FQTree: Fine-grained Quantization and Hardware Generation of Boosted Decision Trees

Zhiqiang (ZQ) Que, Chang Sun, Haiyang Wang, Dinesh Pamunuwa, Roshan Weerasekera, Qijia Tang, Bakhtiar Zadeh, Wayne Luk, Maria Spiropulu

University of Bristol, Caltech, Imperial College London
ASAP 2026

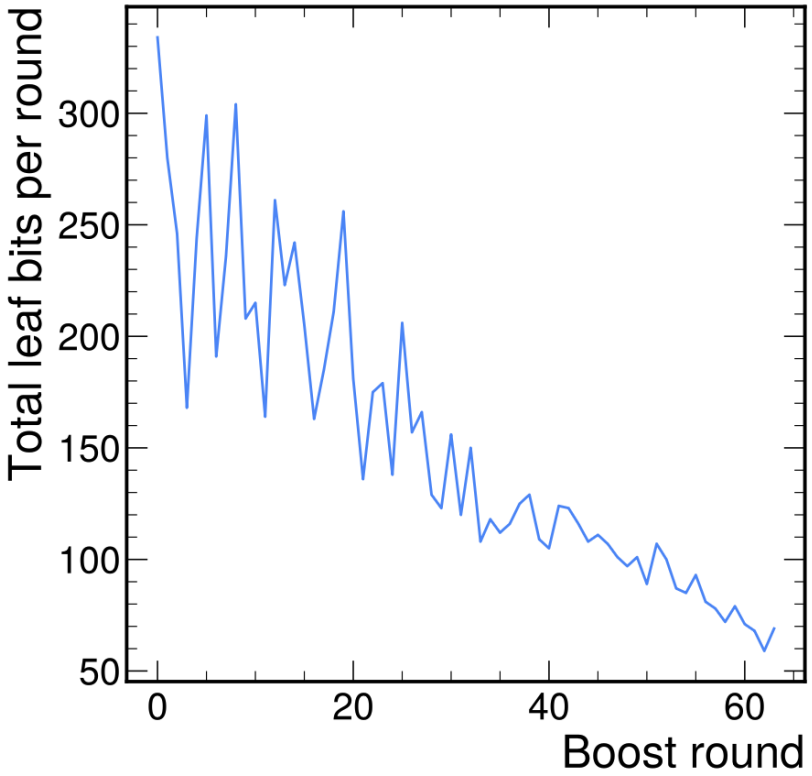
GitHub: https://github.com/ecs-bristol/FQTree

BDTs need hardware-aware quantization

- Why BDTs?
  - strong accuracy on tabular data
  - compact, deterministic inference
  - well suited to latency-critical systems

- **Why quantization of BDTs is difficult?**
  - Feature and threshold errors can change the selected leaf
  - Leaf precision sets datapath width and LUT cost
  - Uniform precision cannot balance accuracy and hardware cost

Precision demand varies across boosting rounds



- **Early trees**
  - Capture the dominant structure
  - Need more range and more bits
- **Late trees**
  - Refine residual errors
  - Can be quantized more aggressively

- **One fixed-point format forces every tree to pay for the range of the early trees.**

FQTree Overview

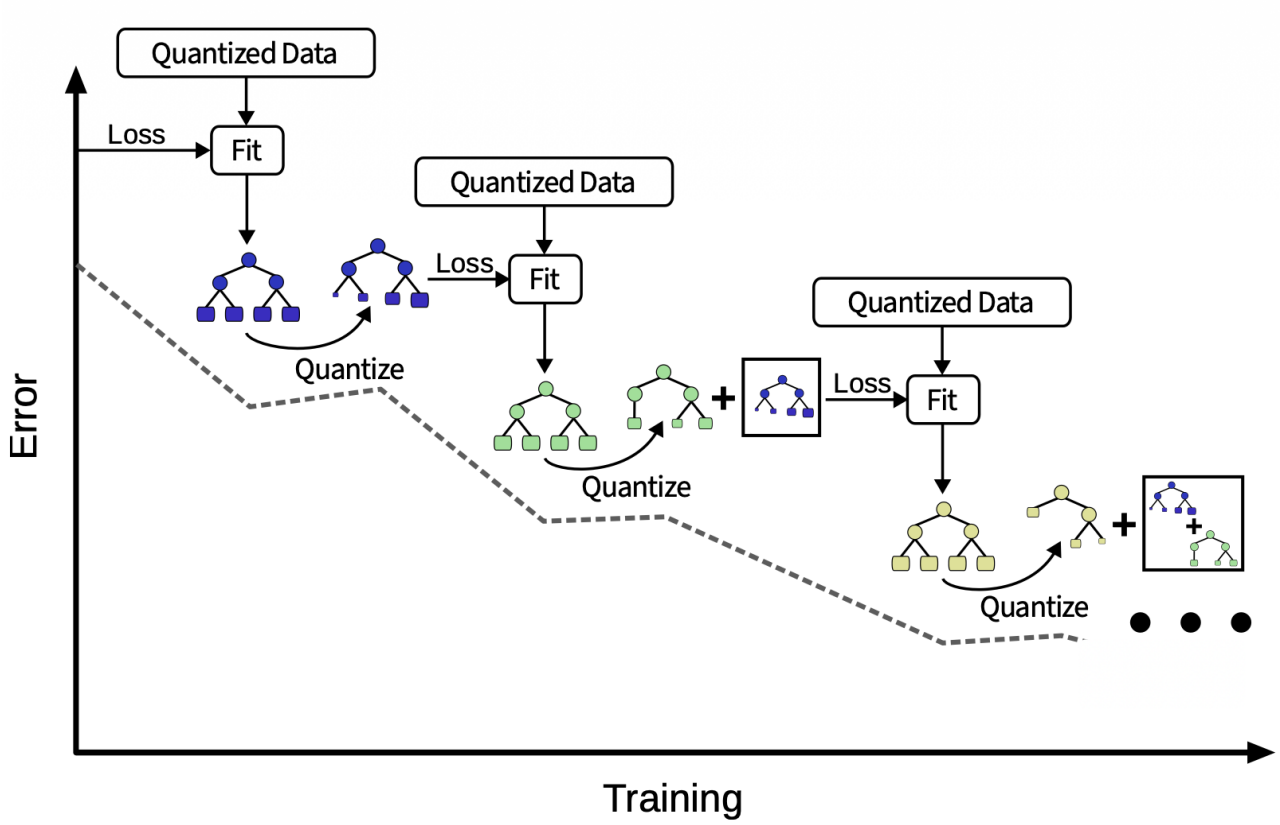
- Efficient FPGA Deployment Challenge
  - BDTs offer strong accuracy for latency-critical inference
  - uniform or manually tuned precision wastes resources or degrades accuracy

- Our innovations
  - 1. Fine-grained mixed-precision quantization-aware training
    - **quantization-aware boosting:** later trees adapt to quantization errors
    - hardware-aware leaf quantization with **tree-wise shifts and bias folding**
  - 2. QXGB: compiler-based scalable flow for automatic RTL/HLS generation

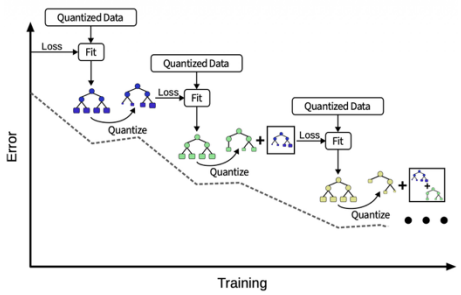
- Results
  - **26–57% fewer LUTs** than state-of-the-art FPGA BDT designs
  - **1–2-cycle latency** with **0 DSP blocks**
  - best accuracy among state-of-the-art BDT designs

Key idea 1: Quantization-aware boosting

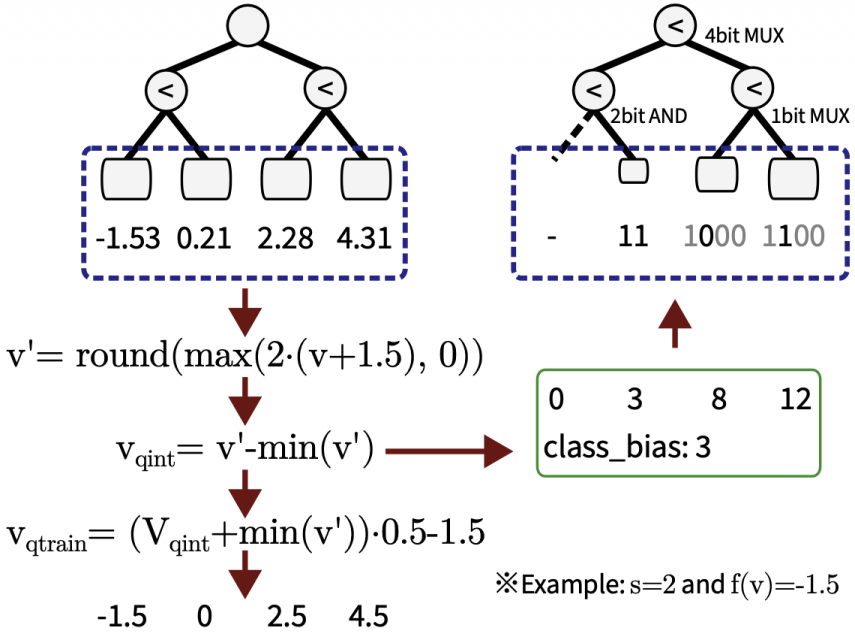
- FQTree brings quantization inside the boosting loop



Key idea 1: Quantization-aware boosting

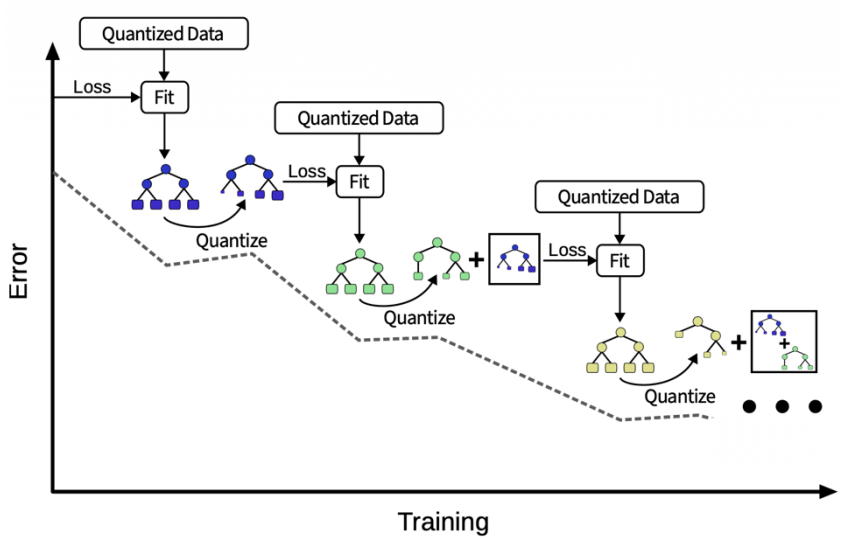


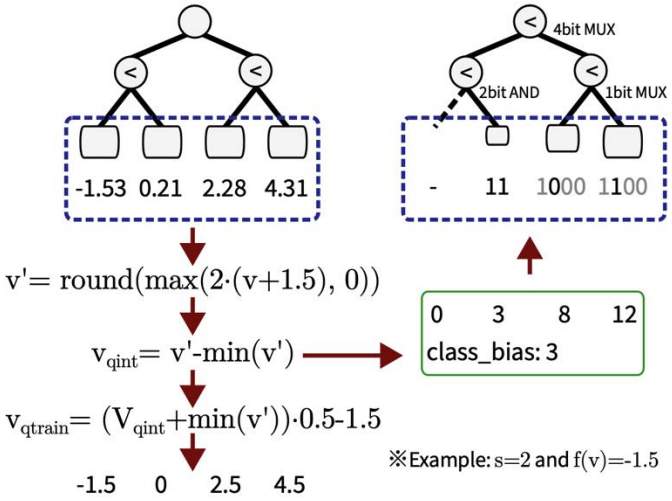
- FQTree brings quantization inside the boosting loop
- It quantizes every newly fitted tree immediately
  - hardware-aware leaf quantization
  - a tree-wise shift makes integers non-negative; bias folding



Key idea 1: Quantization-aware boosting

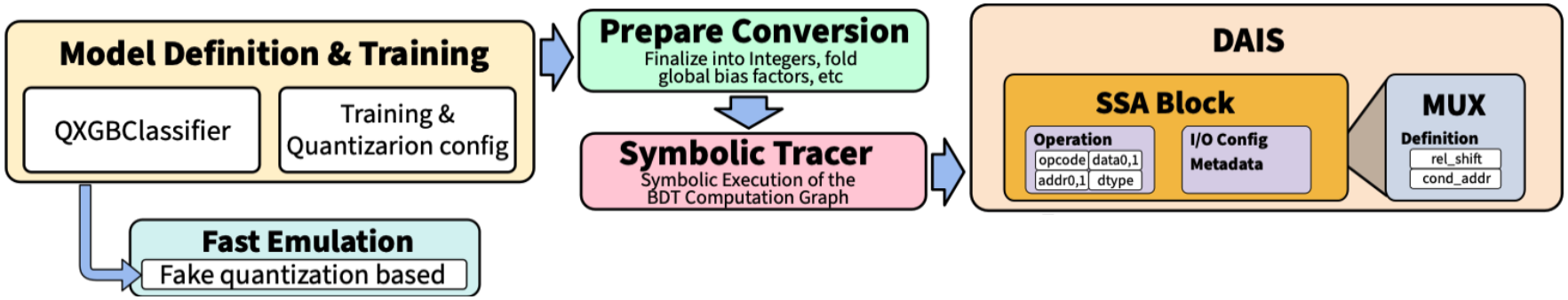
- FQTree brings quantization inside the boosting loop
- It quantizes every newly fitted tree immediately
  - hardware-aware leaf quantization
  - a tree-wise shift makes integers non-negative; bias folding
  - **Later trees** compensate for errors introduced by **earlier quantized trees**
- Unlike PTQ, in FQTree, the model **adapts to quantization** during training





Key idea 2: **QXGB** framework lowers quantized trees to hardware

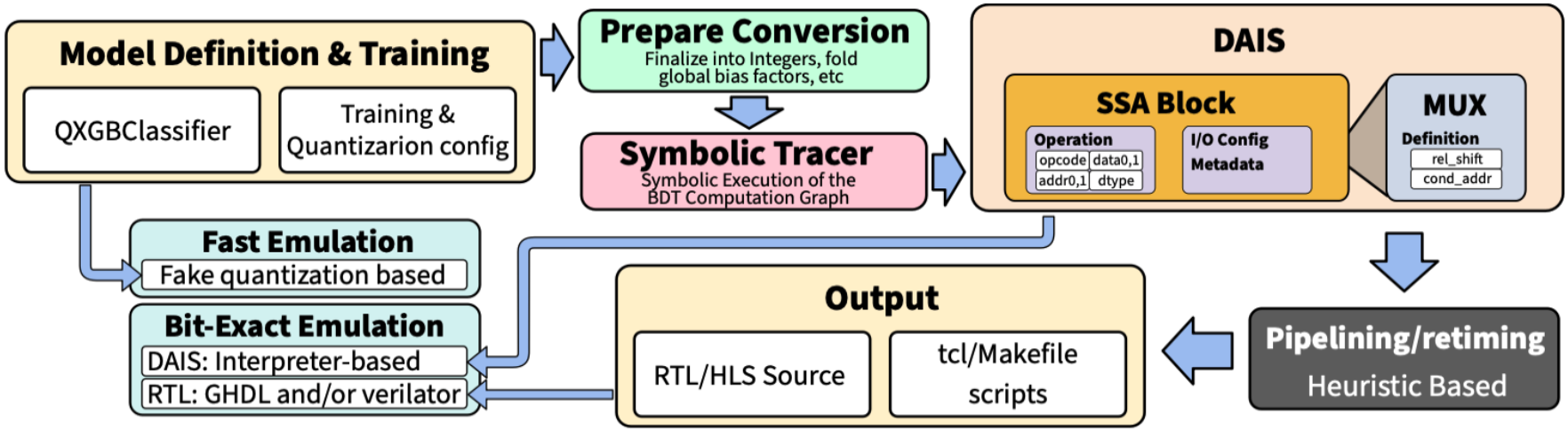
- trace the BDT into the extended DAIS IR [R1]



[R1] C. Sun et al., “da4ml: Distributed arithmetic for real-time neural networks on fpgas,” ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2026.

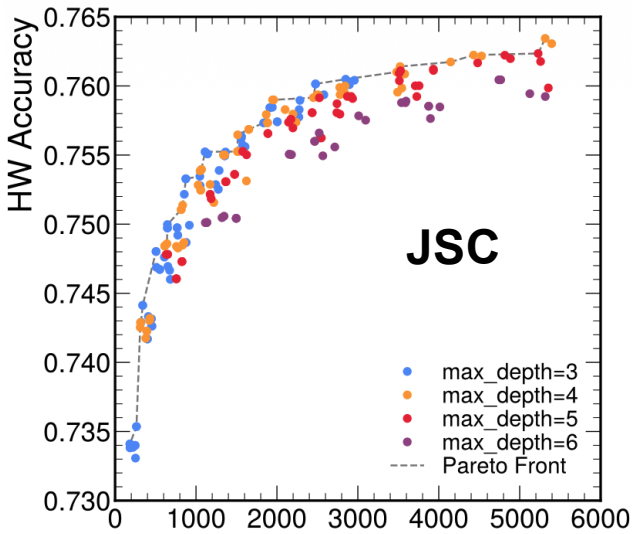
Key idea 2: **QXGB** framework lowers quantized trees to hardware

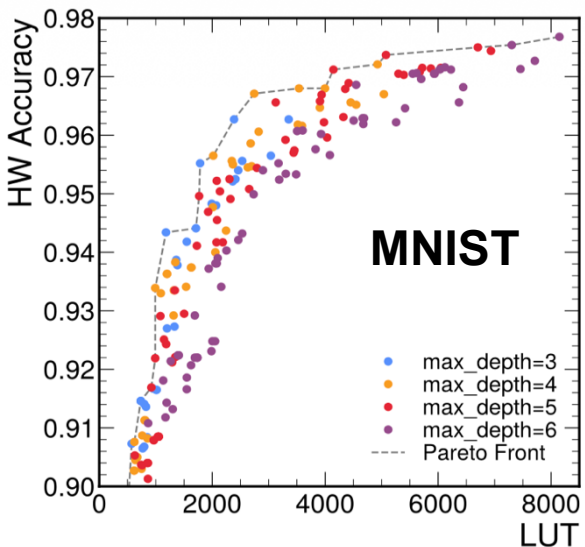
- trace the BDT into the extended DAIS IR [R1]
- DAIS IR -> RTL/HLS
- validate with fast emulation (fake-quantized) and bit-exact emulation



Evaluation: a selectable Pareto frontier

- AMD VU13P; Vivado 2025.1 post-route;
- Bit-exact RTL accuracy
- A full design space - not one fixed point
- Deeper trees can raise the accuracy ceiling, but consume more LUTs.
- Choose a point for each resource budget





Evaluation: Performance and resource comparison

Task	Implementation	Acc. ↑	Latency [cycles]	LUT	DSP	FF	F_{max}(MHz)	II
JSC HLF	**FQTree (This work)**	75.7%	2 (4.0 ns)	1,652	0	446	505	1
	FQTree (This work)	74.8%	1 (2.0 ns)	548	0	146	499	1
	PTQ (**This work**)	75.6%	2 (4.0 ns)	2,194	0	361	501	1
	PTQ (**This work**)	74.7%	1 (2.0 ns)	642	0	143	502	1
	FPGA'25 [6] TreeLUT	75.6%	3[a] (3.9 ns)	2,234	0	347	735	1
	FPGA'25 [6] TreeLUT	74.6%	2[a] (2.1 ns)	796	0	74	887	1
	TCAS-I'25 [7] QBDT-8bit	75%[c]	5 (7.1 ns)	6,500[b]	-	-	670	1
	TCAS-I'25 [7] QBDT-1bit	73.5%	1 (1.4 ns)	906	-	-	724	1
	JINST'20 [5] Conifer	73.8%	12 (60 ns)	96,148	-	42,802	~ 200	1
MNIST	**FQTree (This work)**	97.7%	2 (4.0 ns)	8,147	0	2,873	504	1
	FQTree (This work)	96.7%	2 (3.5 ns)	2,744	0	1,156	576	1
	FQTree (This work)	95.6%	2 (3.2 ns)	2,019	0	1,150	633	1
	PTQ (**This work**)	96.6%	2 (3.9 ns)	4,439	0	1,374	518	1
	PTQ (**This work**)	96.0%	2 (3.4 ns)	3,377	0	1,478	584	1
	FPGA'25 [6] TreeLUT	96.6%	3[a] (3.6 ns)	4,478	0	997	791	1
	FPGA'25 [6] TreeLUT	95.6%	3[a] (3.2 ns)	3,499	0	759	874	1
	JSPS'20 [24] POLYBiNN	97.2%	900 (90 ns)	109,653	-	-	100	-
	JSPS'20 [24] POLYBiNN	95.6%	700 (70 ns)	9,943	-	-	100	-
NID	**FQTree (This work)**	93.1%	1 (1.9 ns)	157	0	155	524.	1
	FQTree (This work)	92.8%	1 (1.4 ns)	83	0	76	740	1
	FQTree (This work)	91.7%	1 (1.1 ns)	38	0	45	871	1
	FPGA'25 [6] TreeLUT	92.7%	2[a] (2.7 ns)	345	0	33	681	1
	FPGA'25 [6] TreeLUT	91.5%	2[a] (1.7 ns)	89	0	19	1047	1
	TCAS-I'25 [7] QBDT-8bit	92%[c]	5 (7.1 ns)	1,800[b]	-	-	714	1
	TCAS-I'25 [7] QBDT-1bit	91.5%	1 (1.4 ns)	170	-	-	724	1

26%

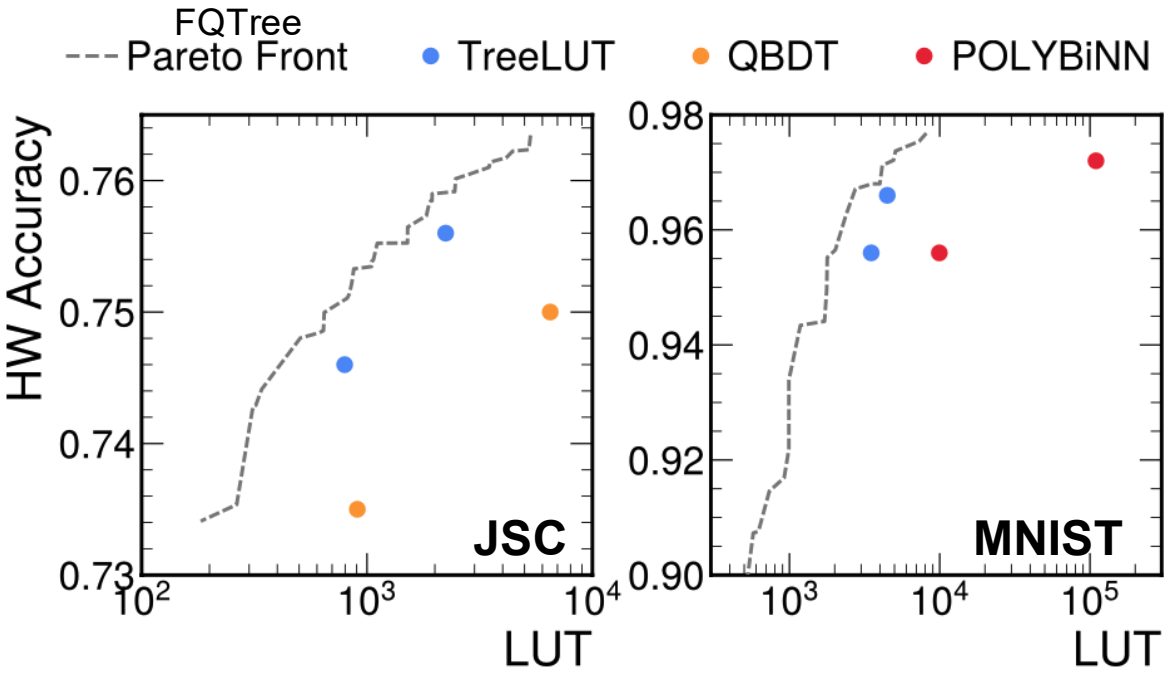
42%

57%

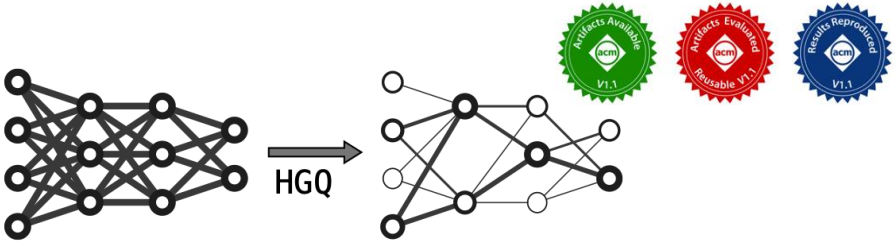
26-57% fewer LUTs

Evaluation: Extends the accuracy–LUT Pareto frontier

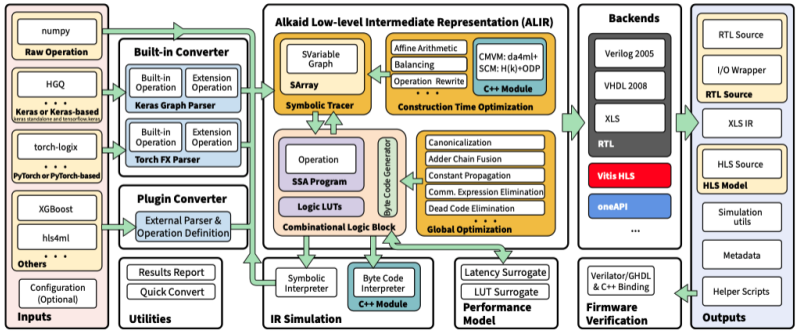
- **New** Pareto Frontier
- Better accuracy with lower LUTs
- Better accuracy-LUT trade-offs than the compared prior designs



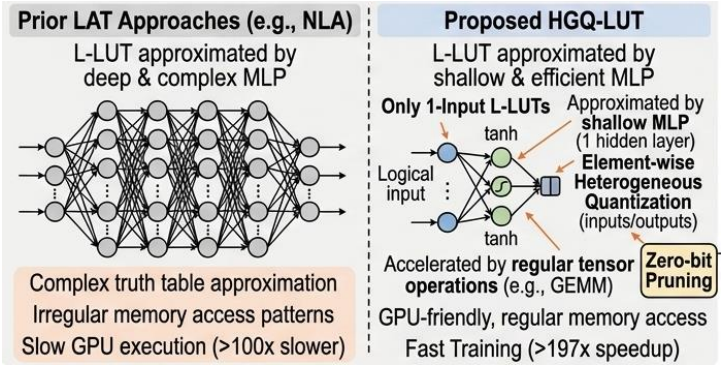
FQTree in Our Broader Research Programme



HGQ: High Granularity Quantization for Real-time Neural Networks on FPGAs, **FPGA'26**
A gradient-based quantization-aware training framework
https://github.com/calad0i/HGQ2



Alkaid: A Compiler Framework for Ultra-Low-Latency Kernels on Hardware, **ICCD'26**, **https://github.com/calad0i/alkaid**

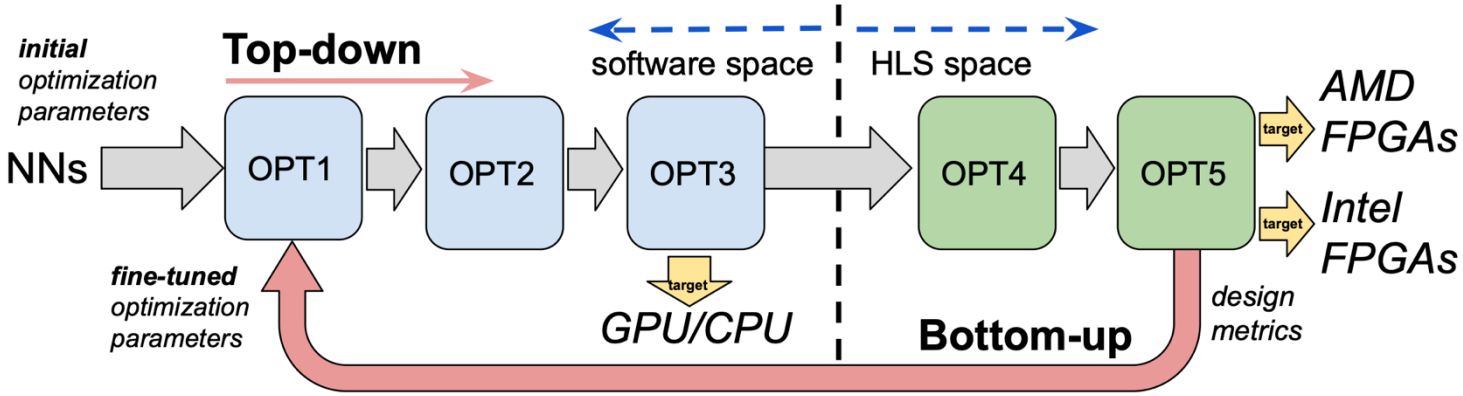


HGQ-LUT: Fast LUT-Aware Training and Efficient Architectures for DNN Inference (**FCCM'26, Best Paper Award**)

FQTree: **BDT-specific** co-design through quantization-aware boosting and automated hardware generation.

Future Work

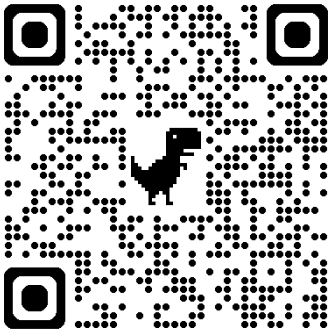
- Trustworthiness-aware BDT inference
  - uncertainty estimation and monitoring, robust low-precision deployment
- Larger models and broader FPGA platforms
  - automated accuracy-resource exploration
- Cross-stage design-flow automation using MetaML [R2]
  - reuse FQTree's HW-aware representation



[R2] Z. QUE et al., "MetaML-Pro," TRETS, 2026.

Summary

- FQTree: Fine-Grained QAT for BDT
  - **quantization-aware boosting**
  - hardware-aware leaf quantization
- Automated hardware generation
  - **QXGB** lowers BDTs into extended DAIS IR
  - end-to-end automation with RTL generation
  - bit-exact validation
- Evaluation: compared to state-of-the-art
  - open source: **https://github.com/ecs-bristol/FQTree**
  - **26-57% fewer LUTs; 1-2 cycles; 0 DSPs**
  - best accuracy among state-of-the-art designs



My webpage