Quantum inspired ML on FPGA

Marco Zanetti
University of Padova and INFN
BOOSTlab

Setting the stage

In the following there won't be

- Answers about what can Quantum Computers do for HEP[1]
- A review of Quantum inspire algorithms for HEP pipelines
- Guidelines on how to use FPGAs to control/simulate quantum hardware

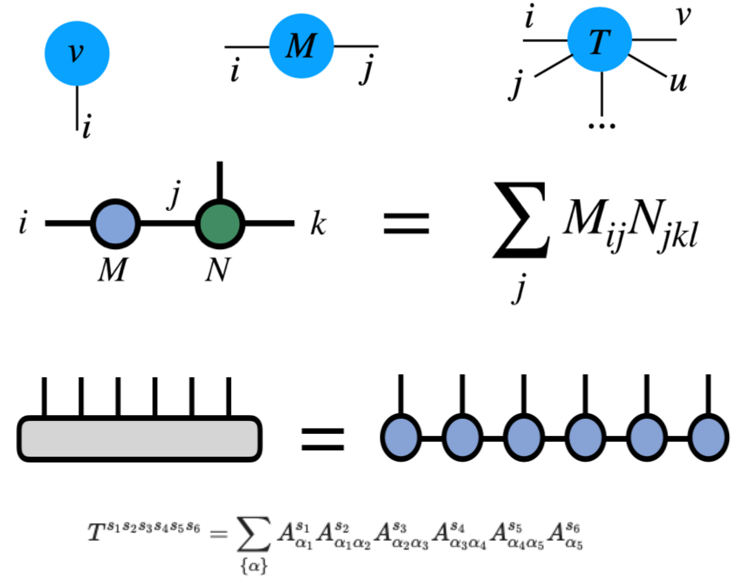
This is only about

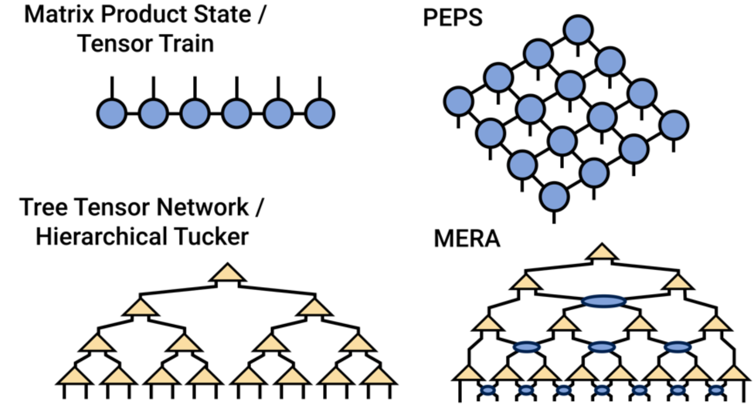
- Tensor Networks are "quantum" tools that can be used for classification/selection and anomaly detection
- Their features make them promising models to be deployed on resurces-limited contexts, notably first Trigger layers
- A case study of the deployment of TN as classifier for HEP tasks is reviewed

[1] See e.g. A. Di meglio et al. «*Quantum Computing for High-Energy Physics: State of the Art and Challenges*», PRX QUANTUM 5, 037001 (2024)

Tensor Networs and their usage

- TN: collections of tensors with indeces contracted in specific patterns
  - convenient graphical representation
- Rapresent/solve many-body quantum entangled states, factorizing rank-N tensors into smaller tensors
  - allows linear scaling (vs exp.) on the number of sites
  - allows computation, by reducing number of parameters and algo complexity
- Expressivity of the TN tuned by the bond dimenson
  - the dimension of the index connecting one tensor to the next
- Several topologies are suitable for various tasks
  - Tree TN: the most general loopless architecture



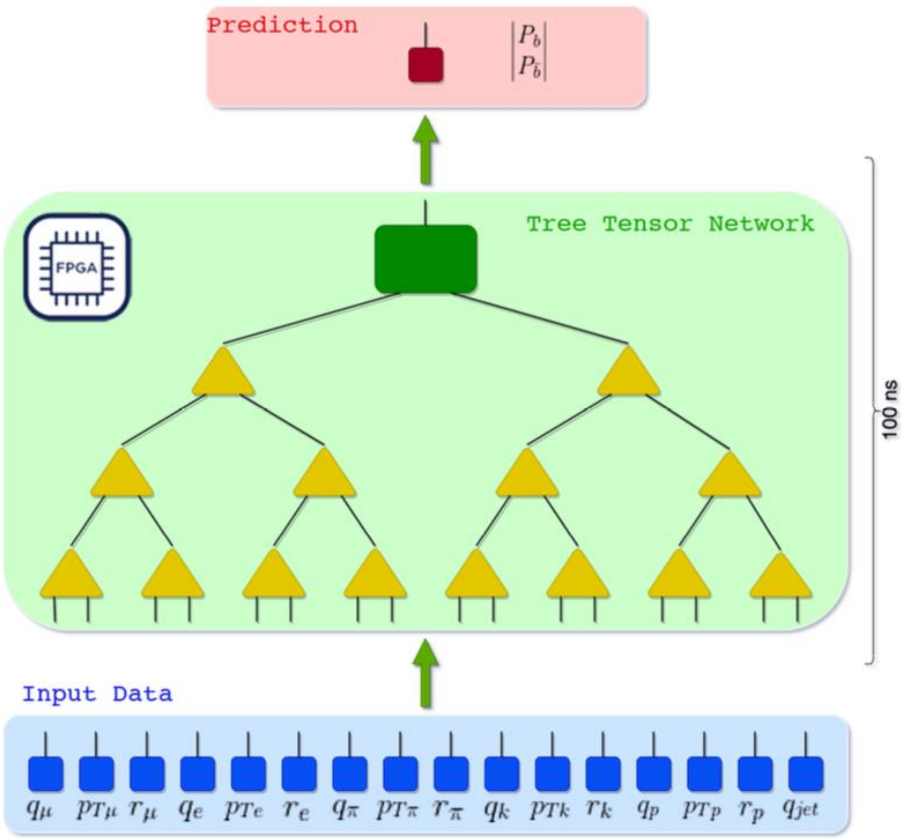


TN for (Q)ML

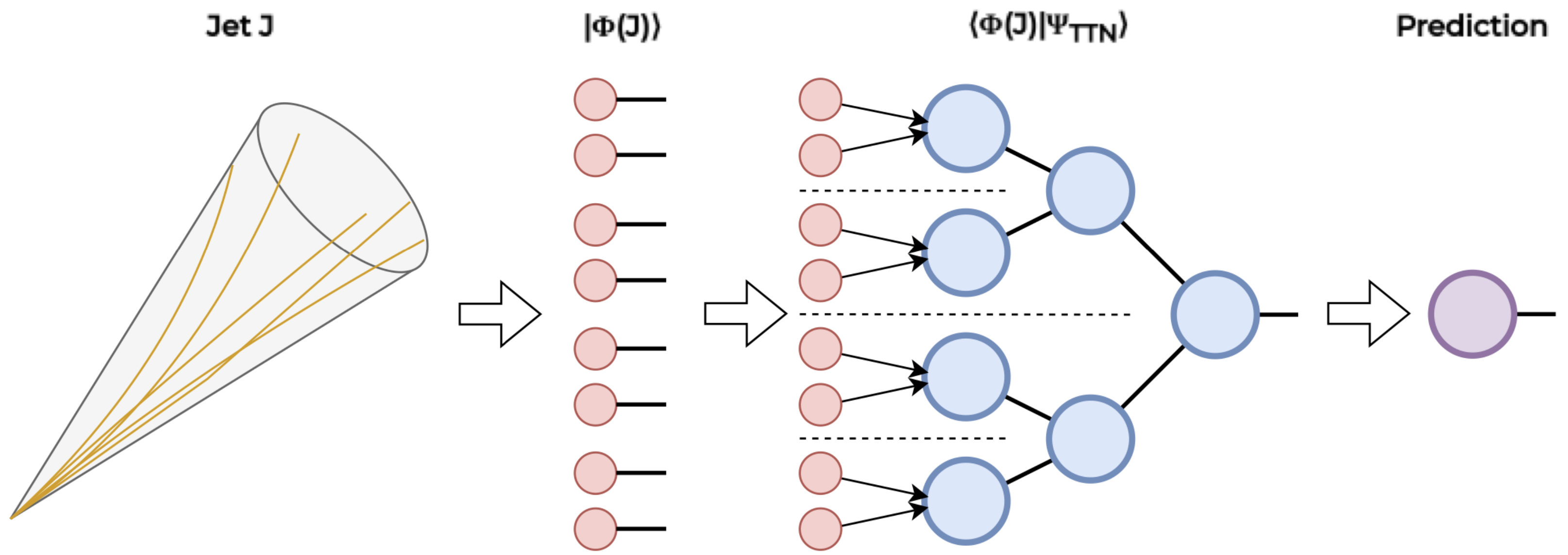
- Being a network, a TN can be trained as any other ML model.
- Methodology:
  - A TN used as "weight tensor" W, acting as a classifier on the input data {x}
  - the sample {x} is encoded into a feature map φ(x)
  - the confidence for a certain label *g* is

$$P_g = W \cdot \varphi(x)$$

- Eventually, the TN architecture encode the learned information representing a quantum entangled state.



TN4ML workflow



Data

- Different fields (HEP, OD)
- High rate

Embedding

- Basis embedding
- Spin or polynomial mapping

(Parallel) Contraction

- Calculate fidelity of TTN state over data
- Embarrassingly parallel operation

Prediction

- Scalar or vector out
- Plug into loss function for optimisation

TN features

Linearity

Contractions are linear operation → Inference robustness against hallucinations, eases computational representation

Compression (while learning)

bond dimensions optimized during training: reduction of number of parameters by truncating the size of the hidden links with SVD.

Quantum correlations

remove redundant information by studying feature correlation and highlighting the ones that are the least correlated.

$$C^l_{i,j} = \frac{\langle \psi_l | \sigma^z_i \sigma^z_j | \psi_l \rangle}{\langle \psi_l | \psi_l \rangle}$$

Von Neumann Entropy

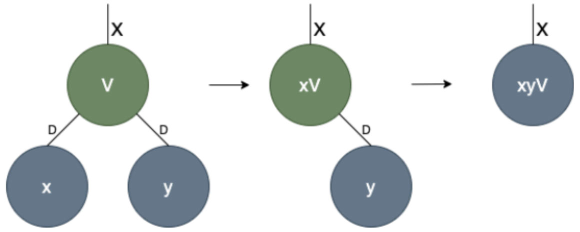
asses the relevance of the learned information encoded in each TTN bipartition → prune useless branches

$$S(\rho_A) = -\mathrm{Tr}[\rho_A \log \rho_A] = -\mathrm{Tr}[\rho_B \log \rho_B] = S(\rho_B)$$

Inference on Hardware

- FPGA programmed with architecture-specific firmware.
- Software-trained weights loaded on static RAM blocks or hardcoded in firware.
- Data to be classified streamed to the FPGA.
- Full contraction with the TTN architecture.
- Final probability is retrieved for subsequent steps (e.g. selection)

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



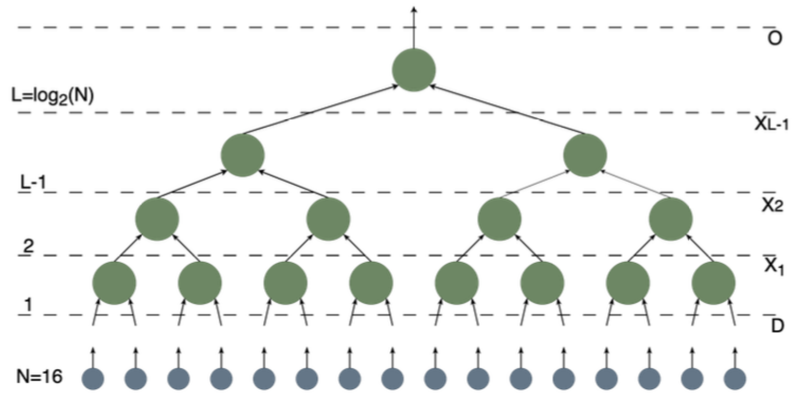
Digital Signal Processors (DSP) exploited for the actual node contraction; just products and sums.

Case study

- Consider datasets with increasing complexity
- Start of with a couple of ML banchmarks
- Main goal are HEP “standards”
  - b/anti-b classification datasets from LHCb
    - from A. Giannelle et al. «Quantum-inspired machine learning on high-energy physics data» Nature, 2021. https://doi.org/10.1038/s41534-021-00443-w
  - “hlt4ml” jet tagging dataset
    - from Duarte et al. «Fast inference of Deep Neural Networks in FPGA for particle Physics» https://arxiv.org/abs/1804.06913
- Methodology and results described in:
  - L. Borella et al. «*Ultra-low latency quantum-inspired machine learning predictors implemented on FPGA*», arxiv:2409.16075

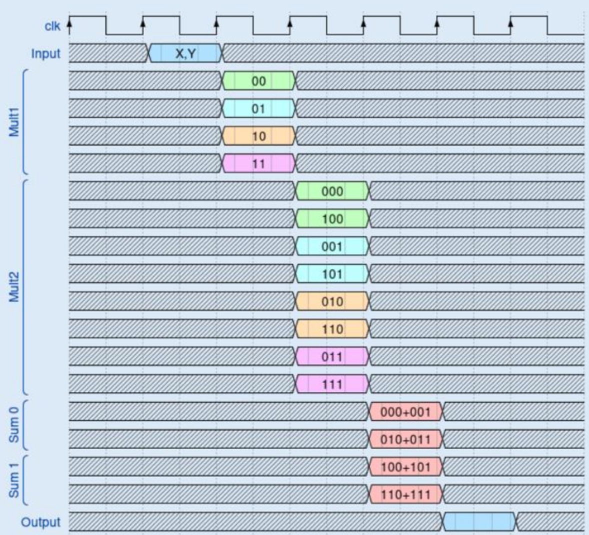
Full parallel implementation

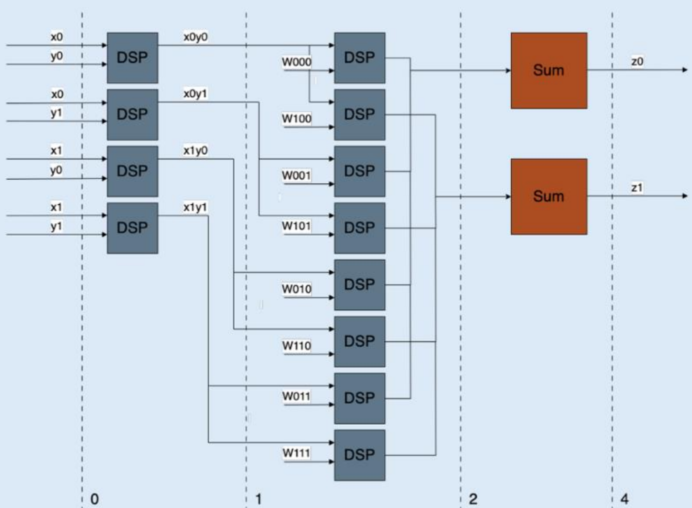
Maximize number of DSPs used and minimize total algorithmic latency



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

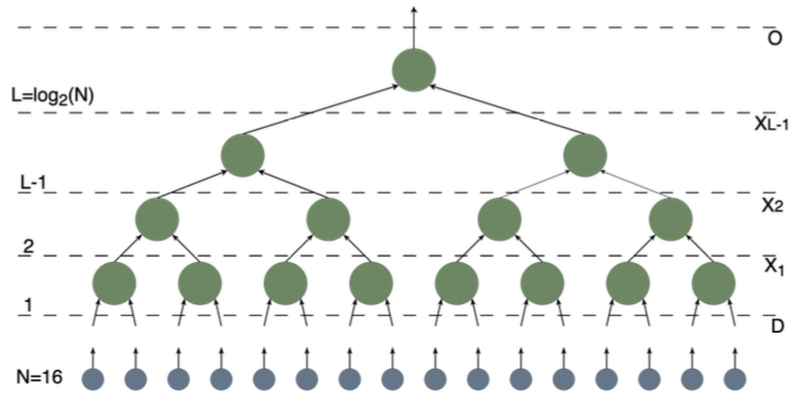
$$DSP = \sum_{i=1}^{L} \chi_{i-1}^2(\chi_i + 1)\frac{N}{2^i}$$



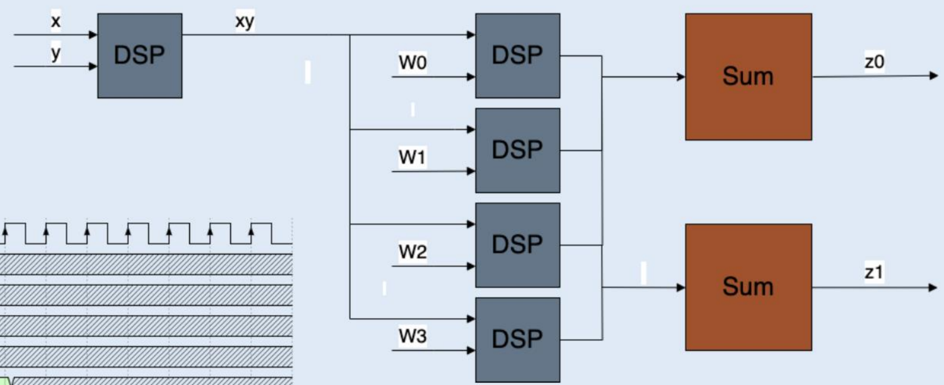


Partial parallel implementation

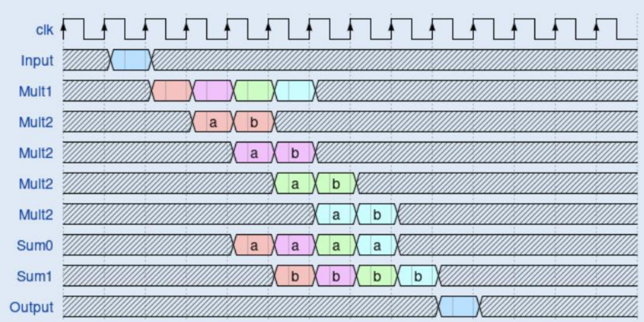
Reuse some of the DSPs, with a resulting increase in latency



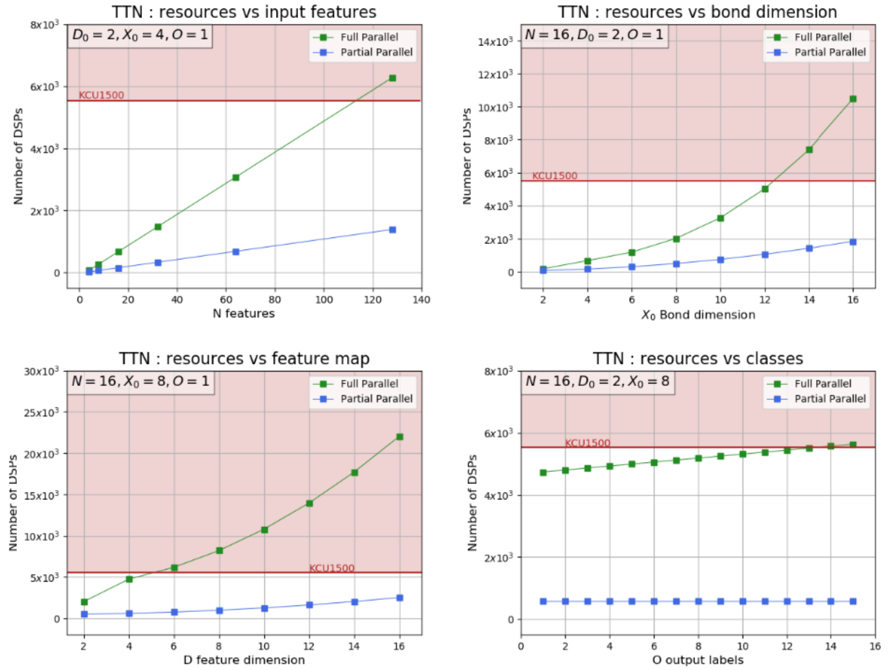
$$latency = \sum_{i=1}^{L} \chi_{i-1}^{2} + \chi_i + 1$$



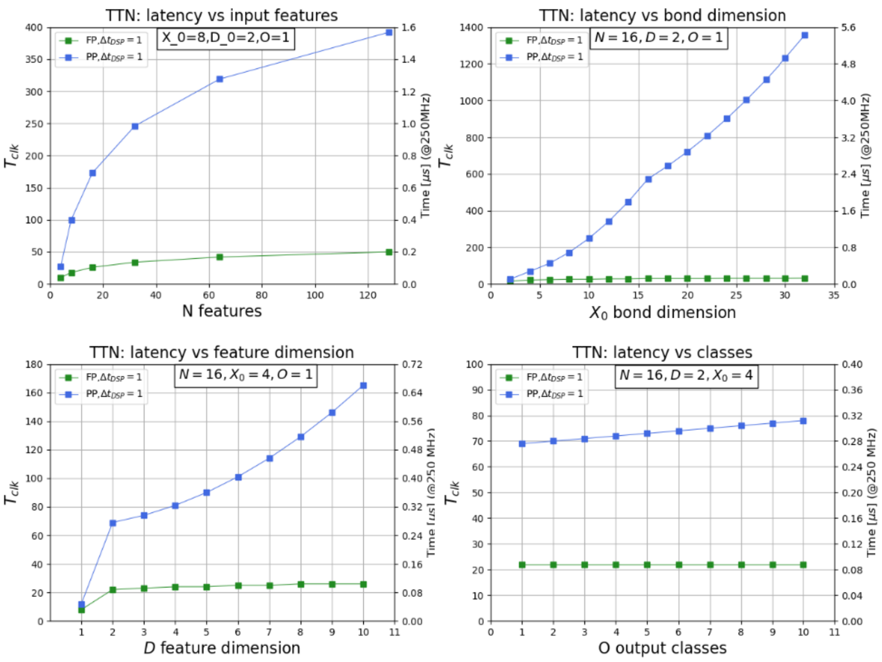
$$DSP = \sum_{i=1}^{L} (\chi_{i-1}^{2} + 1) \frac{N}{2^i}$$



Results: Latency and Resources

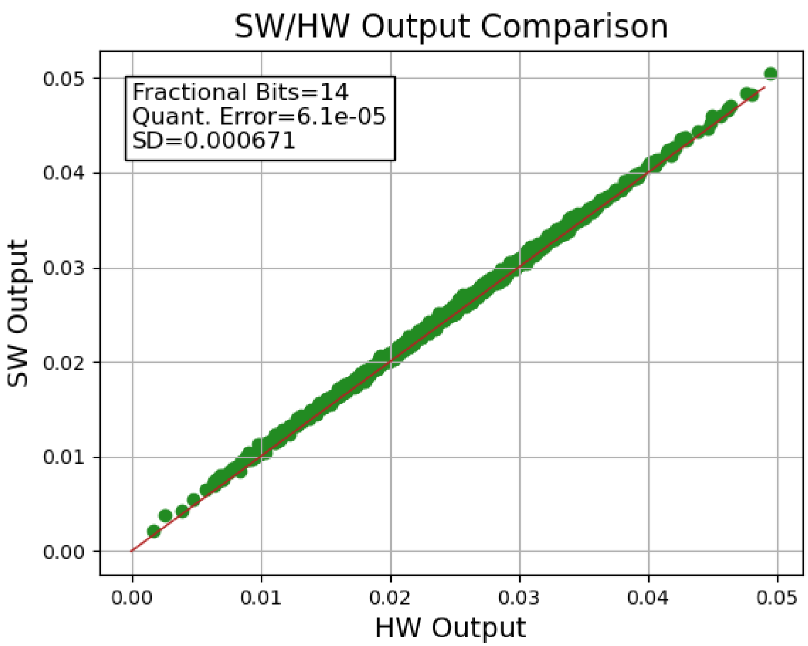


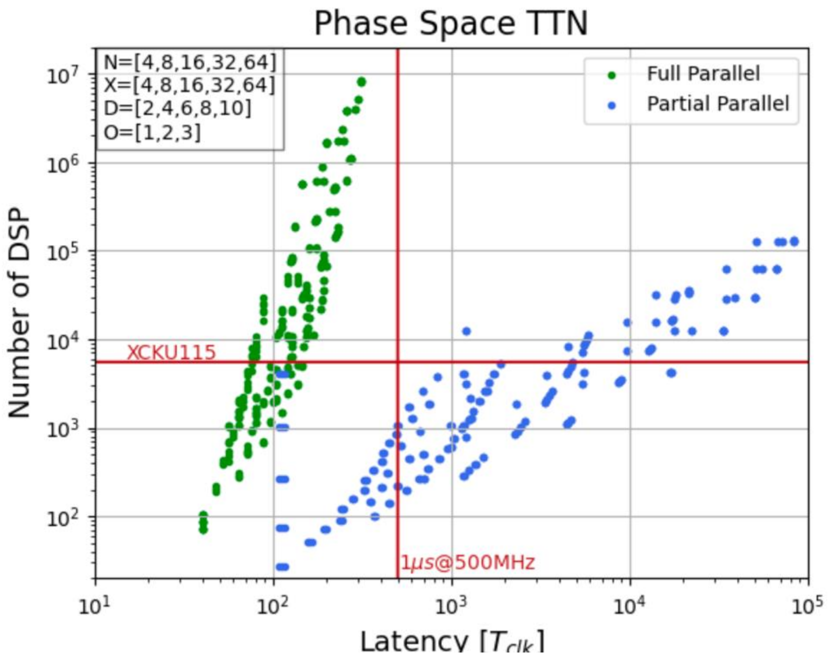
Latency



DSP usage

Results





Dataset	Iris	Titanic	LHCb [6]	hls4ml [5]
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

Conclusions and Outlook

- TN as a valid option to tackle ML tasks
- TN's features make them suitable for deployment in hardware to achieve the ultrafast inference needed by HEP trigger systems
- Next steps:
  - Test different TN topologies (MPS, MERA, PEPS etc.) and tasks.
  - Move firmware programming to higher level languages (e.g. from VHDL to HLS4ML).
  - Check Hardware inference on Versal AI Engines.
- Other applications
  - Consider possibility of TN training on FPGA
  - Simulation of quantum circuits