Tree Tensor Network predictors implemented on FPGA for ultra-low latency inference.

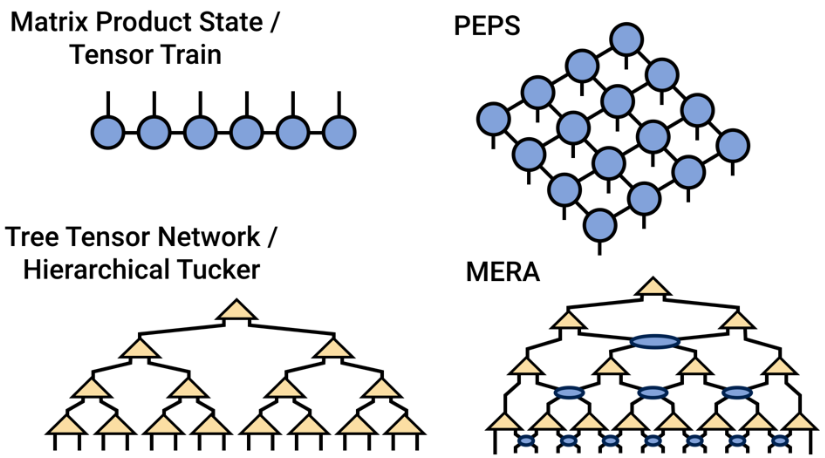
International Conference on Quantum Technologies for High-Energy Physics

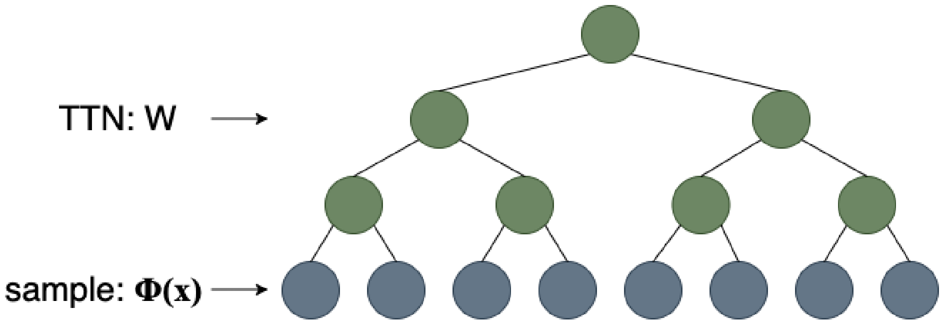
22 January 2025

L. Borella, A. Coppi, J. Pazzini, A. Stanco, A. Triossi, M. Zanetti.
University of Padua and INFN. lorenzo.borella.1@phd.unipd.it

This work is partially supported by ICSC, Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union, NextGenerationEU.

Tree Tensor Networks for machine learning





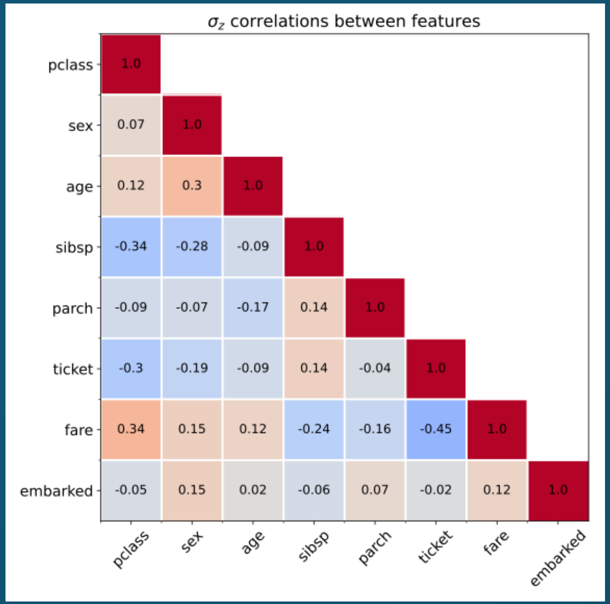
- **Tensor network methods:** represent wavefunctions $|\psi\rangle$ and hamiltonians H of many-body quantum systems on classical computers[1] .
- Tree Tensor Networks can be trained as **machine learning classifiers.**
- **Classical data** samples are represented as separable quantum states, encoding each feature as a qubit.
- A **supervised learning** algorithm can train the tree tensors according to a classification decision function.
- After training, the TTN architecture encodes the **learned information** as quantum entangled state.

[1] E.Miles Stoudenmire and David J. Schwab. «*Supervised learning with quantum-inspired tensor networks*.» *ar*Xiv: 1605.05775.

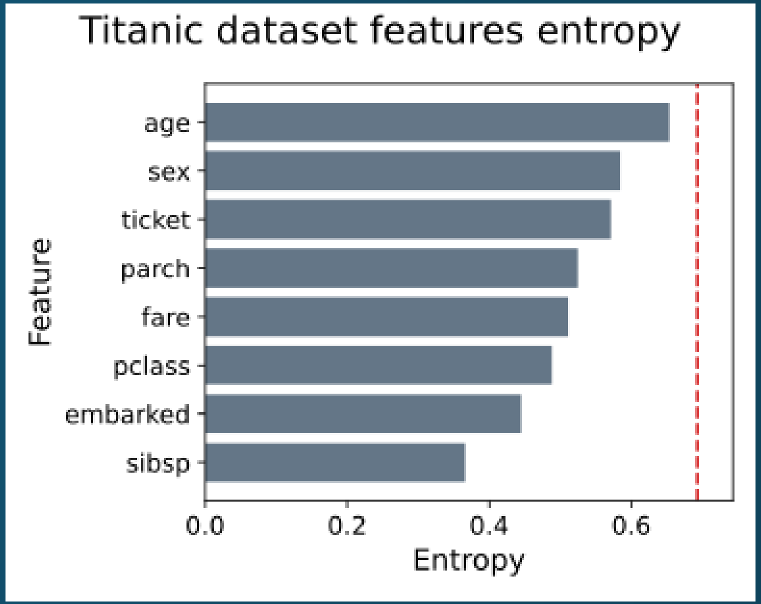
Tree Tensor Network for machine learning

- **Compression while learning**: bond dimensions can be optimized during training, reducing the total number of parameters by truncating the size of the inner links of the network with SVD.

- **Quantum correlations:** remove redundant information by studying feature correlation and highlighting the ones that are the least correlated.



- **Von Neumann Entropy:** study the relevance of the learned information encoded in each TTN bipartition and prune useless branches[2].



[2] A. Giannelle D. Zuliani T. Felser D.Lucchesi S.Montangero M. Trenti, L. Sestini. «*Quantum-inspired machine learning on high-energy physics data*» *Nature*, 2021. https://doi.org/10.1038/s41534-021-00443-w

Tree Tensor Network inference on FPGA

TTN

- **Optimized learning**: SVD, bond dimension tuning.
- **Safe post-training pruning**: entropy and correlation.
- **Linear algebra**: only tensor contractions involved.
- **Highly parallelizable** inference algorithm.
- **Performances** comparable to classic ML methods.

+

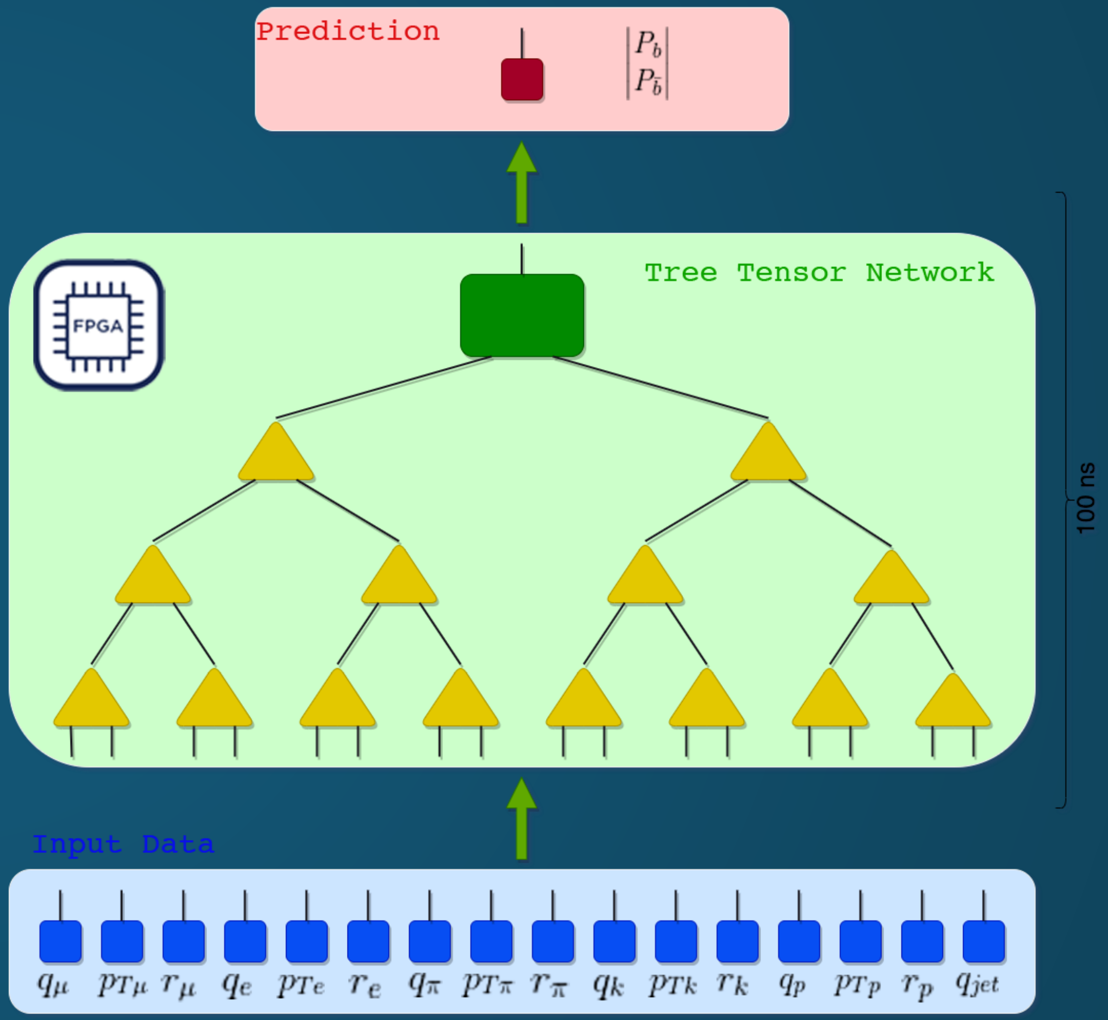
FPGA

- Versatile **programmable hardware.**
- **Pipelined parallel computations.**
- **Deterministic latency.**
- **Limited resources:** need for **compressed architectures** and optimal exploitation of logic.
- **Sub-microsecond latency**: deployable for **online processing** for HEP experiments.

Tree Tensor Network inference on FPGA

Dataset	Iris	Titanic	LHCb [3]	hls4ml
Features	4	8	16	16
Bond dimensions	[2,4]	[2,4,8]	[2,4,8,8]	[2,4,10,10]
Classes	2	2	2	5
Accuracy	99%	77%	62%	73%
Memory	96 B	768 B	3 kB	6 kB

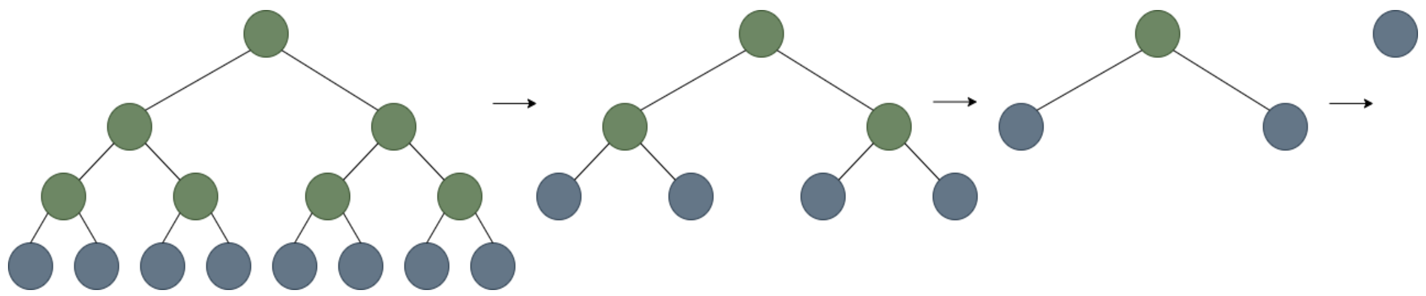
- **Task:** binary and multi classification.
- **Software:** successfully trained several TTN architectures.
- **Hardware:** inference offloaded in FPGA and validated.



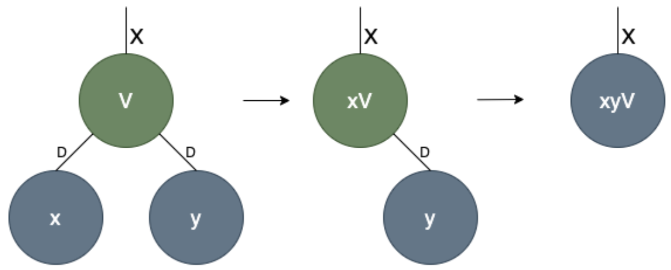
[3] L. Borella, A. Coppi, J. Pazzini, A. Stanco, M. Trenti, A. Triossi, M. Zanetti «*Ultra-low latency quantum-inspired machine learning predictors implemented on FPGA*», arxiv:2409.16075

Tree Tensor Network inference on FPGA

1. FPGA is programmed with **architecture-specific firmware**.
2. Software-trained **weights** are **loaded** on static blocks of RAM.
3. Data that needs to be classified is **streamed** to the FPGA.
4. Feature **mapping** is applied on input data and implemented **in hardware** with LUTs.
5. Full contraction with the TTN architecture.
6. Retrieve **final probability** and classify sample.



Tensor contraction is the base operation that needs to be defined on FPGA: choose **different degrees of parallelization** and iterate it for different layers.



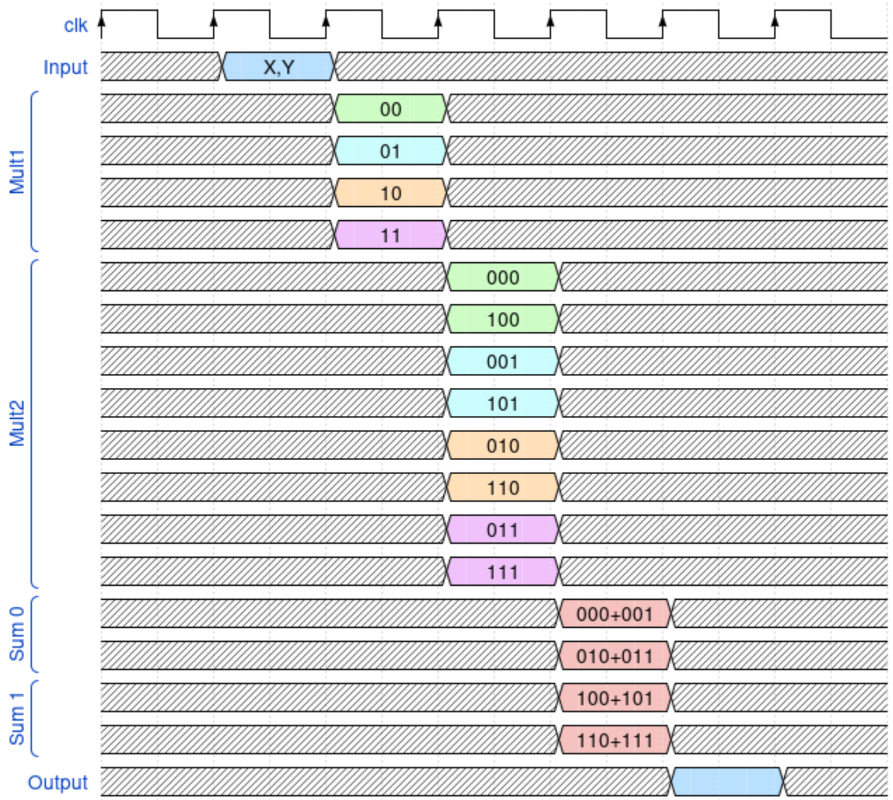
$$z_i = \sum_{j,k} x_j y_k V_{ijk}$$

Digital Signal Processor (DSP) is the resource devoted to arithmetic calculations on FPGA.

Full Parallel implementation

$$latency = \sum_{i=1}^{L} 2 + \log_2(\chi_{i-1}^2)$$

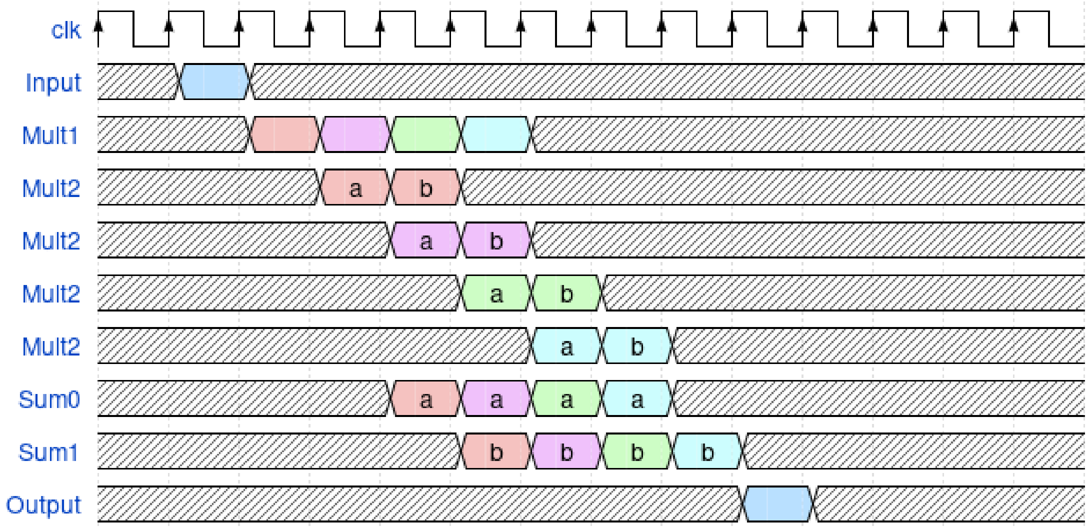
$$DSP = \sum_{i=1}^{L} \chi_{i-1}^2(\chi_i + 1)\frac{N}{2^i}$$

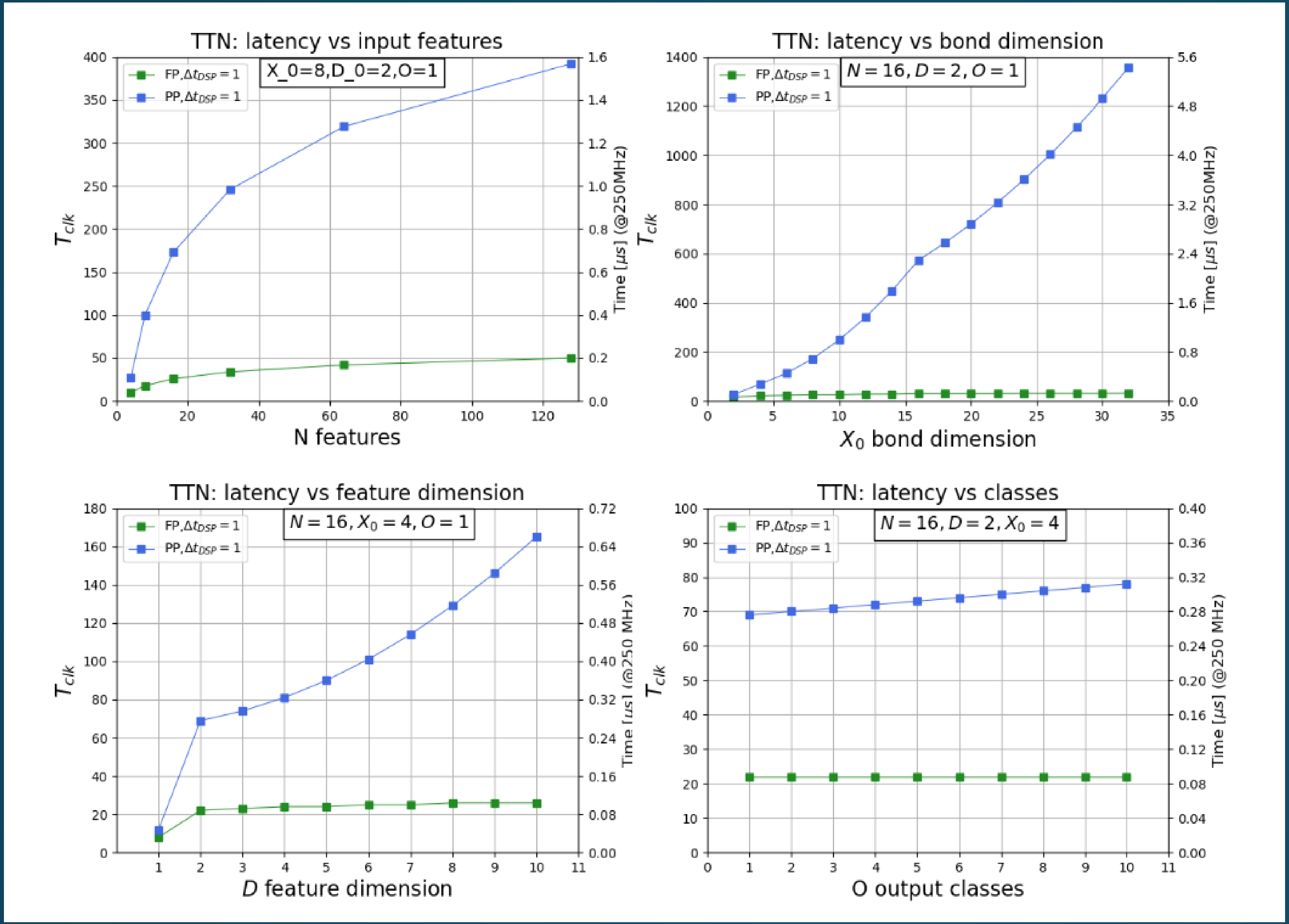


Partial Parallel implementation

$$latency = \sum_{i=1}^{L} \chi_{i-1}^2 + \chi_i + 1$$

$$DSP = \sum_{i=1}^{L} (\chi_{i-1}^2 + 1)\frac{N}{2^i}$$

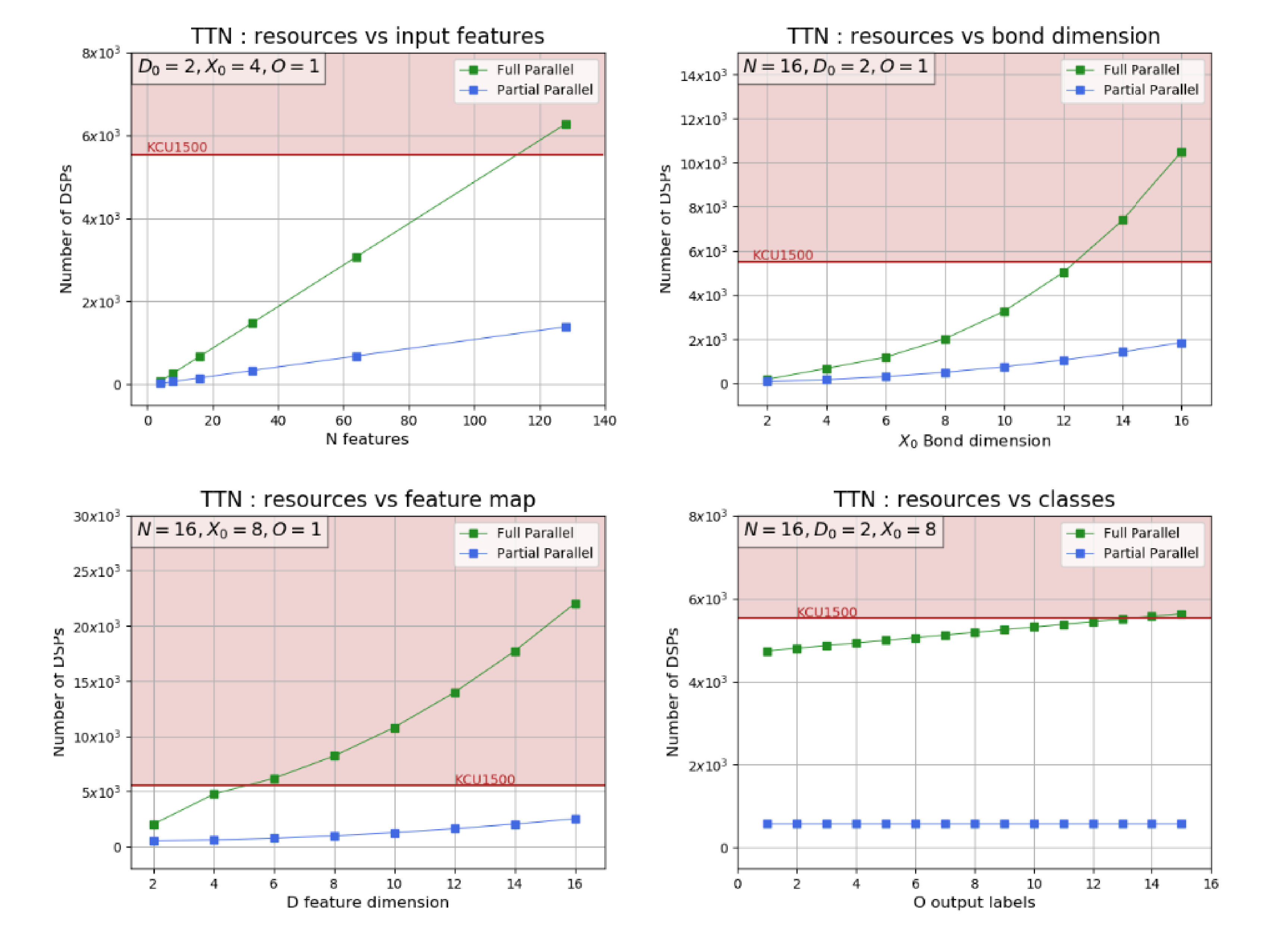




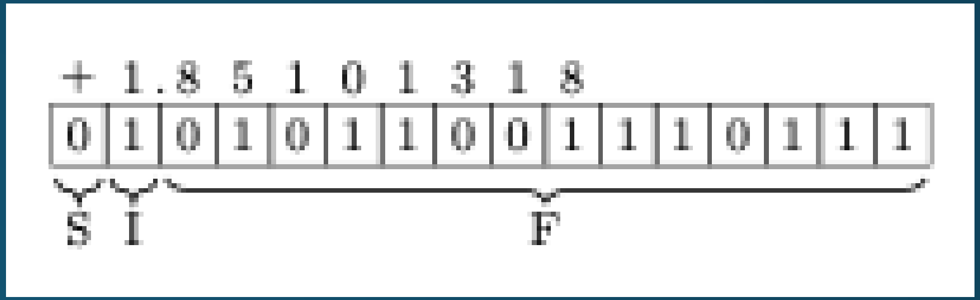
TTN: latency vs input features
X_0=8,D_0=2,O=1
FP,$\Delta t_{DSP} = 1$
PP,$\Delta t_{DSP} = 1$
T_{clk}
Time [μs] (@250MHz)
N features
TTN: latency vs bond dimension
$N = 16, D = 2, O = 1$
FP,$\Delta t_{DSP} = 1$
PP,$\Delta t_{DSP} = 1$
T_{clk}
Time [μs] (@250MHz)
X_0 bond dimension
TTN: latency vs feature dimension
$N = 16, X_0 = 4, O = 1$
FP,$\Delta t_{DSP} = 1$
PP,$\Delta t_{DSP} = 1$
T_{clk}
Time [μs] (@250 MHz)
D feature dimension
TTN: latency vs classes
$N = 16, D = 2, X_0 = 4$
FP,$\Delta t_{DSP} = 1$
PP,$\Delta t_{DSP} = 1$
T_{clk}
Time [μs] (@250 MHz)
O output classes

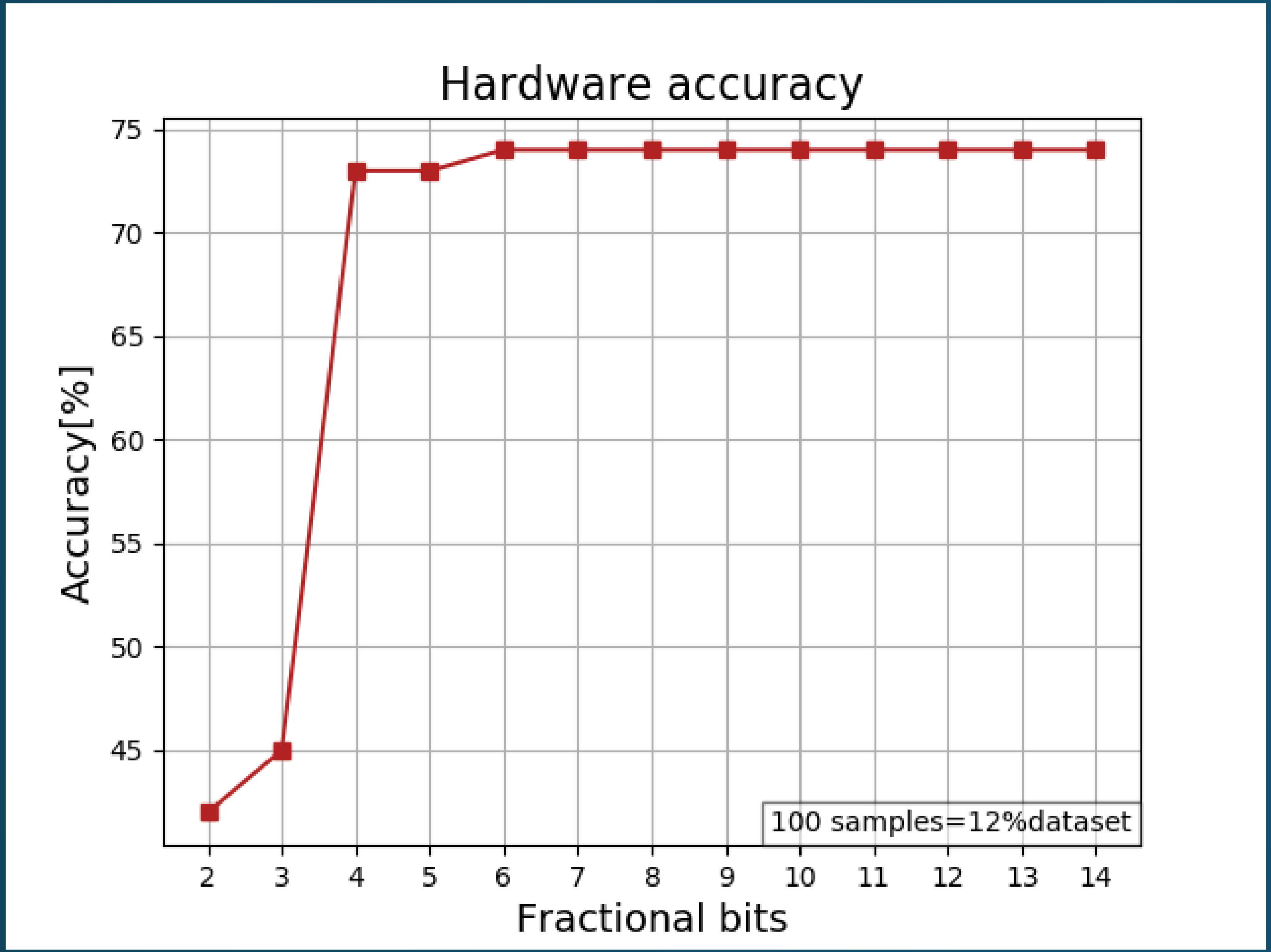
Latency

Resources


TTN : resources vs input features
$D_0 = 2, X_0 = 4, O = 1$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
N features
TTN : resources vs bond dimension
$N = 16, D_0 = 2, O = 1$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
X_0 Bond dimension
TTN : resources vs feature map
$N = 16, X_0 = 8, O = 1$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
D feature dimension
TTN : resources vs classes
$N = 16, D_0 = 2, X_0 = 8$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
O output labels

Quantization

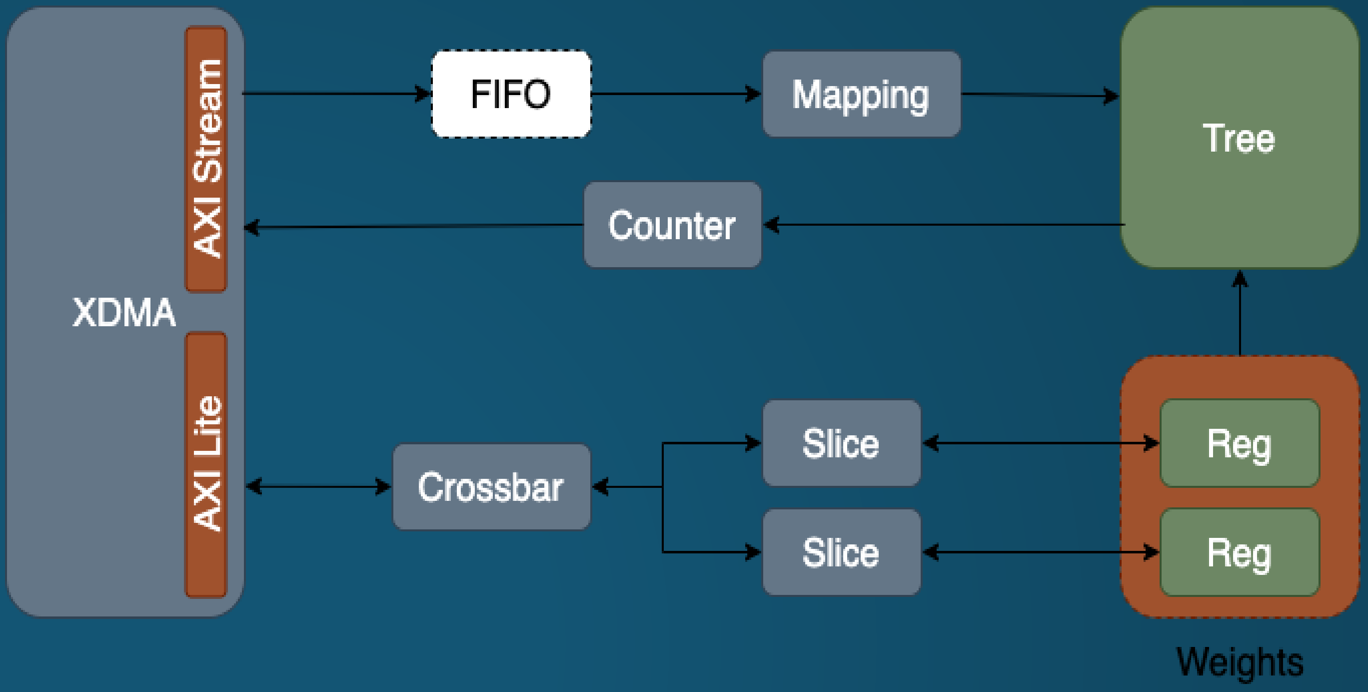


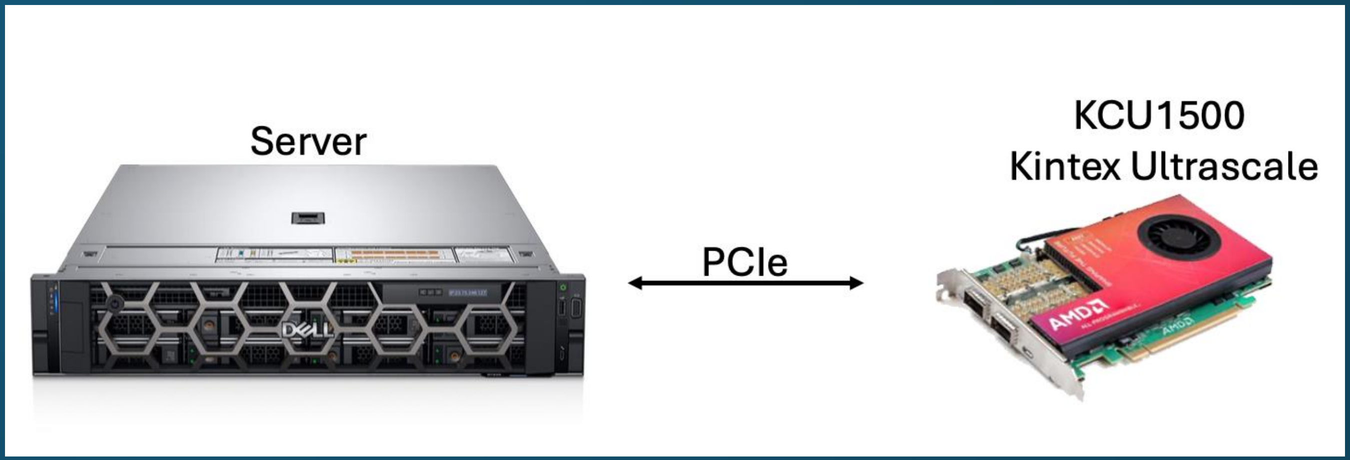


- Real values represented as **16 bits fixed points.**
- The choice for **numeric precision** is **model-specific**.
- **Avoid DSP usage** for number of fractional bits below hardware-specific threshold.
- Additional network compression without loosing classification accuracy.

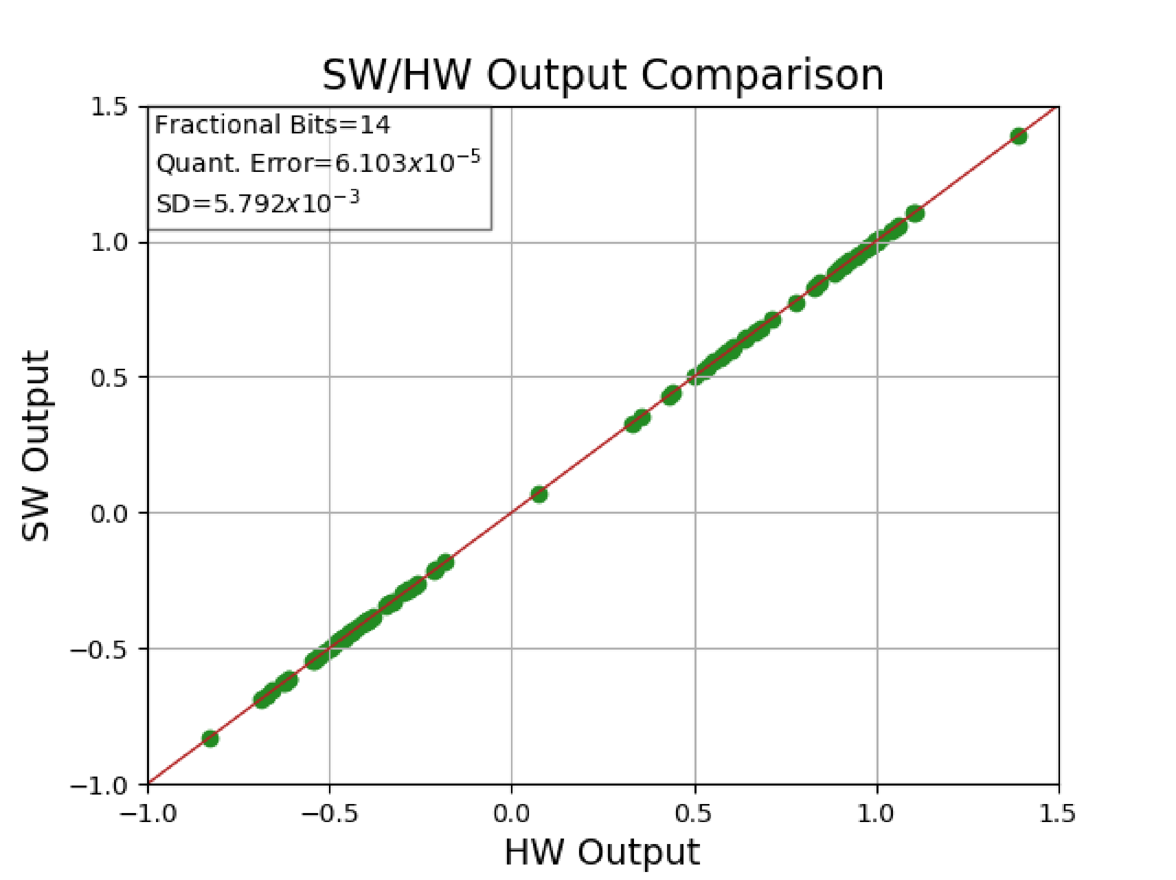
Firmware

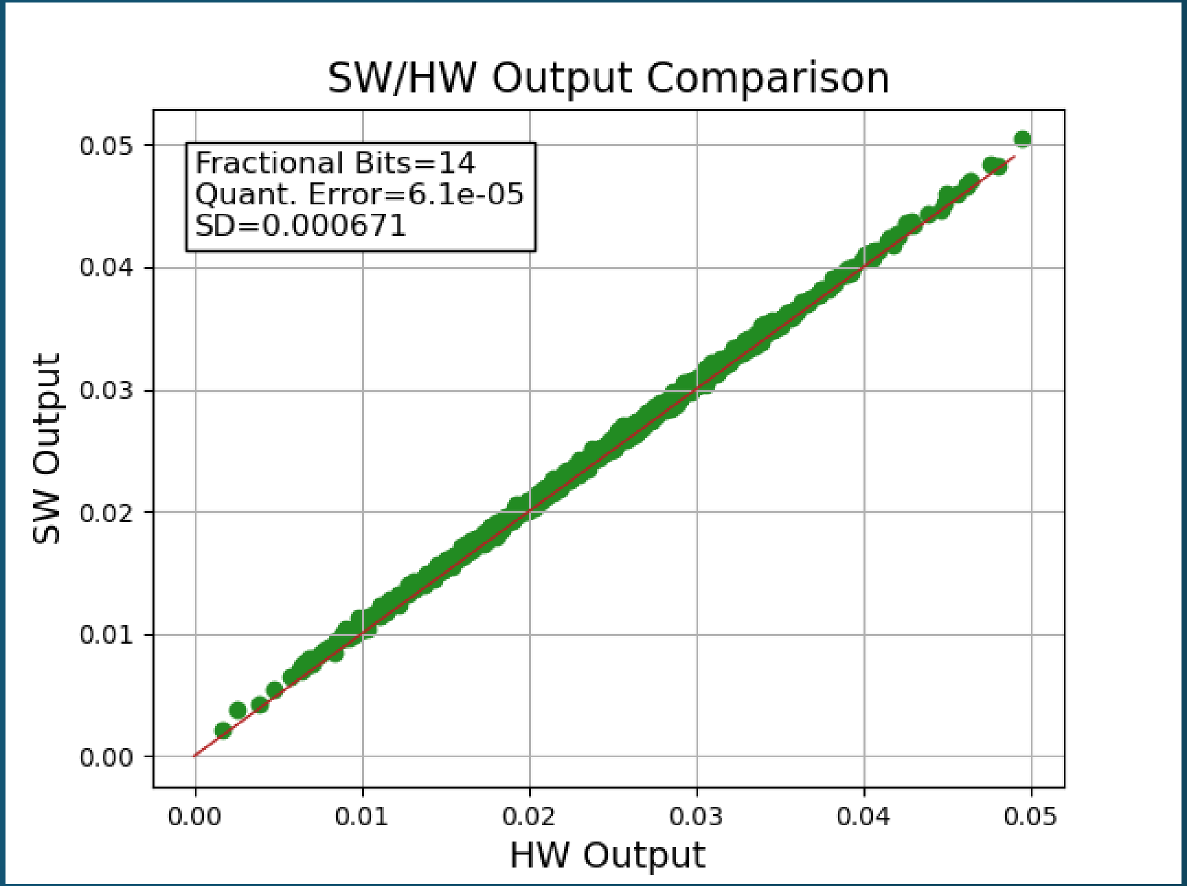
- Firmware described in **VHDL** using **Vivado 2024**.
- Project developed on an **KCU 1500 Kintex UltraScale** board.
- Development board plugged on host PC with **PCIe communication.**
- **AXI Lite** and **AXI Stream** protocols with global clock frequency of **250MHz.**
- OOC TTN implementation reaching **500 MHz.**





Results of inference



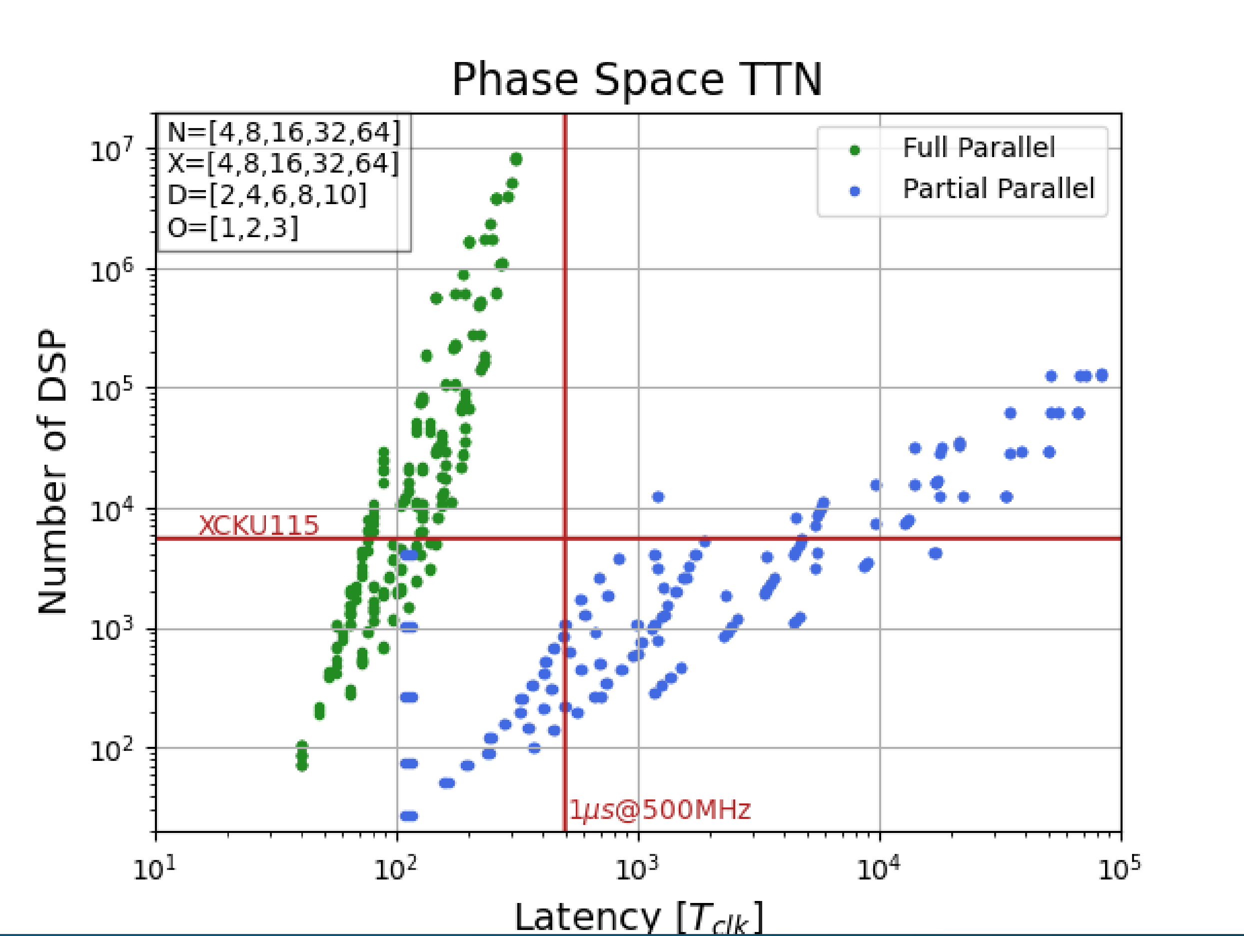


- Titanic dataset, 8 features, 100 samples
- Input: 10^{-5} -> Output (average): 10^{-3} (7 LSBs)
- Hw accuracy drops with less than **4 frac. bits**

- LHCb dataset, 16 features, 500 samples
- Input: 10^{-5} -> Output (average): 10^{-4} (4 LSBs)
- Hw accuracy drops with less than **10 frac. bits**

Conclusions

- Trained **TTN architectures** and derived **accuracies comparable with NN** counterparts.
- **VHDL firmware** for TTN inference with **different degrees of parallelization**.
- **Deterministic projections** of **resources and latency** values for different TTN architectures.
- **Exact replication** of software behaviour in **FPGA hardware.**
- Inference algorithm latency **below 1 us:** possible deployment in **trigger pipeline** of HEP experiments.



Dataset	TTN	DSP	Latency
Iris	PP [2,4,1]	1%	108 ns
Titanic	FP [2,4,8,1]	8%	72 ns
LHCb	FP [2,4,8,8,1]	36.5%	104 ns
hls4ml	FP [2,4,10,10,5]	66.38%	104 ns

Thank you!