XILINX
ALL PROGRAMMABLE.

FINN: An End-to-End Framework for Accelerating Quantized Neural Network Inference on FPGAs

Michaela Blott, Nicholas Fraser, Giulio Gambardella, Thomas Preusser, Yaman Umuroglu, Andrea Solazzo, Ken O'Brien, Julian Faraone, Jiang Su, Philip Leong, Magnus Jahre, Kees Vissers

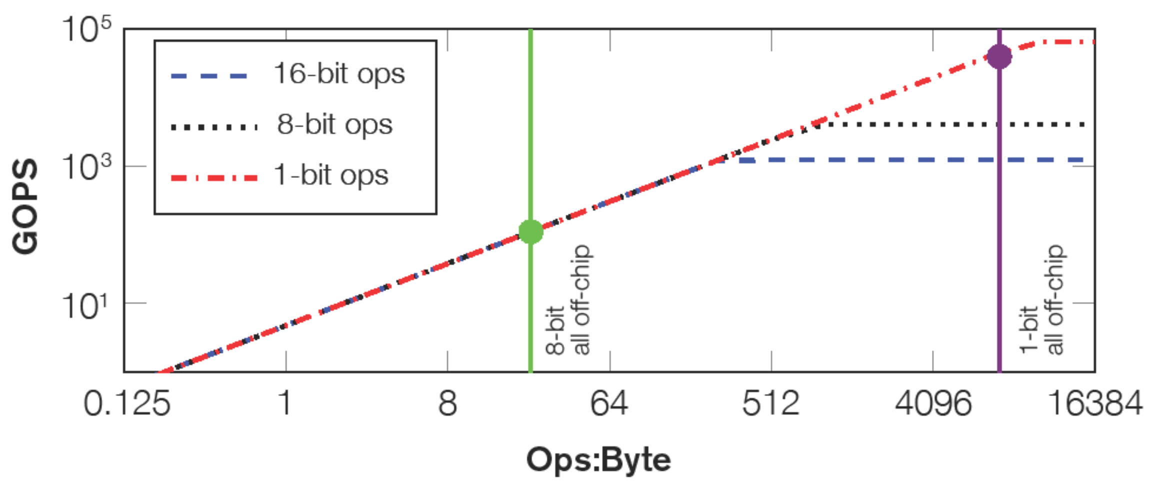
1 Quantizing Neural Networks works - high accuracy on state of the art benchmarks

- Trained via back-propagation on GPU, weights constrained during training
- Leveraging different quantization strategies
- Convolutional, fully-connected, pooling and batchnorm layers
- Competitive accuracy for image classification tasks

Topology	[GOPS/frame]	Top5 32/32 [%]	Top5 n/m [%]	Top5 bin or ternary [%]
GoogLeNet	3.0	10.7	10.72	15.12
ResNet-18	3.6	10.67	10.90	12.80
ResNet-34	7.2	8.66	8.65	9.63
ResNet-50	7.6	7.13	7.55	8.20
VGG-16	30.8	9.9	9.7	-

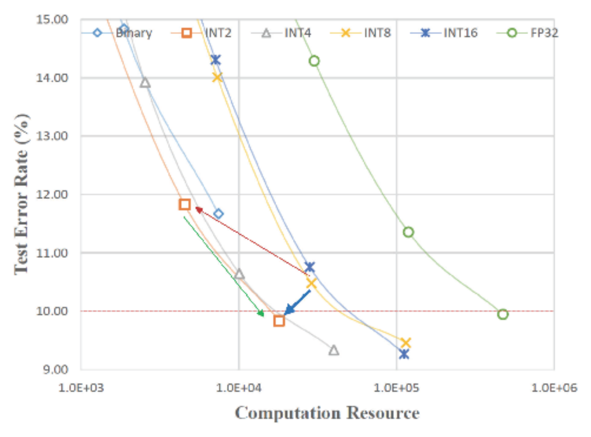
2 Hardware Cost of Different Precisions

- Hardware cost reduces 40-100x by going from 32b float to 1b binary, thereby performance scales
- ZU19EG: 66 TOPS for binary, 4 TOPS for Int8, 0.3 TOPS for SP
- Power consumption greatly reduced: 100x for 32SP to Int8
- No need for external memory => on-chip only

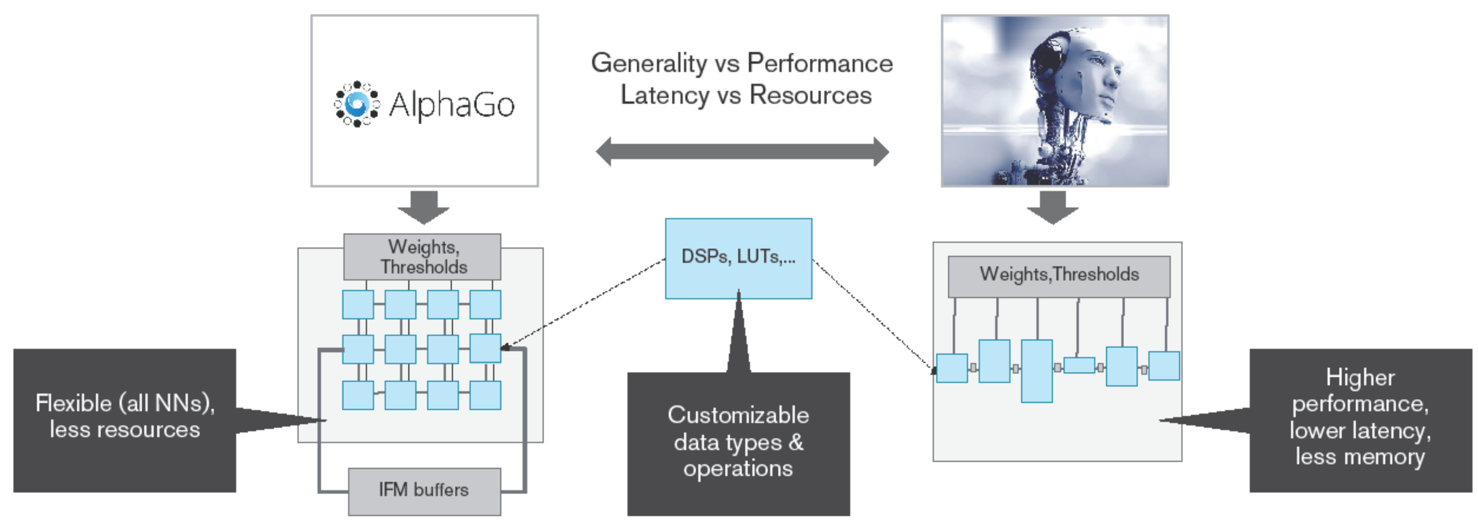


3 Accuracy–Hardware Cost Trade-off

- Just reducing precision, reduces hardware cost & increases error
- Recuperate accuracy by retraining & increasing network size
- 1b, 2b and 4b provide Pareto-optimal solutions

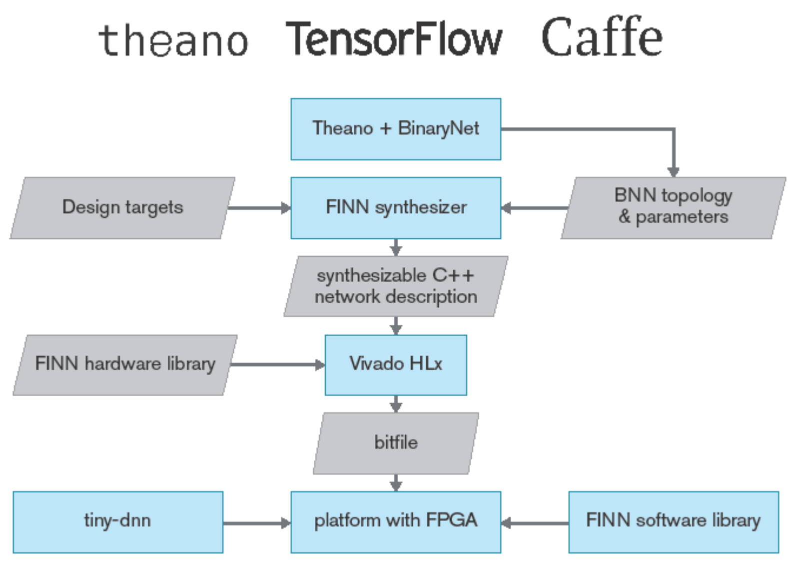


4 Customized Hardware Architectures for different topologies to minimize latency & wasted execution resources

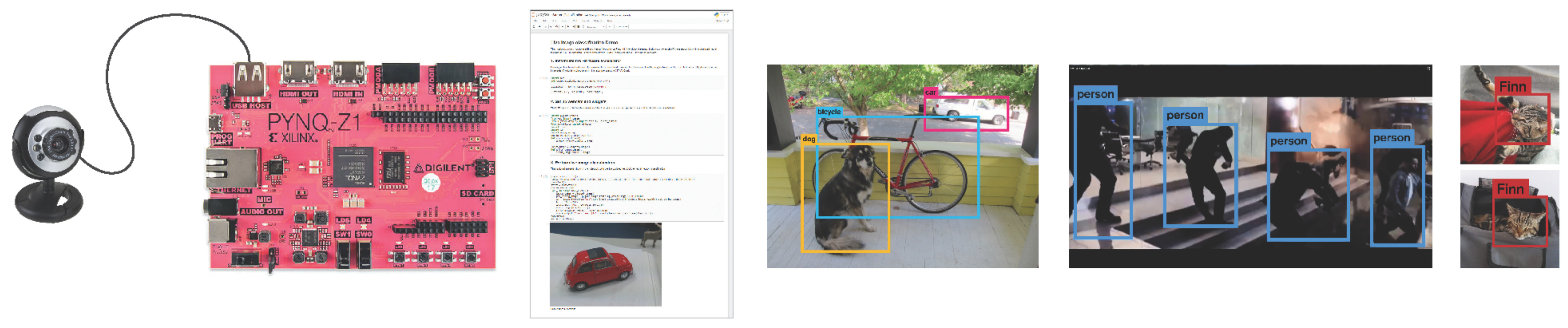


5 End-to-End Tool Flow with Automated Optimizations

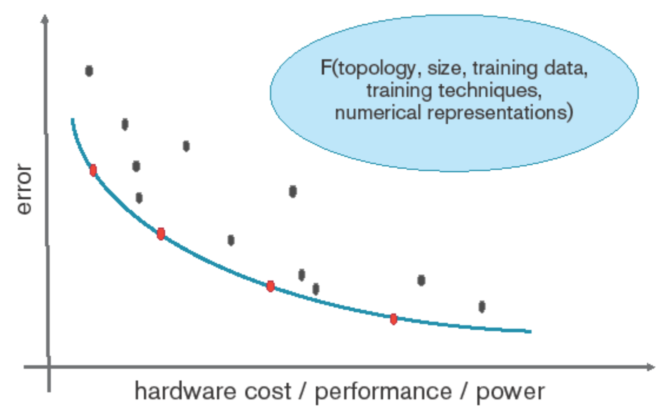
```
void DoCompute(ap_uint<64> * in, ap_uint<64> * out) {
#pragma HLS DATAFLOW
   stream<ap_uint<64> > memInStrm("memInStrm");
   stream<ap_uint<64> > InStrm("InStrm");
       .
       .
       .
   stream<ap_uint<64> > memOutStrm("memOutStrm");

   Mem2Stream<64, inBytesPadded>(in, memInStrm);
   StreamingMatrixVector<L0_SIMD, L0_PE, 16, L0_MW, L0_MH, L0_WMEM, L0_TMEM>
       (InStrm, inter0, weightMem0, thresMem0);
   StreamingMatrixVector<L1_SIMD, L1_PE, 16, L1_MW, L1_MH, L1_WMEM, L1_TMEM>
       (inter0, inter1, weightMem1, thresMem1);
   StreamingMatrixVector<L2_SIMD, L2_PE, 16, L2_MW, L2_MH, L2_WMEM, L2_TMEM>
       (inter1, inter2, weightMem2, thresMem2);
   StreamingMatrixVector<L3_SIMD, L3_PE, 16, L3_MW, L3_MH, L3_WMEM, L3_TMEM>
       (inter2, outstream, weightMem3, thresMem3);
    StreamingCast<ap_uint<16>, ap_uint<64> >(outstream, memOutStrm);
    Stream2Mem<64, outBytesPadded>(memOutStrm, out);
}
```



6 Demo: ImageNet Classification, Bounding Boxes 30fps, <25ms, 2.5Watt



7 Benefits of **Customizable Hardware Accelerators** in form of **Performance**, **Low Latency**, and **Power Efficiency** Enabled through a **Software Framework**



Experimental Results

	Speed [TOP/s]	Latency [ms]	Power [Watt]	[GOPS/ Watt]	Platform
Bin. MLP	0.97	0.102	2.5	487	PYNQ
Bin. CNV	2.3	0.22	10.7	271	PYNQ-ZU3EG
Bin. MLP	5.1	0.08	11.8	432	PYNQ-ZU3EG
Bin. MLP	34.2	0.001	42	834	KU115
1b-2b Dorefanet	9.3	0.86	42	226	KU115
1b-2b Dorefanet	48.7 (23.6kfps)	~0.14	~100	~487	AWS F1

Experimental datasets: MNIST, CIFAR-10, ImageNet respectively
AWS estimated performance
GPU comparison: theoretical 290GOPS/watt for int8 on P4 theoretical

QNN on FPGAs offer…
- Extreme performance through customization
- Low latency through dataflow – no batching needed
- Flexibility
- Low power total solution: keep data on chip, compute at reduced precision

Get started with FINN & PYNQ
- www.pynq.io
- https://github.com/Xilinx/BNN-PYNQ

For a diminishing reduction in accuracy

We're hiring and offer great internships ☺

Imperial College London | Norwegian University of Science and Technology | Politecnico di Milano | The University of Sydney