Piano Nazionale
di Ripresa e Resilienza

Finanziato
dall'Unione europea
NextGenerationEU

ICSC
Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

Tree Tensor Network implementation on FPGA

Workshop on Tensor Networks and (Quantum) Machine Learning for High-Energy Physics

5 Novembre 2024

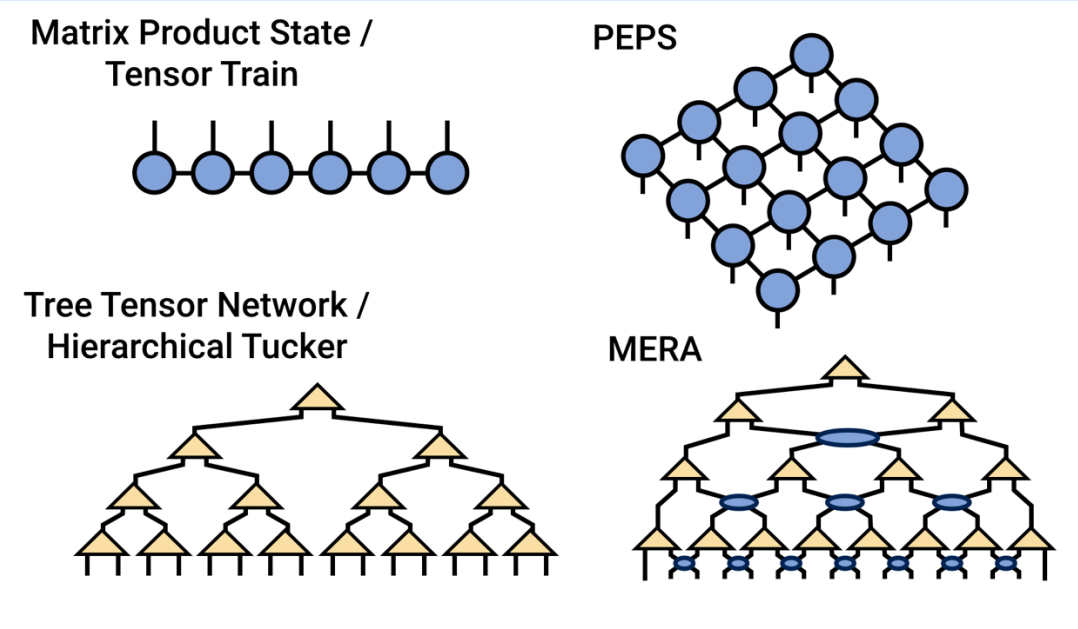
Lorenzo Borella
University of Padua and INFN
lorenzo.borella.1@phd.unipd.it

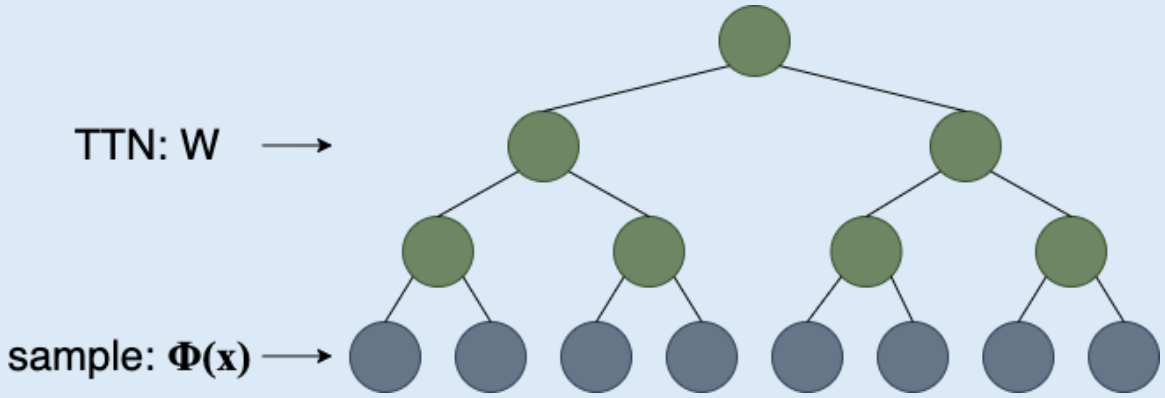
INFN PADOVA

UNIVERSITÀ DEGLI STUDI DI PADOVA

QUANTUM FRONTIERS

Introduction





- **Tree Tensor Networks (TTN)** can be trained as Machine Learning (ML) classifiers.
- **Classical data** samples mapped to represent a separable quantum state $\phi(x)$.
- **Supervised Learning**: teach the TTN how to classify each sample, following the decision function:

$$f(x) = W \cdot \phi(x)$$

- Eventually, the TTN architecture can **encode the learned information** representing a quantum entangled state.

TTN inference on FPGA

TTN

- **Optimized learning**: SVD, bond dimension tuning
- **Safe pruning** post-training: entropy and correlation.
- **Linear algebra**: only tensor contraction operations.
- **Highly parallelizable** inference algorithm.

FPGA

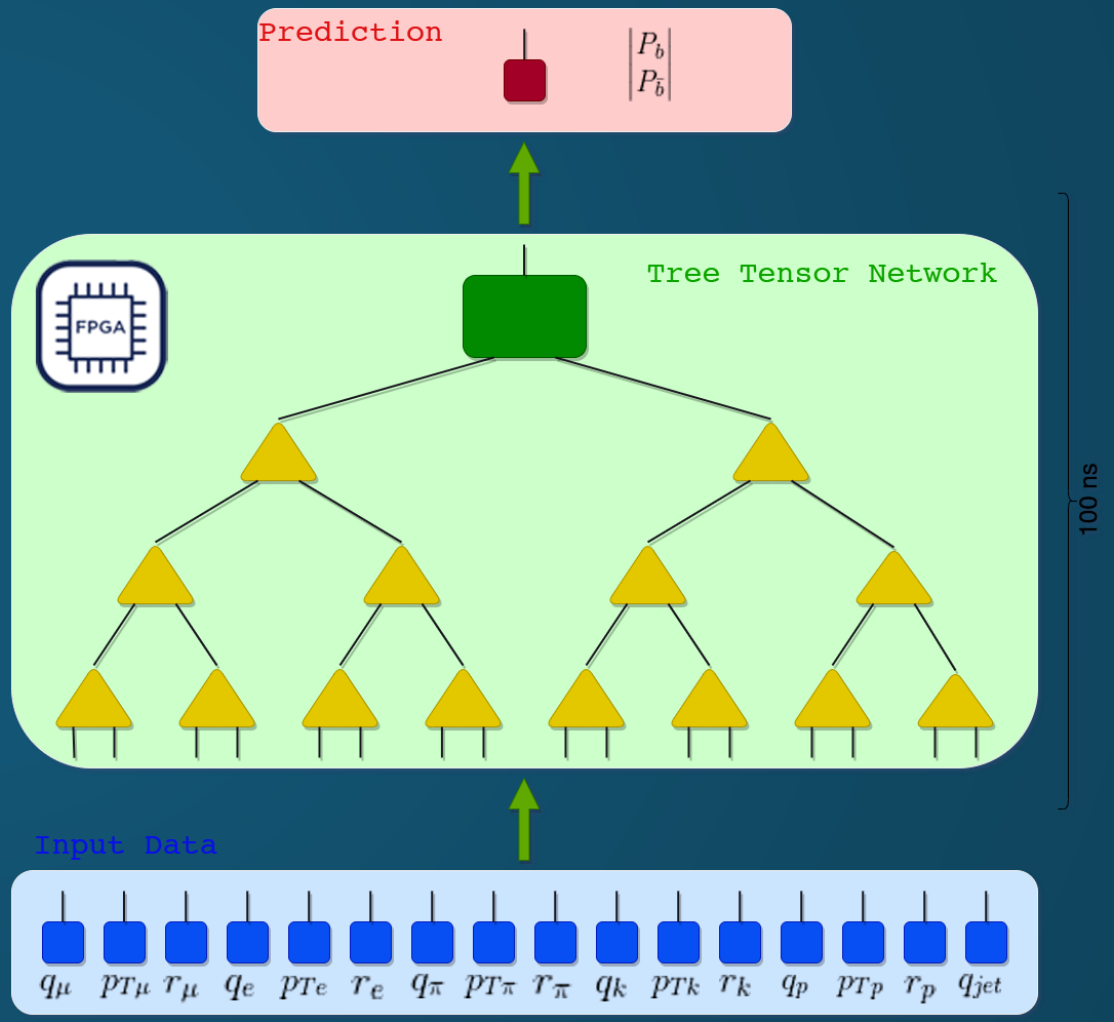
- **Programmable** hardware, extremely versatile.
- Well suited for **parallel computation.**
- **Limited resources**, need for architecture compression.
- Achievable **ultra-low latency.**

TTN on FPGA

- **Compressed architectures:** optimal exploitation of FPGA resources.
- **Performances** comparable to classic ML methods.
- **Sub-microsecond latency**: deployable for online processing.

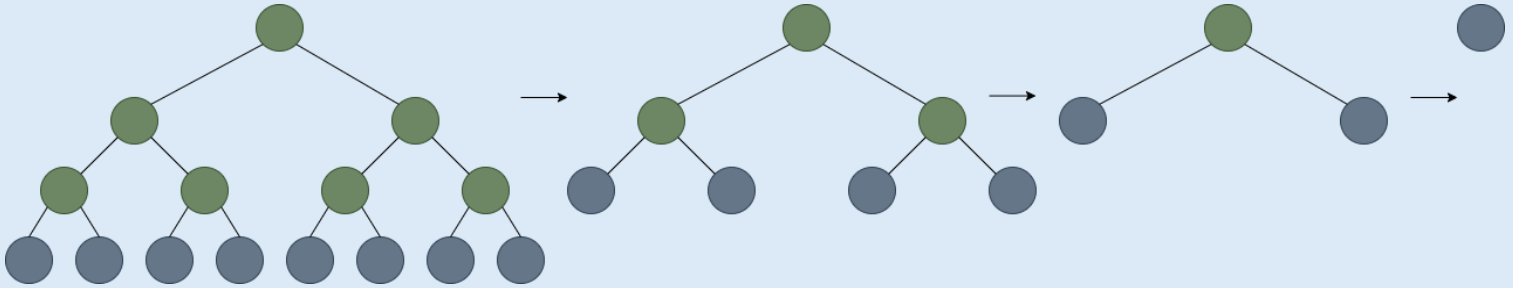
TTN inference on FPGA

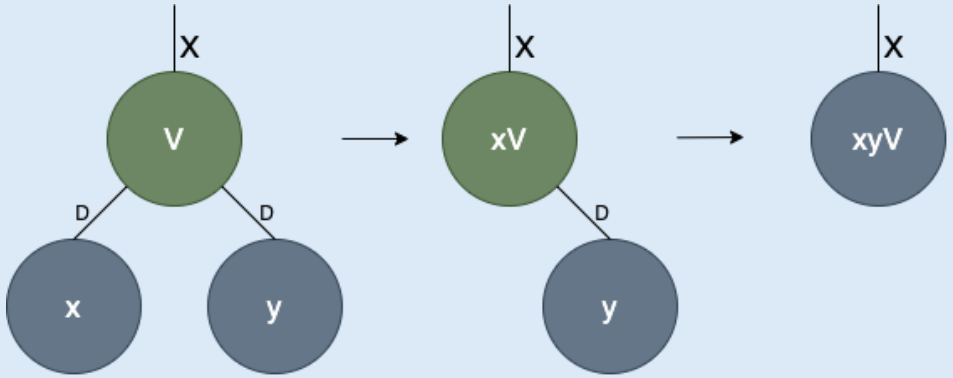
- **Task:** binary classification.
- **Datasets:** Iris, Titanic for benchmarking and LHCb open data for physics case.
- **Architectures**: several combinations of hyperparameters, number of features and feature mappings tested.
- **Software:** successfully trained several architectures.
- **Hardware:** inference offloaded in FPGA and validated.



Inference in hardware

1. **Stream of data** that needs to be classified is sent to the FPGA.
2. **Feature mapping** is applied on input data.
3. **Full contraction** with the TTN architecture.
4. Retrieve **final probability** and classify sample.





Single node contraction: implement this operation with different degrees of parallelization to **trade-off between resources and latency**.

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

Repeat for each node and layers in the tree, executing independent computations parallelly.

FPGA resources

Dataset	TTN	DSP	BRAM	Latency
Iris	[2,4,1] PP	1%	2%	108 ns
Titanic	[2,4,8,1] FP	8%	19%	72 ns
LHCb	[2,4,8,8,1] FP	36.5%	84%	104 ns

- **Digital Signal Processors (DSP):** units that perform arithmetic operations inside FPGAs.

Used in the definition of the **single node contraction** to multiply 16-bit numbers.

- **Read-Only Memory (ROM):** memory blocks that can be configured and read from internal logic.

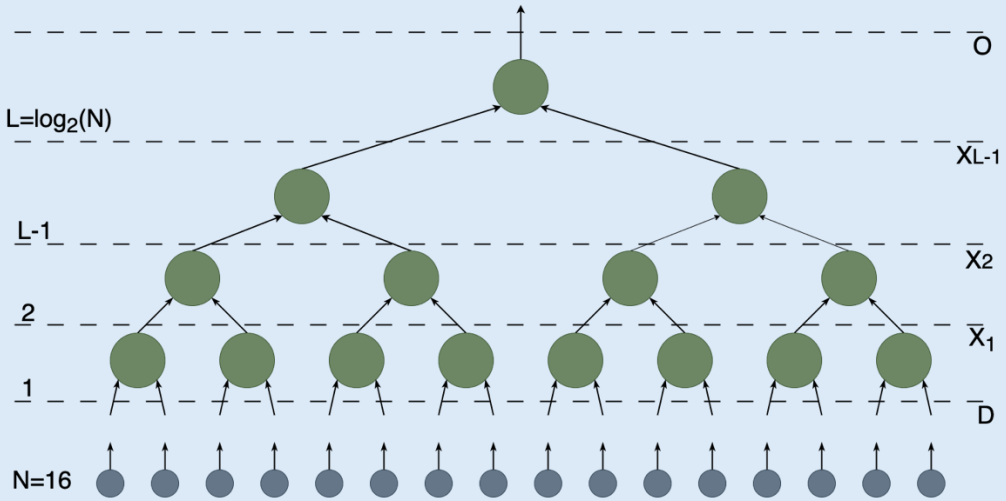
The **trained weights** of TTNs are written on FPGA register blocks, **stored in memory** and read by the logic for each operation.

- **Look-Up Tables (LUT):** necessary to implement complex functions with finite precision.

Used for building the static function needed for the **input feature mapping.**

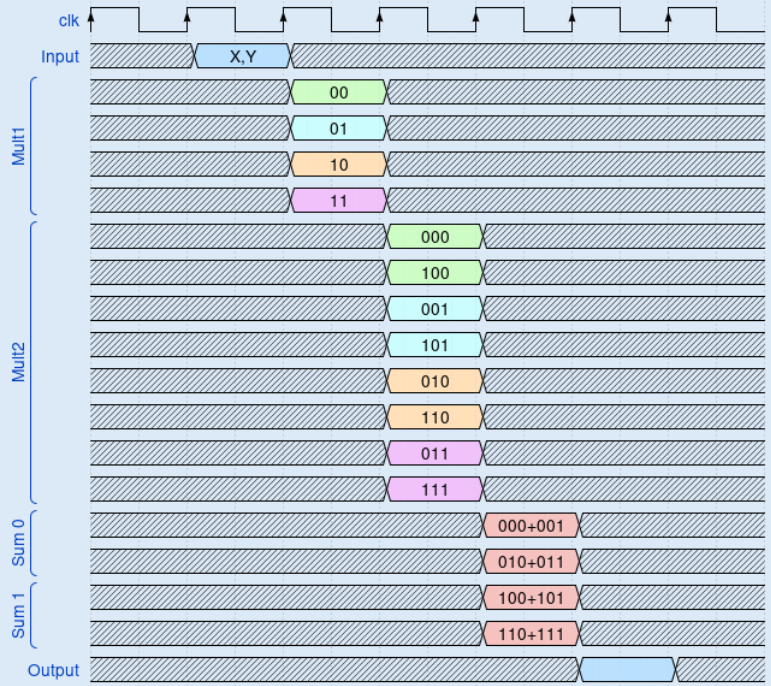
Full Parallel implementation

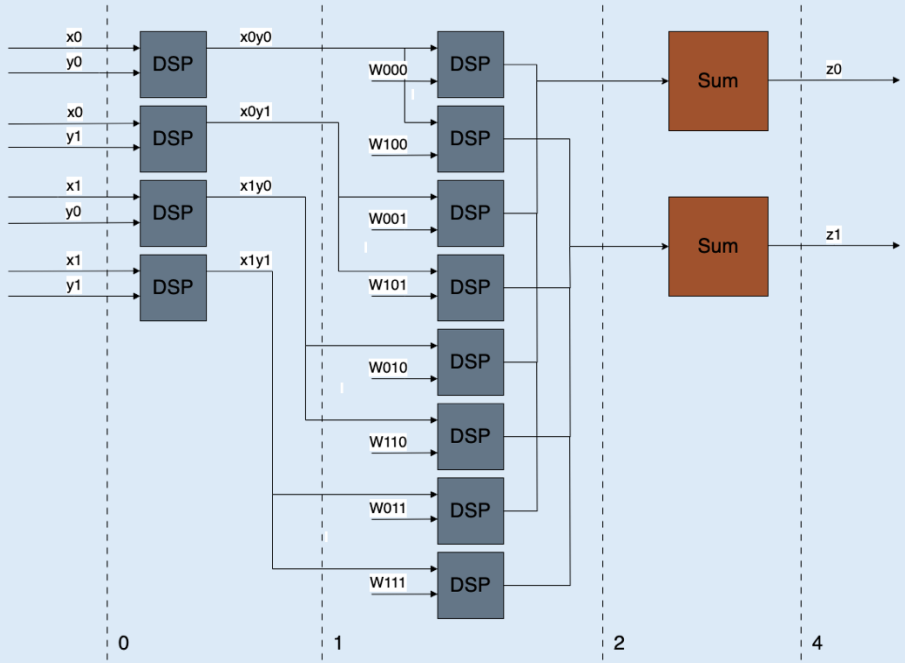
Maximize number of DSPs used and minimize total algorithmic latency



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

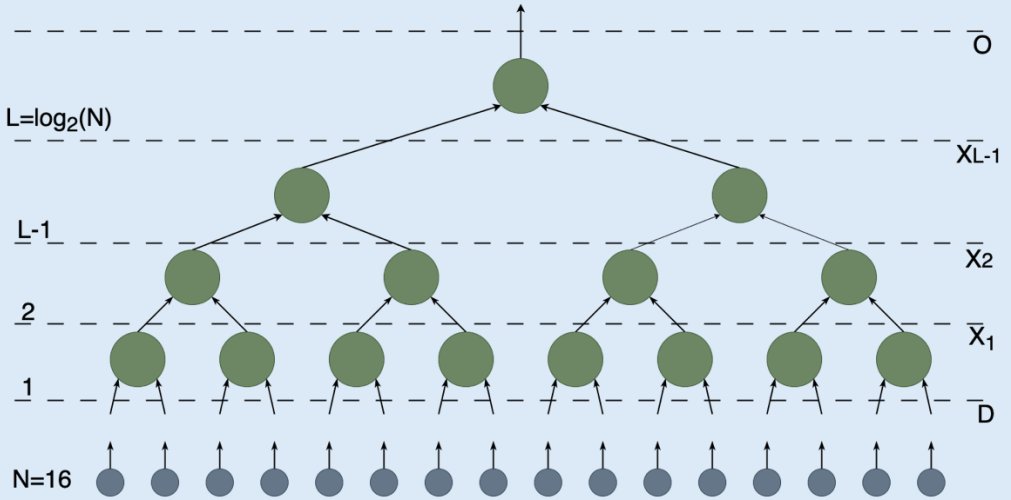
$$DSP = \sum_{i=1}^{L} \chi_{i-1}^2(\chi_i + 1)\frac{N}{2^i}$$





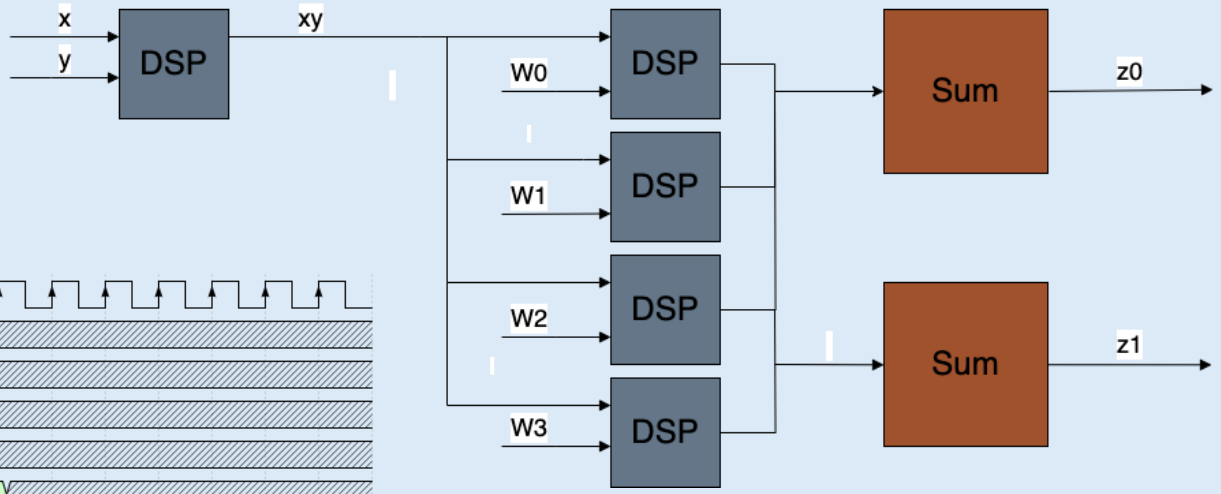
Partial Parallel implementation

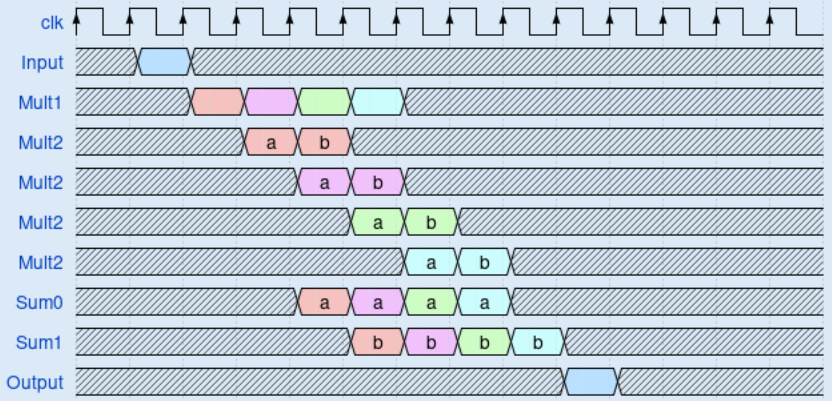
Reuse some of the DSPs, with a resulting increase in latency.



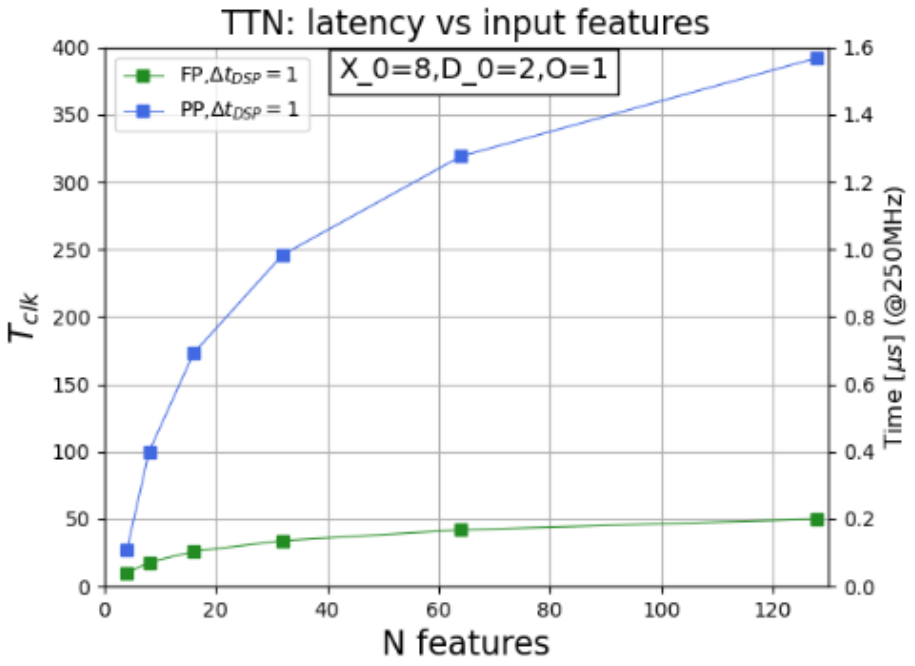
$$latency = \sum_{i=1}^{L} \chi_{i-1}^2 + \chi_i + 1$$

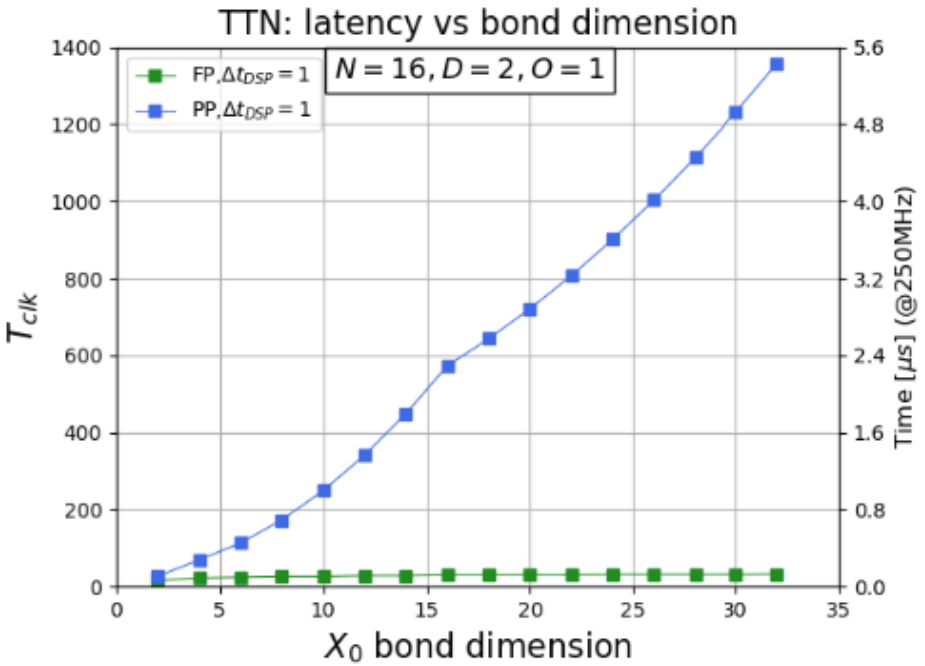
$$DSP = \sum_{i=1}^{L} (\chi_{i-1}^2 + 1)\frac{N}{2^i}$$

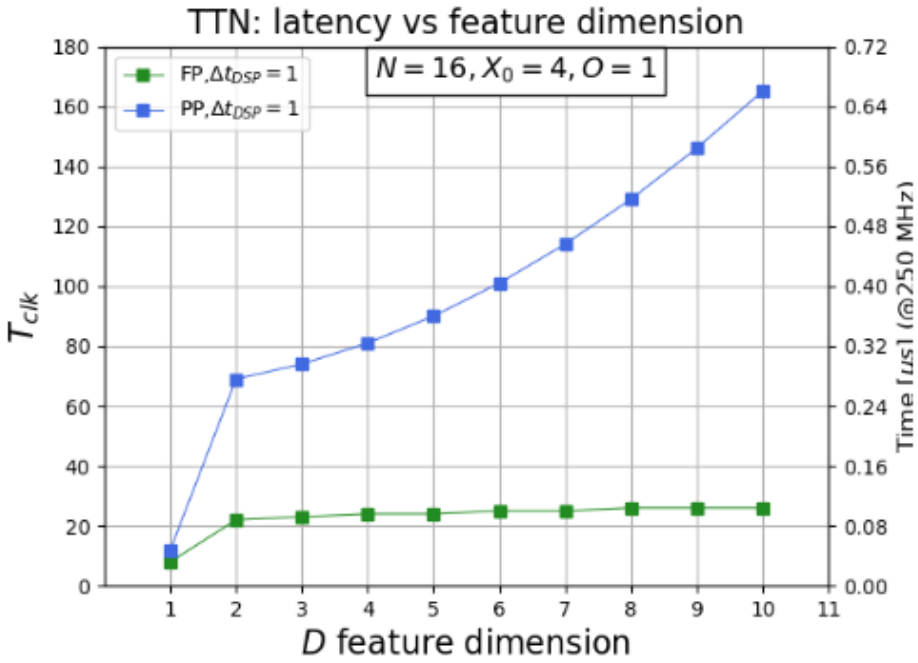


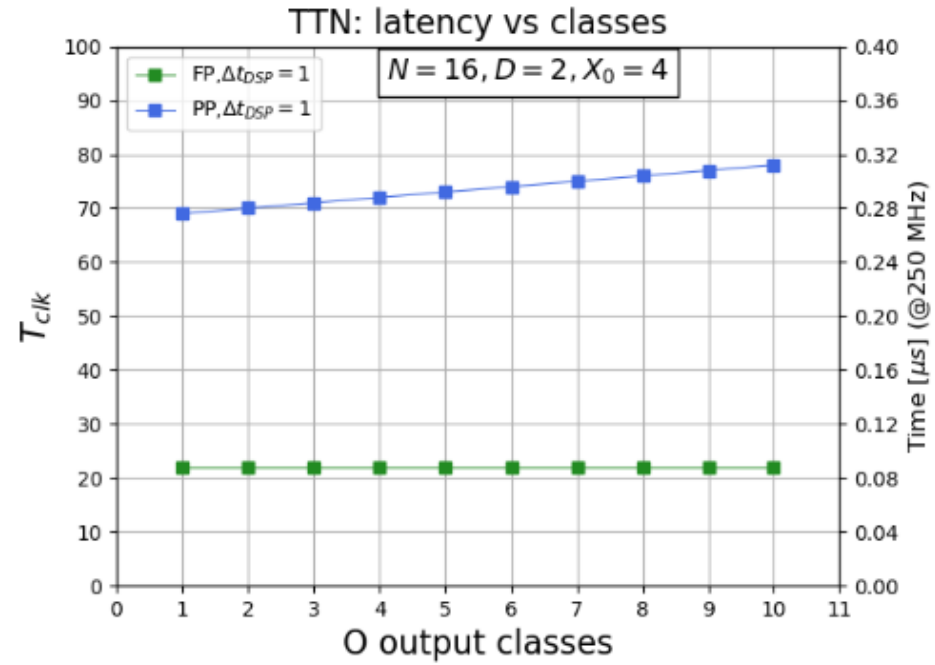


Latency

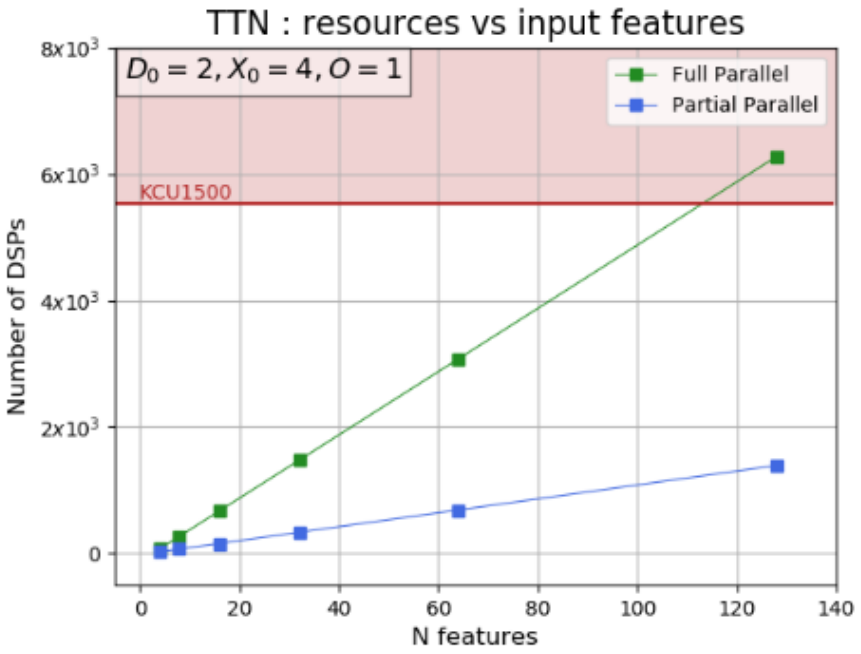

TTN: latency vs input features
X_0=8,D_0=2,O=1
FP,$\Delta t_{DSP}=1$
PP,$\Delta t_{DSP}=1$
T_{clk}
Time [μs] (@250MHz)
N features

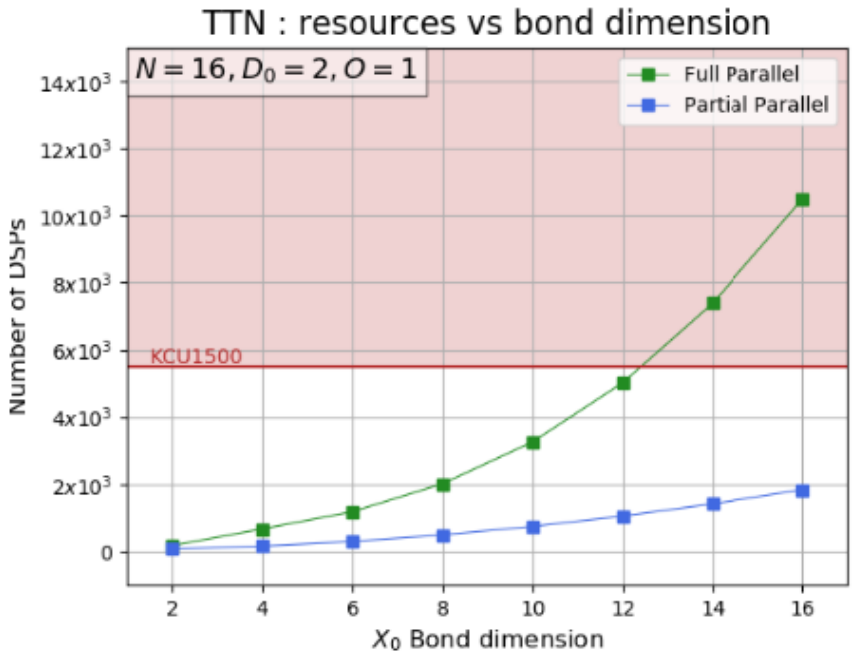

TTN: latency vs bond dimension
$N=16, D=2, O=1$
FP,$\Delta t_{DSP}=1$
PP,$\Delta t_{DSP}=1$
T_{clk}
Time [μs] (@250MHz)
X_0 bond dimension

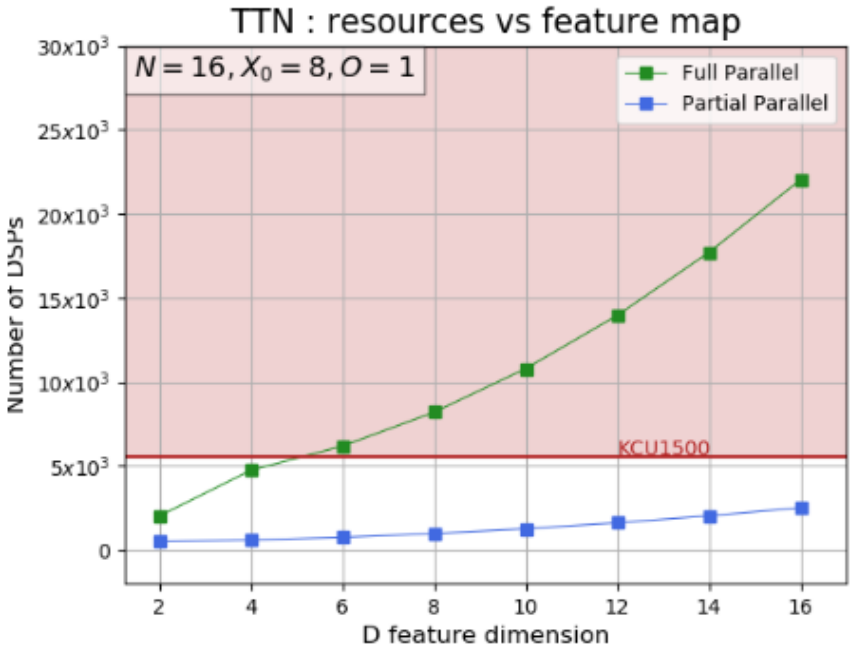

TTN: latency vs feature dimension
$N=16, X_0=4, O=1$
FP,$\Delta t_{DSP}=1$
PP,$\Delta t_{DSP}=1$
T_{clk}
Time [μs] (@250 MHz)
D feature dimension

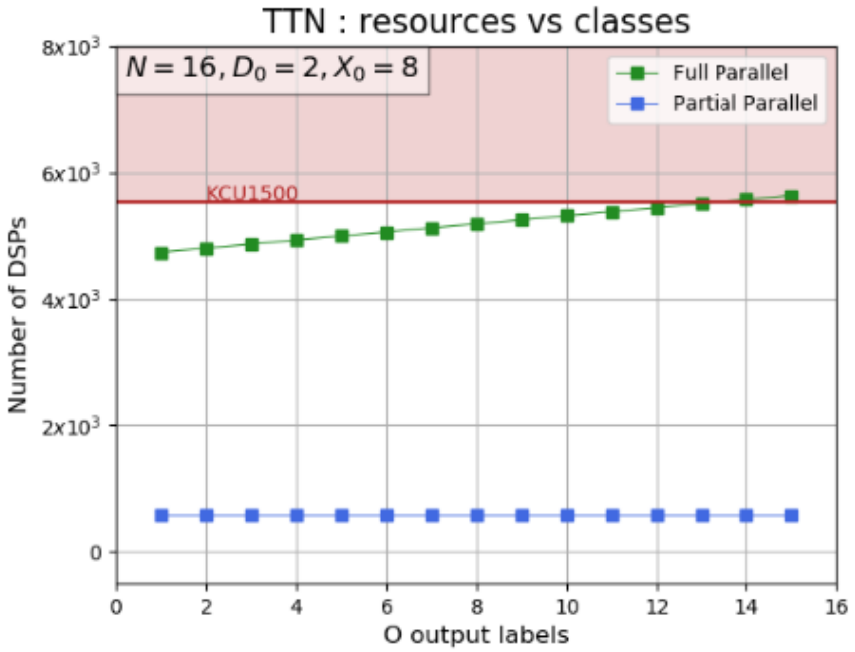

TTN: latency vs classes
$N=16, D=2, X_0=4$
FP,$\Delta t_{DSP}=1$
PP,$\Delta t_{DSP}=1$
T_{clk}
Time [μs] (@250 MHz)
O output classes

Resources

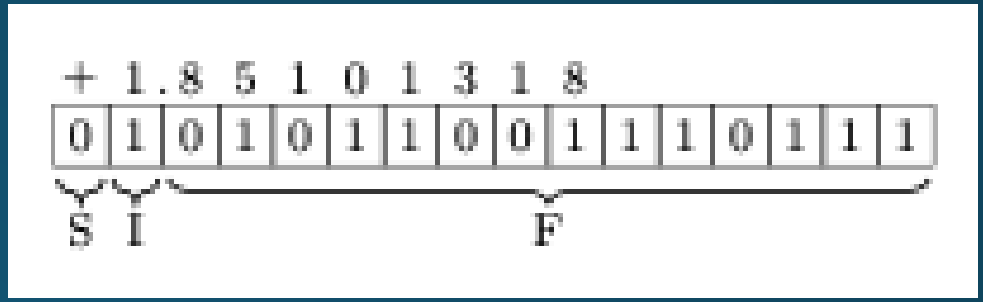

TTN : resources vs input features
$D_0 = 2, X_0 = 4, O = 1$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
N features

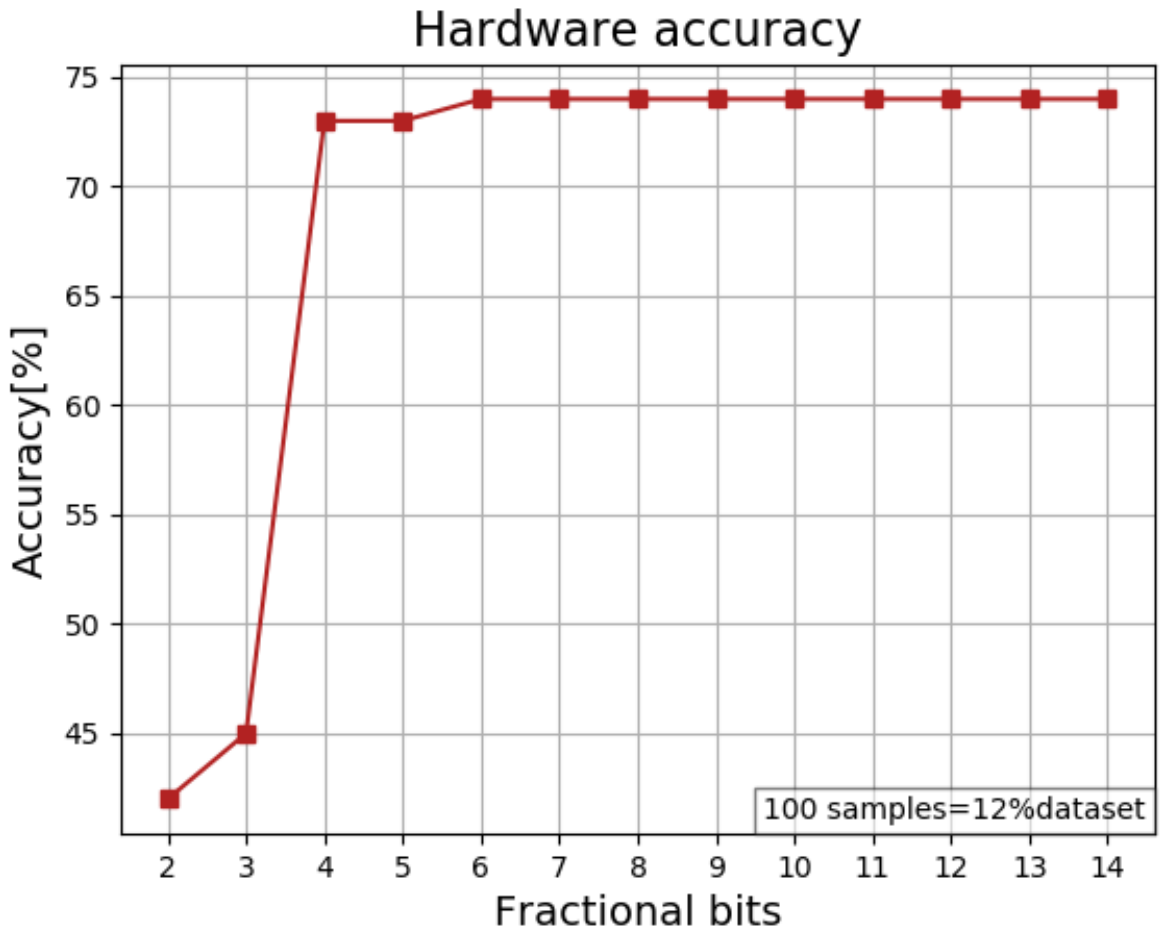

TTN : resources vs bond dimension
$N = 16, D_0 = 2, O = 1$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
X_0 Bond dimension

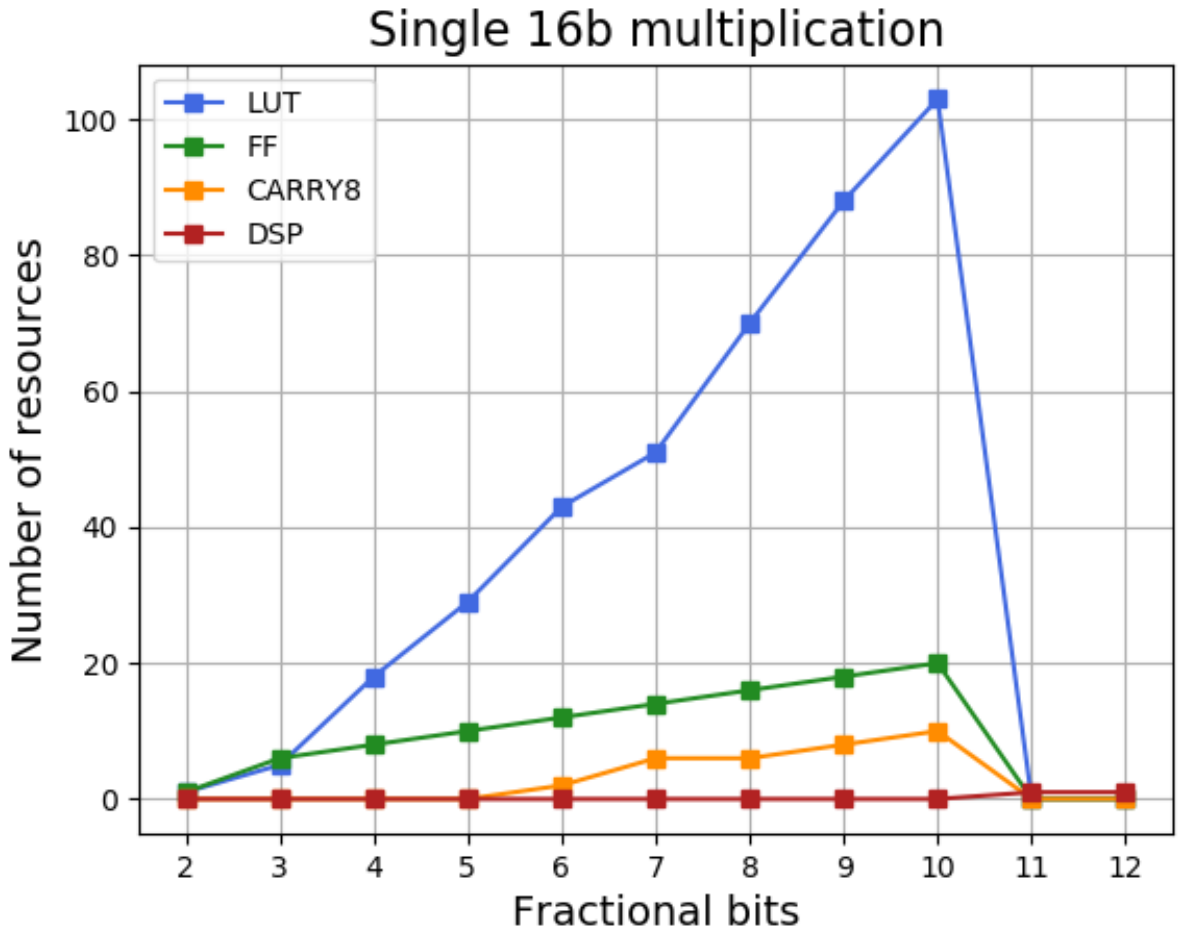

TTN : resources vs feature map
$N = 16, X_0 = 8, O = 1$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
D feature dimension


TTN : resources vs classes
$N = 16, D_0 = 2, X_0 = 8$
Full Parallel
Partial Parallel
KCU1500
Number of DSPs
O output labels

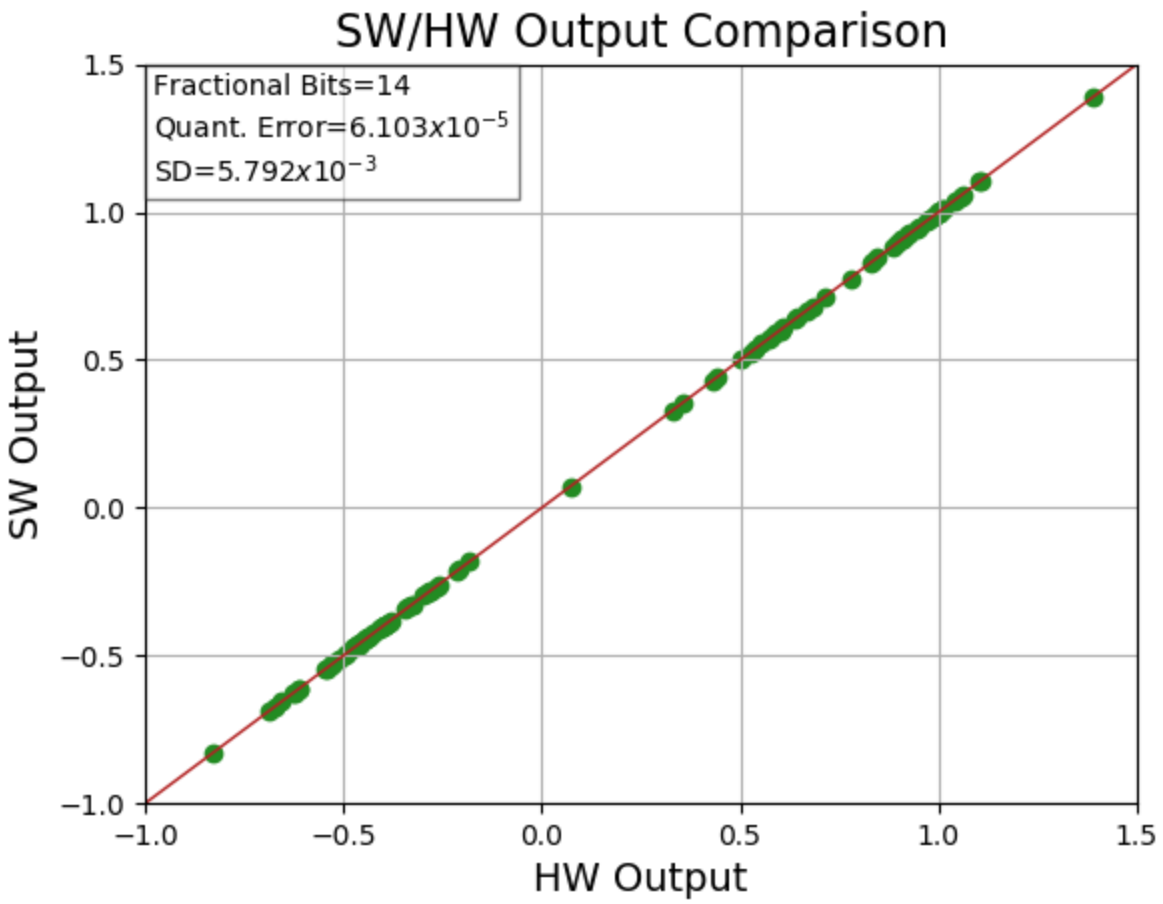
Quantization



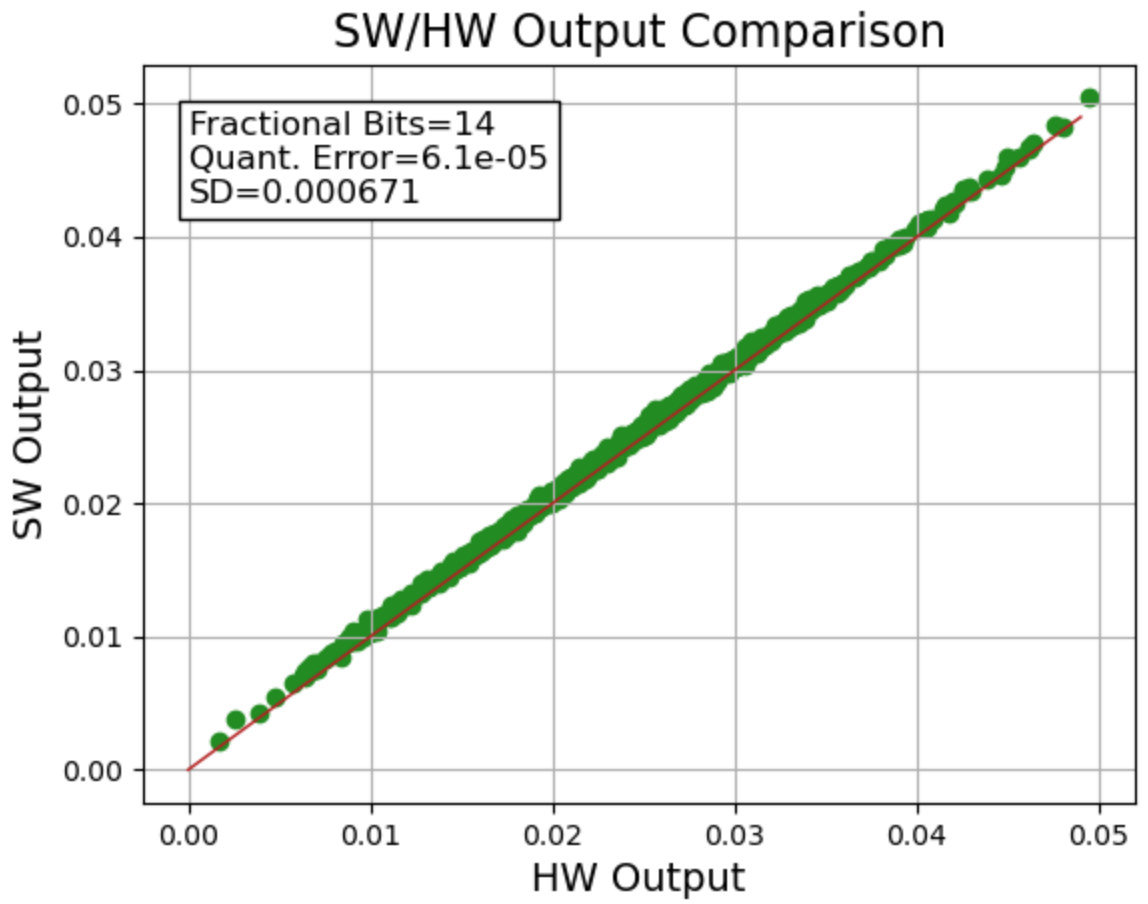




Inference validation



Titanic [2,4,8,1], N=8, 100 samples

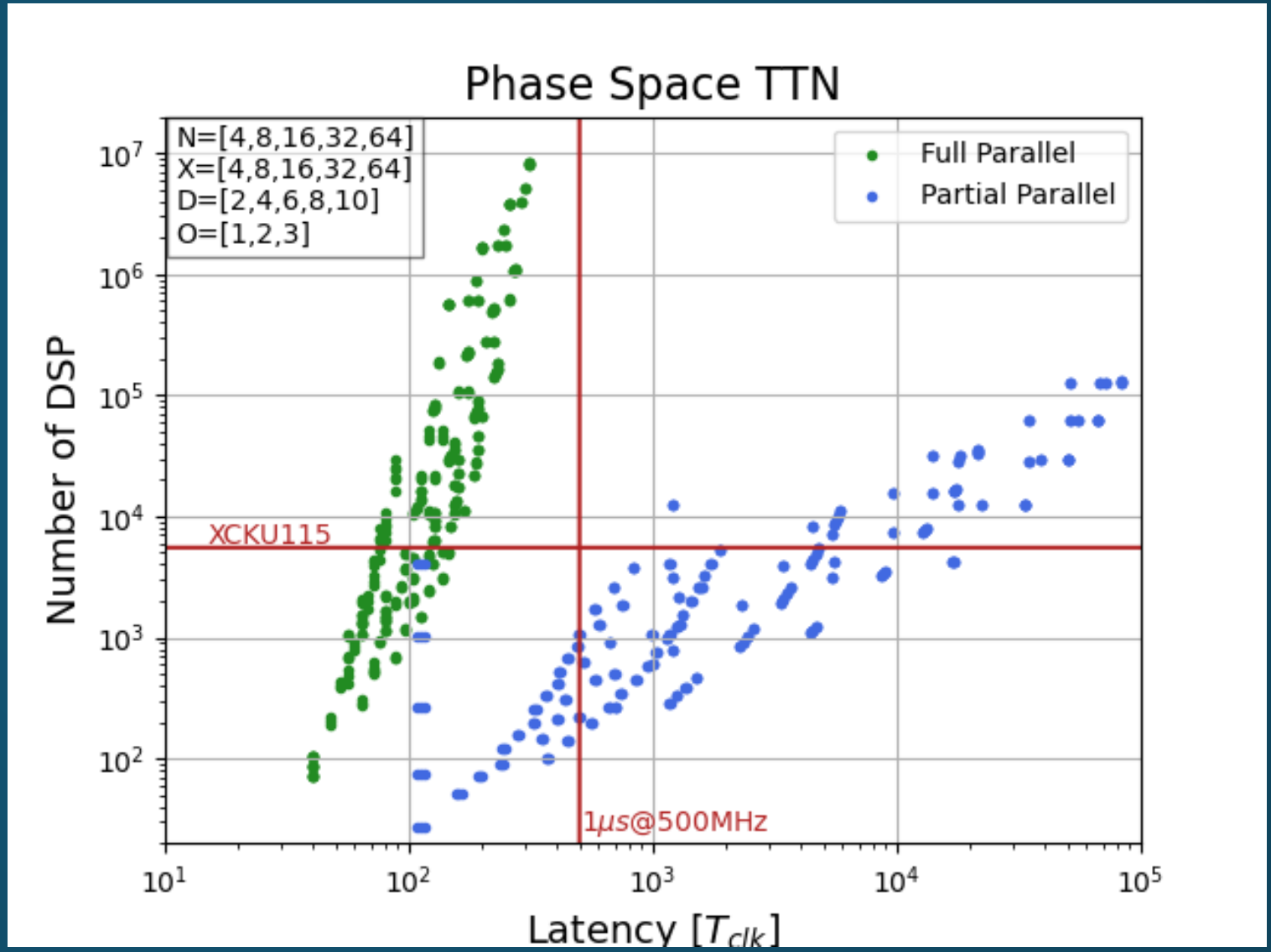


LHCb [2,4,8,16,1], N=16, 500 samples

Conclusions

- **Validated VHDL Firmware** for TTN inference on FPGA with **different degrees of parallelization.**
- Deterministic **projections of resources and latency** values for different TTN architectures.
- **Exactely reproduced** the behaviour of the b-tagging classifier studied in:

  Felser, T., Trenti, M., Sestini, L. et al. npj Quantum Inf 7, 111 (2021), Quantum-inspired machine learning on high-energy physics data



Possible future prospects

- Explore **different TN** examples (MPS, MERA, PEPS etc.) and tasks.
- **Improve firmware** with additional trade-off between latency and resource consumption.
- Move the project to **higher level programming language** (e.g. from VHDL to HLS4ML).
- Hardware inference on **Versal AI Engines** and compare with current implementation.
- Consider possibility of **training on FPGA** and possible applications.

Borella L., Coppi A., Pazzini J. et al.,
Ultra-low latency quantum-inspired machine learning predictors implemented on FPGA arXiv:2409.16075v2

Borella L., Coppi A., Pazzini J. et al.,
Quantum machine learning classifiers implemented on FPGA for ultra-low latency applications.
ICHEP 2024: https://indi.to/YXJT9

Thank you for your attention!