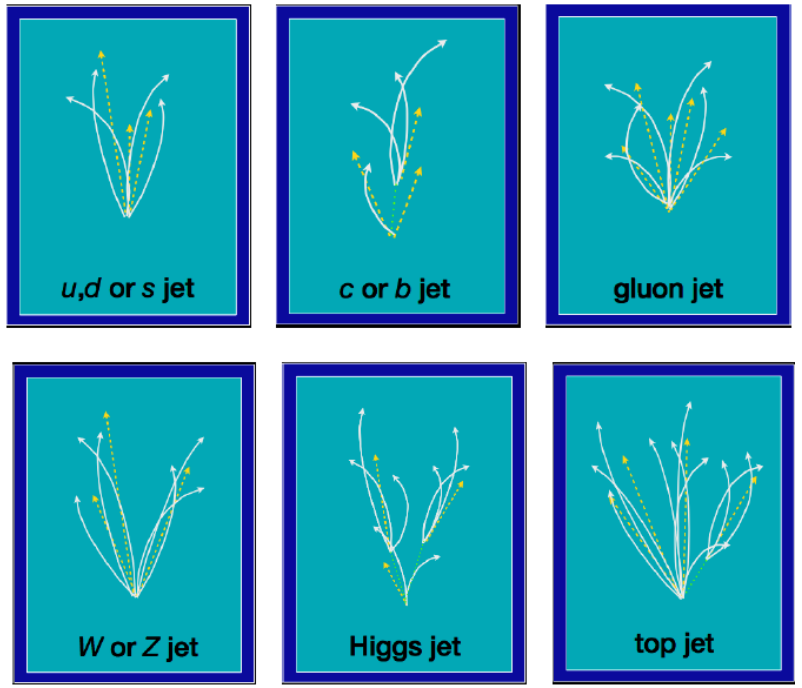
DL Layers


Dense Large
Block RAM
Bank 0
Bank 1
Bank 2
...
Bank n
Weights
Inputs
Tiled PE
Outputs
Bias
Registers
h(t)
c(t-1)
LSTM
c(t)
h(t-1)
Tiled PE
Tiled PE
σ
σ
σ
tanh
tanh
h(t)
x(t)

Test Case

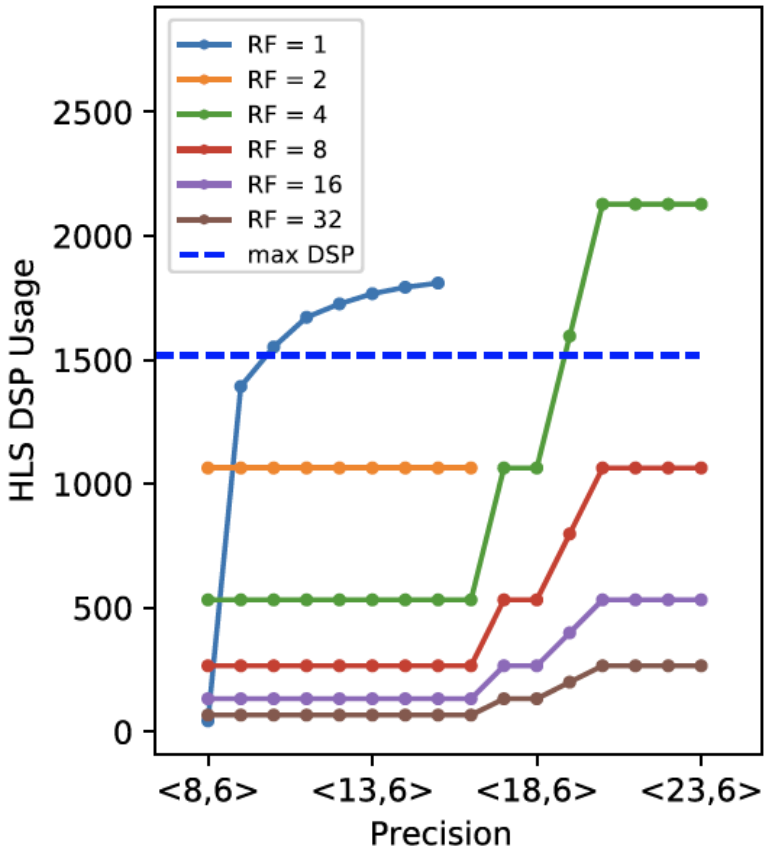
- A Deep Neural Network trained for jet classification
- Five outputs corresponding to each of the **g**, **q**, **w**, **z** & **t** jet.

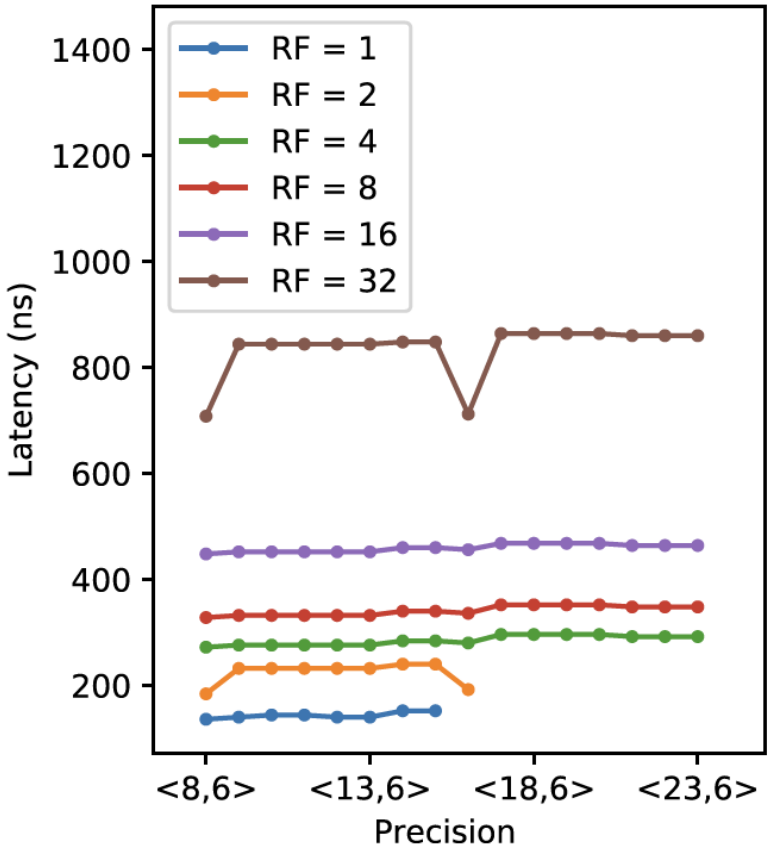




[1] Javier Duarte et al., Fast inference of deep neural networks in FPGAs for particle physics, 2018.

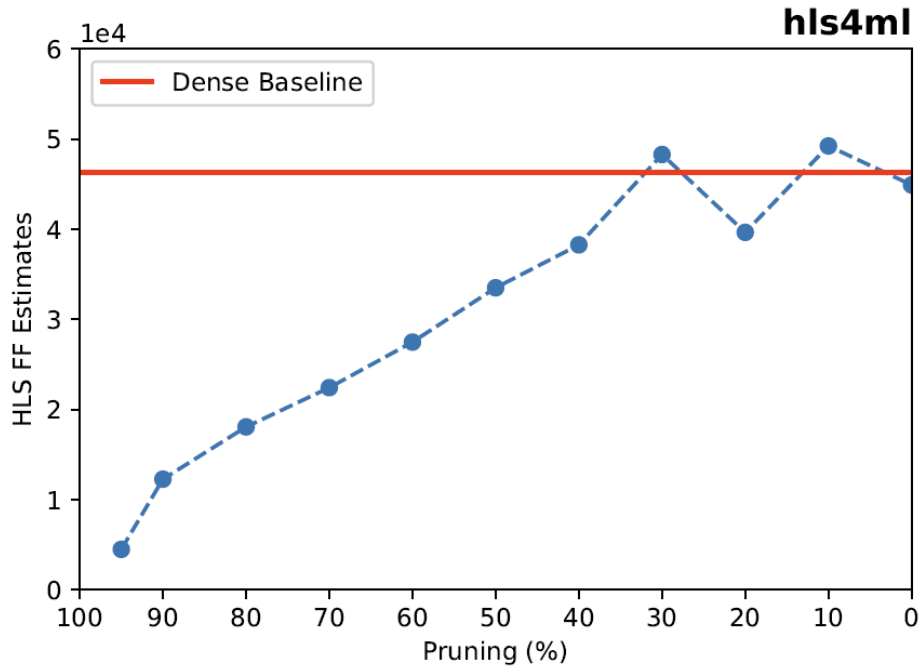
Results

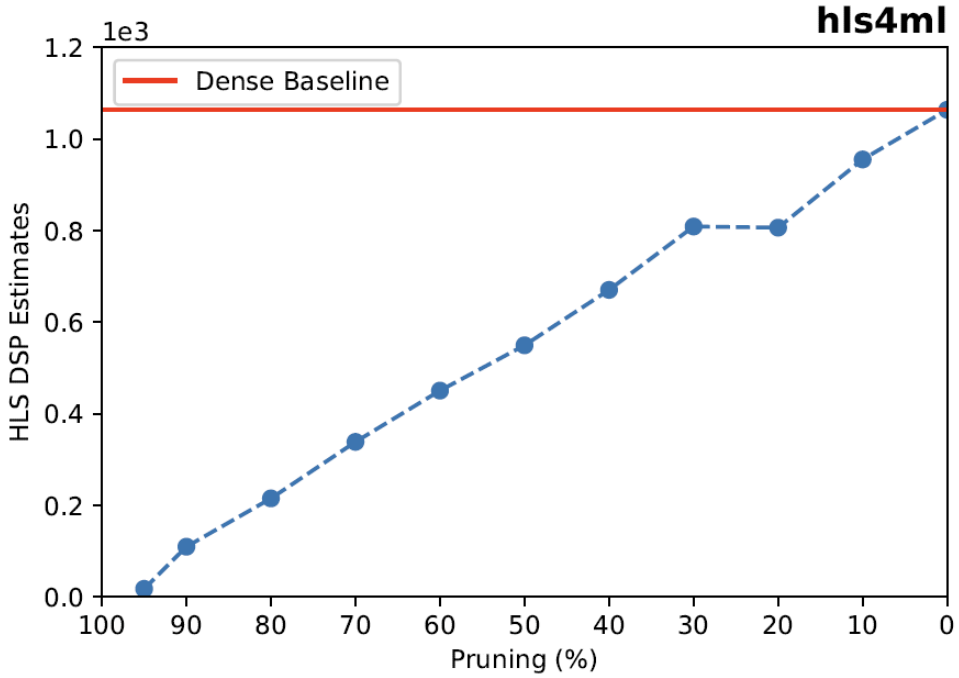




Results are shown for an Arria 10 FPGA with a 4ns clock. Missing points represent designs that consume more resources than those present on the device

Results





Results are shown for an Arria 10 FPGA with a 4ns clock.

Results

- A wider & deeper version of the previous model.
  (16 x 200 x 200 x 200 x 200 x 200 x 5)

Reuse Factor	ALUTs	FFs	RAMs	MLABs	DPSs	Latency	ii
	%					Cycles @ 4ns	
100	20	9	32	0	825	684	100
200	15	7	16	0	412.5	1280.5	200

Results are shown for an Arria 10 FPGA with a 4ns clock.

Test Case - LSTMs

- A Deep LSTM trained for anomaly detection at LIGO.

  Input = (timestep , 1)

```
L1 = LSTM(32, activation='relu', return_sequences=True)(inputs)
L2 = LSTM(8, activation='relu', return_sequences=False)(L1)
L3 = RepeatVector(X.shape[1])(L2)
L4 = LSTM(8, activation='relu', return_sequences=True)(L3)
L5 = LSTM(32, activation='relu', return_sequences=True)(L4)
output = TimeDistributed(Dense(X.shape[2]))(L5)
```

[2] Eric Anton Moreno et al, Anomaly Detection with Spiking Neural Networks. Link.

Results

- Resource Usage (mixed RF @ 1,8,32)

	ALUTs	FFs	RAMs	MLABs	DSPs
System	134233 (16%)	77911 (5%)	2356 (87%)	83 (0%)	1148.5 (76%)

- Latency @ (timesteps = 8) = 320 cycles or 1.1us (@300MHz)

Results are shown for an Arria 10 FPGA

Conclusion

- We proposed a new backend within hls4ml to deploy DL models to Intel FPGAs.
- We introduce a new approach to build scalable layers using basic computational units.
- We then use it to add support for sparse dense and scalable recurrent layers.

Questions?

Future Work

- Performance / Error prediction for fast config-level design-space exploration.
- Support for convolution and pooling layers.
- End-to-end workflow for integration into low-power SoC’s.