XILINX
ALL PROGRAMMABLE™

NTNU
Norwegian University of Science and Technology

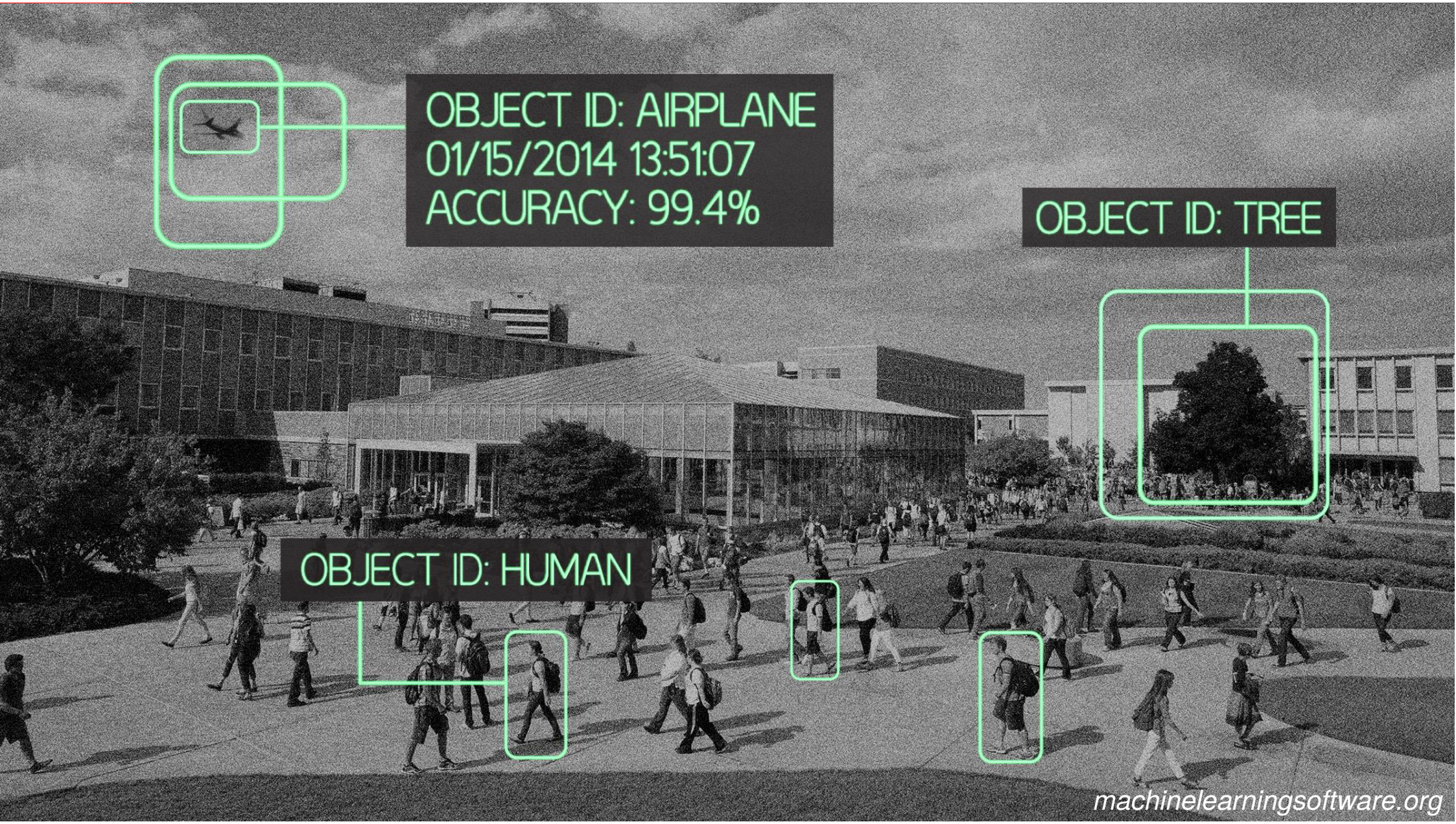
THE UNIVERSITY OF SYDNEY

FINN: A Framework for Fast, Scalable Binarized Neural Network Inference

Yaman Umuroglu (NTNU & Xilinx Research Labs Ireland)
in collaboration with N. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre and K. Vissers

- Introduction
- BNNs: Framework & Architecture
- BNNs: Experimental Evaluation
- Looking ahead: QNNs and Bit Serial
- Conclusions

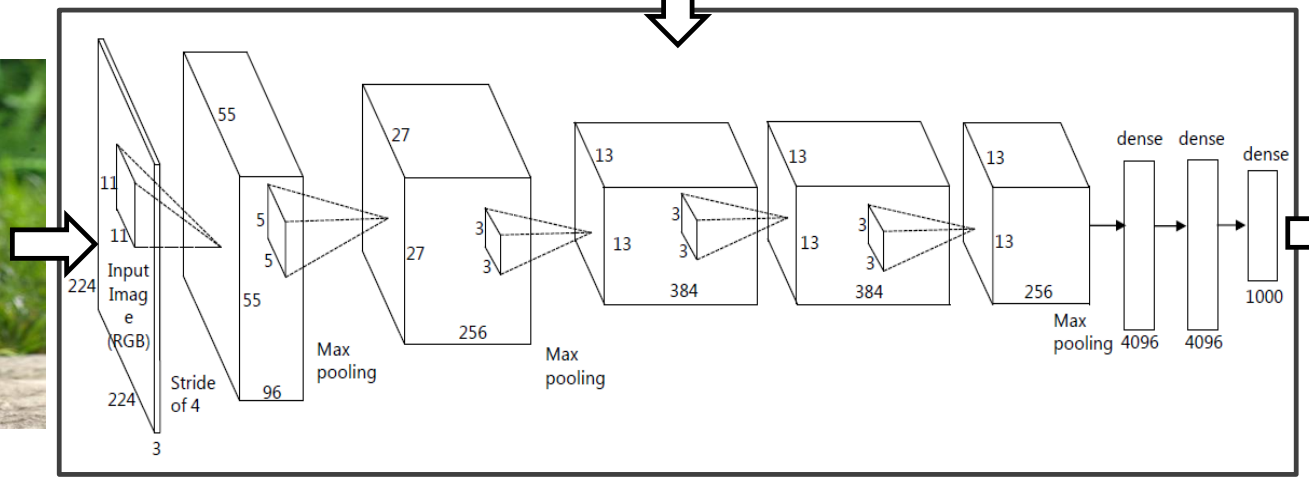
© Copyright 2016 Xilinx


OBJECT ID: AIRPLANE
01/15/2014 13:51:07
ACCURACY: 99.4%
OBJECT ID: TREE
OBJECT ID: HUMAN
machinelearningsoftware.org

Inference with Convolutional Neural Networks (CNNs)

off-chip, tens of megabytes of **floating point** weight data
(from training)

image to be classified



billions of **floating point** multiply-accumulate ops

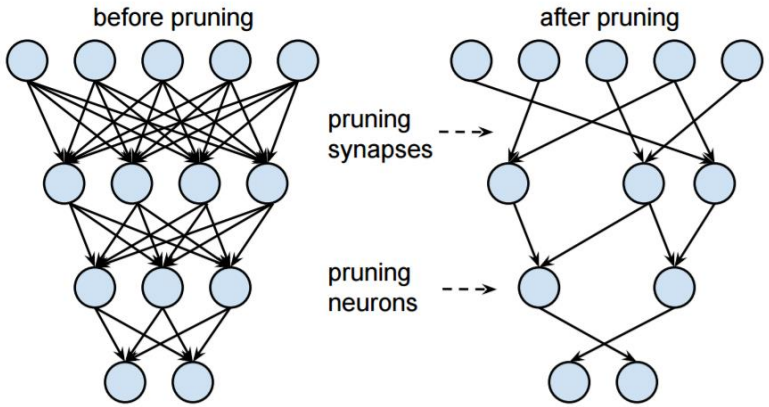
(up to several joules of energy)

«cat»

© Copyright 2016 Xilinx

XILINX ALL PROGRAMMABLE™

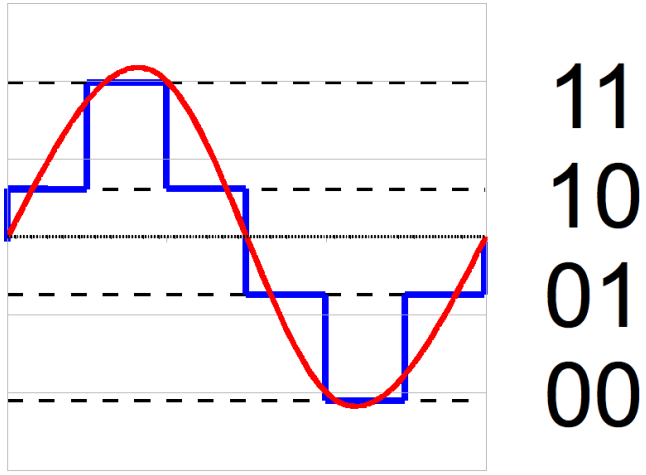
Some Emerging Alternatives for Energy-Efficient Inference



Synapse and neuron pruning

Sparse, irregular computation -- difficult to process efficiently

(Han et al., Learning both Weights and Connections for Efficient Neural Networks)



Quantization

Regular computation, smaller datapaths, fewer bits per weight

© Copyright 2016 Xilinx

Binarized Neural Networks (BNNs)

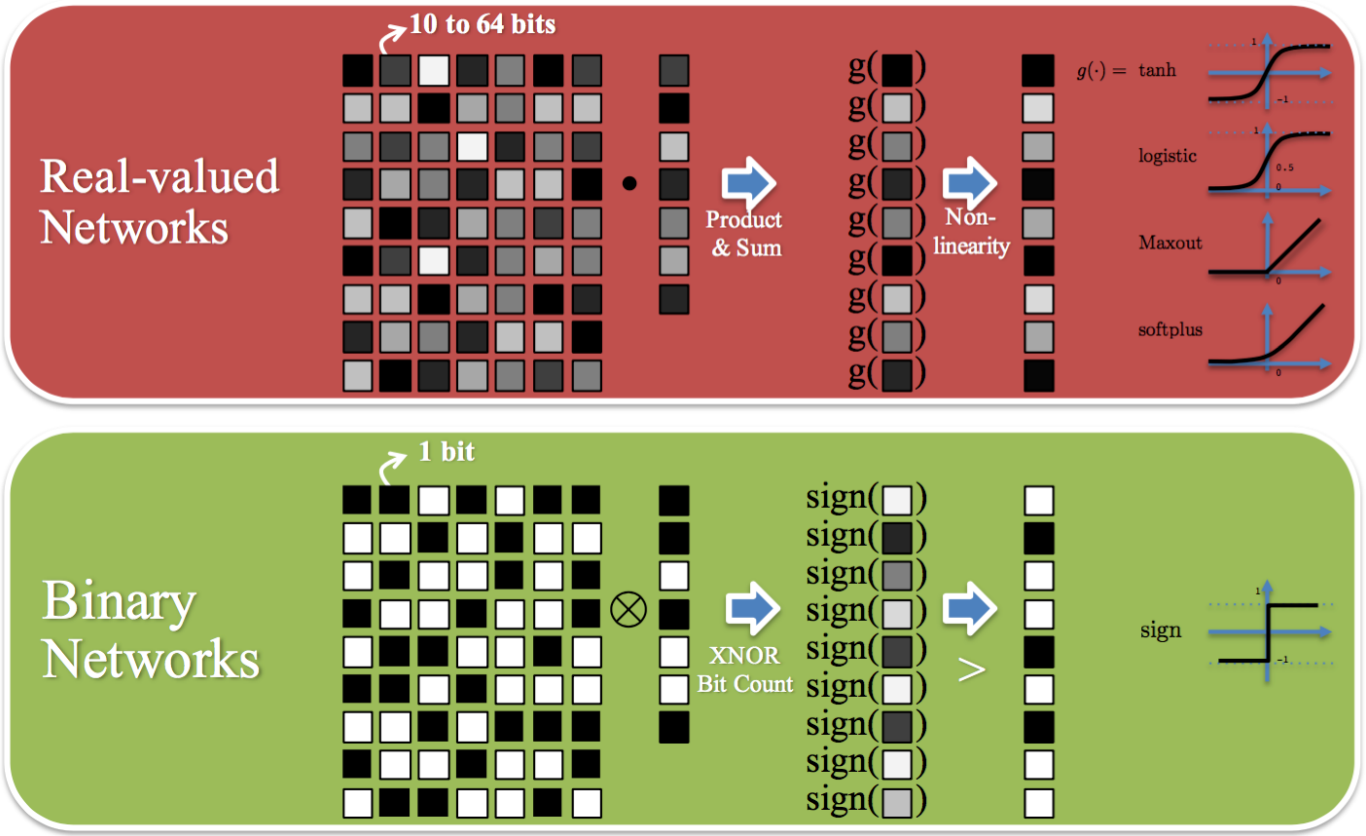
- **The extreme case of quantization**
  - Permit only two values: +1 and -1
  - Binary weights, binary activations
  - Trained from scratch, not truncated float
- **Courbariaux and Hubara et al. (NIPS 2016)**
  - Open source training flow
  - Standard “deep learning” layers
    - Convolutions, max pooling, batch norm, fully connected…
  - Competitive results on three smaller benchmarks
- **Key computation: binary matrix multiplication**

	MNIST	SVHN	CIFAR-10
Binary weights & activations	**0.96%**	**2.53%**	**10.15%**
FP weights & activations	**0.94%**	**1.69%**	**7.62%**
BNN accuracy loss	*-0.2%*	*-0.84%*	*-2.53%*

% classification error (lower is better)

© Copyright 2016 Xilinx

Binary Matrix Multiplication



Minje Kim, Bitwise Neural Networks. http://minjekim.com/demo_bnn.html

© Copyright 2016 Xilinx

Potential of BNNs on FPGAs

- ***Much* smaller datapaths**
  - Multiply becomes XNOR, addition becomes popcount
  - No DSPs needed, everything in LUTs
  - Lower cost per op = more ops every cycle, **trillions** of ops per second
- ***Much* smaller weights**
  - Large networks can fit entirely into on-chip memory (OCM)
  - More bandwidth, less energy compared to off-chip

Xilinx UltraScale+ MPSoC ZU19EG
(Vivado HLS, conservative estimates)

Precision	Peak GOPS	On-chip weights
1b	~66 000	~70 M
8b	~4 000	~10 M
16b	~1 000	~5 M
32b	~300	~2 M

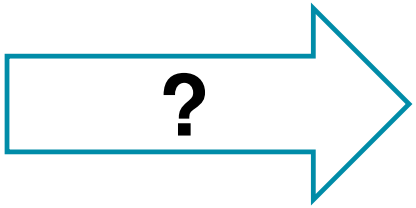
200x

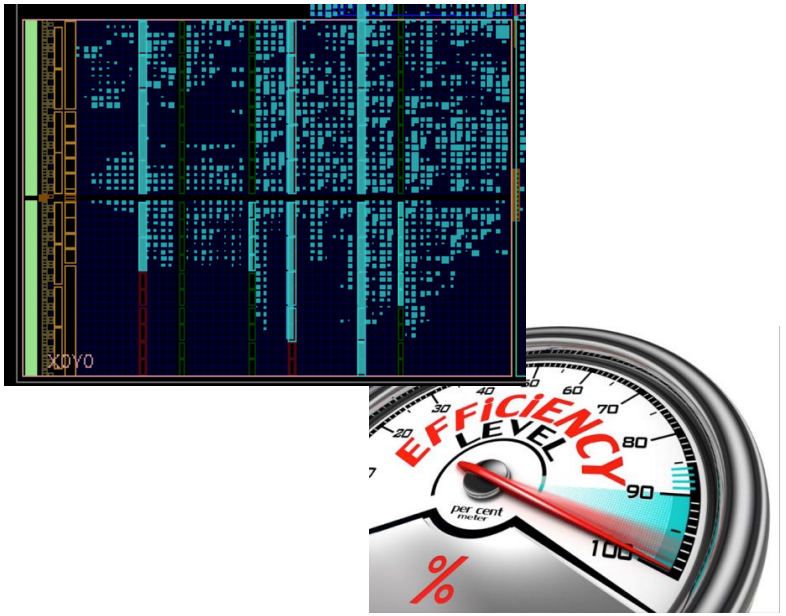
30x

= potential for **blazing fast** inference with **large BNNs** *on today's hardware*

© Copyright 2016 Xilinx

How do we exploit this potential?





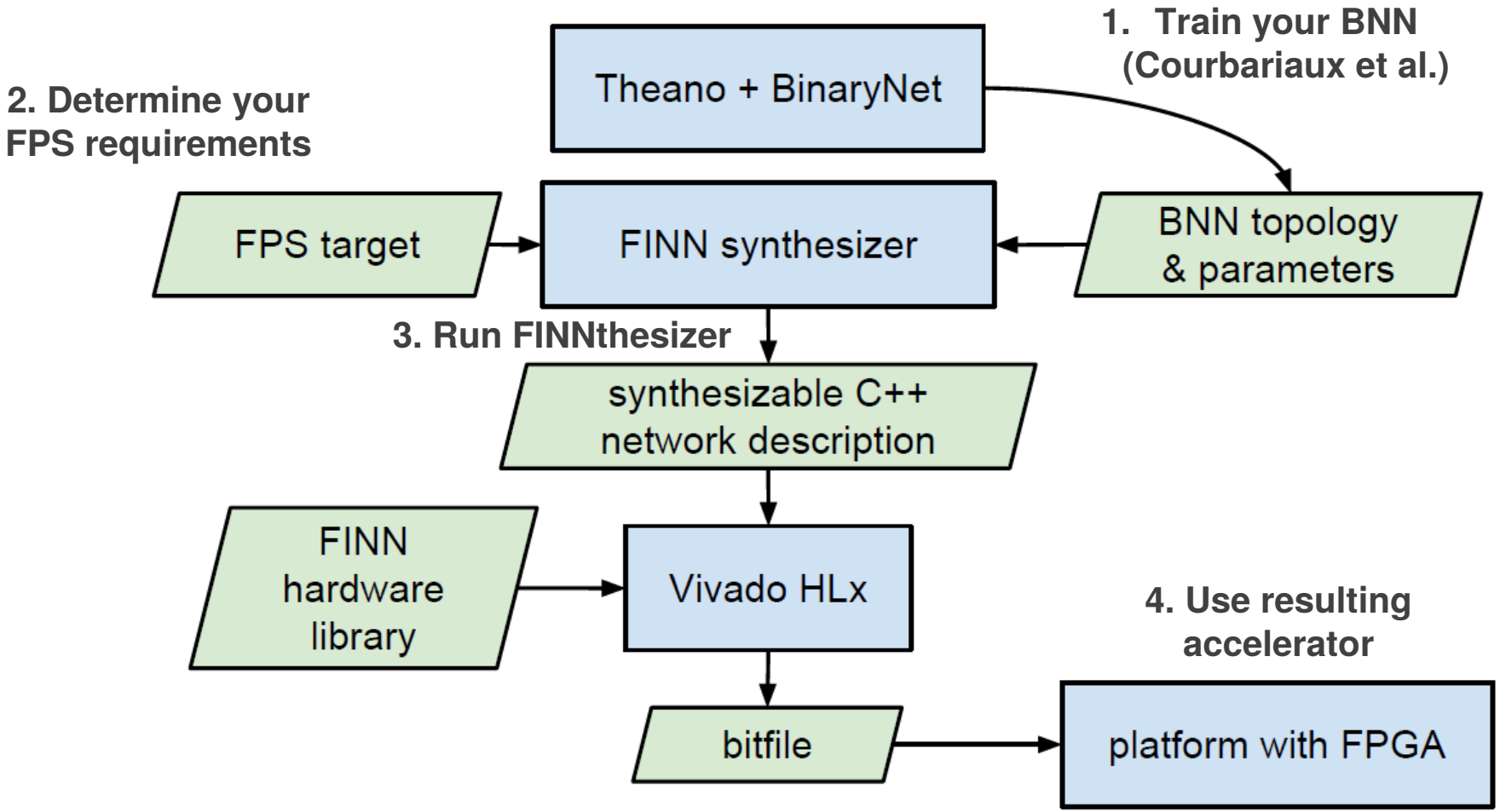
FINN, a Framework for Fast, Scalable Binarized Neural Network Inference

© Copyright 2016 Xilinx

- Introduction
- BNNs: Framework & Architecture
- BNNs: Experimental Evaluation
- Looking ahead: QNNs and Bit Serial
- Conclusions

© Copyright 2016 Xilinx

FINN at a glance



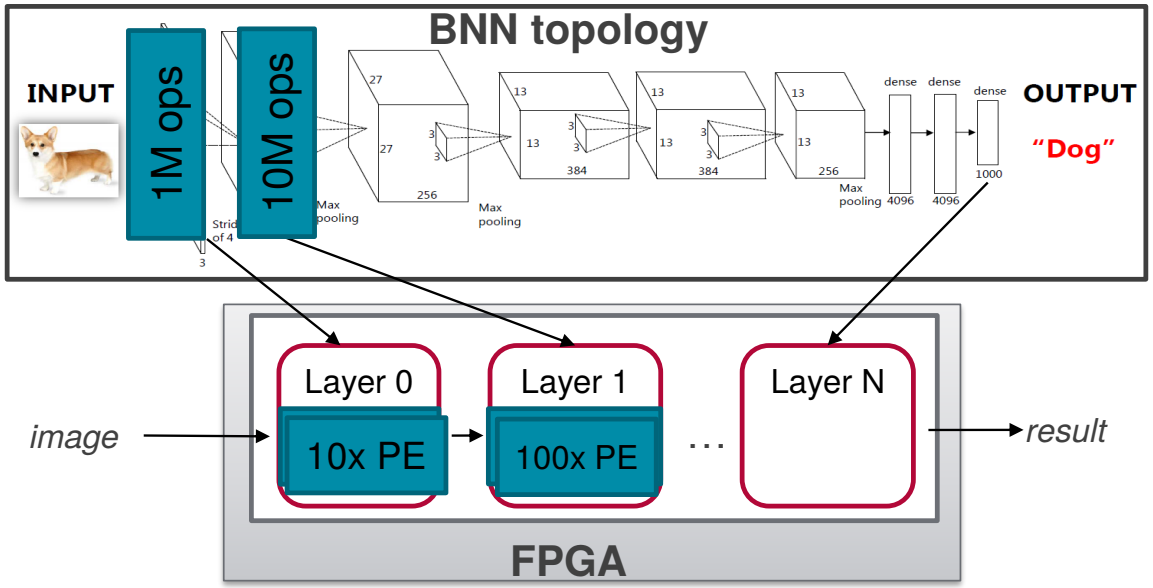
© Copyright 2016 Xilinx
XILINX ALL PROGRAMMABLE.

FINN Design Principles

- **One size does not fit all**
  - Generate tailored hardware for network and use-case
- **Stay on-chip**
  - Higher energy efficiency and bandwidth
- **Support portability and rapid exploration**
  - Vivado HLS (High-Level Synthesis)
- **Simplify with BNN-specific optimizations**
  - Exploit “compile time” optimizations to simplify the generated hardware
  - E.g. batchnorm and activation => thresholding. See details in the paper.

© Copyright 2016 Xilinx

Heterogeneous Streaming Architecture



1x FPS

10x FPS

- **One hardware layer per BNN layer, parameters built into bitstream**
  - Both inter- and intra-layer parallelism
- **Heterogeneous: Avoid "one-size-fits-all" penalties**
  - Allocate compute resources according to FPS and network requirements
- **Streaming: Maximize throughput, minimize latency**
  - Overlapping computation and communication, batch size = 1

© Copyright 2016 Xilinx

The Matrix-Vector Threshold Unit (MVTU)

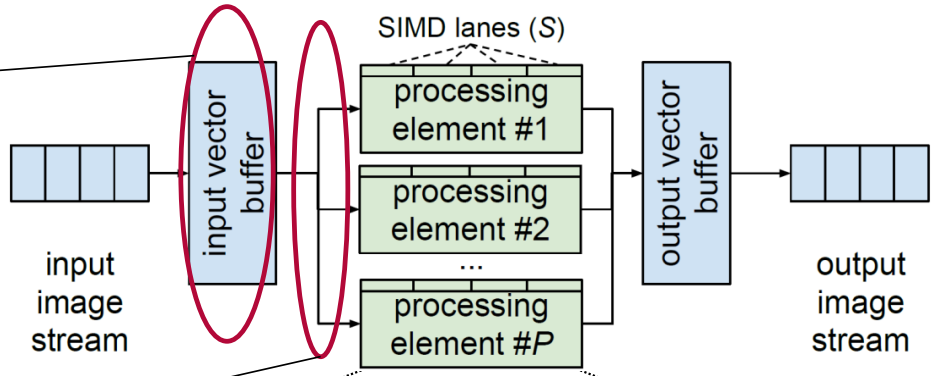
- **Core computational element of FINN, tiled matrix-vector multiply**
- **Computes a (P) row x (S) column chunk of matrix every cycle, per-layer configurable tile size**

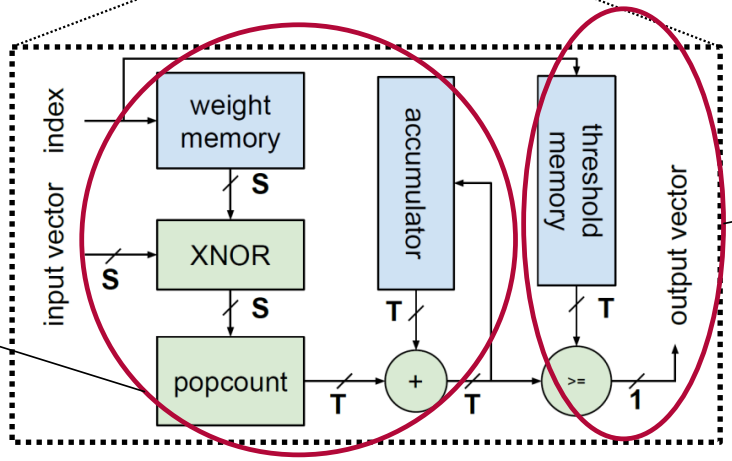
buffer input for reuse (neuron folding)

broadcast S bits of input every cycle

multiply-accumulate, but with XNOR and popcount instead

apply thresholding and send to next layer





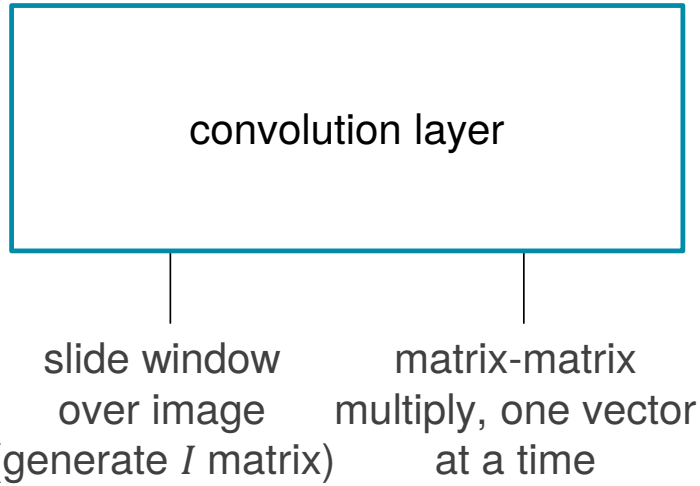
© Copyright 2016 Xilinx

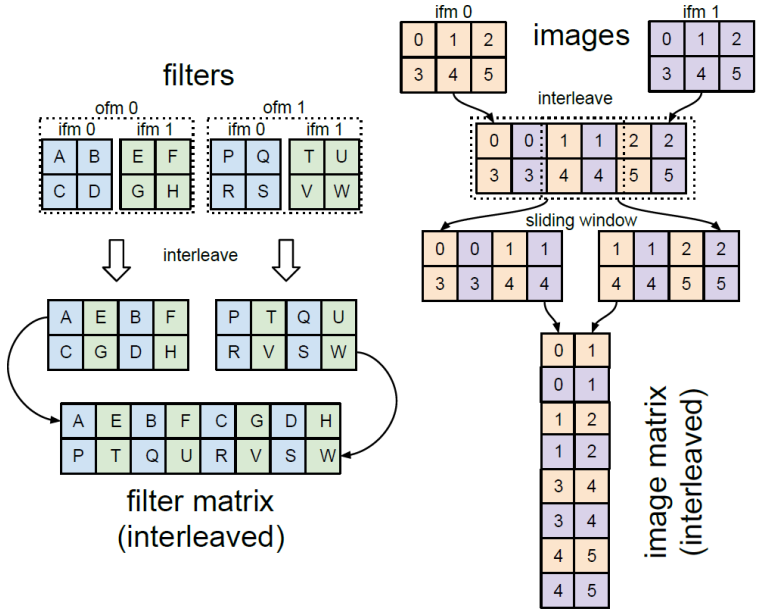
XILINX ALL PROGRAMMABLE

Convolutional Layers

- **Lower convolutions to matrix-matrix multiplication, $W \cdot I$**
  - W : filter matrix (generated offline)
  - I: image matrix (generated on-the-fly)

- **Two components:**





© Copyright 2016 Xilinx

- Introduction
- BNNs: Framework & Architecture
- BNNs: Experimental Evaluation
- Looking ahead: QNNs and Bit Serial
- Conclusions

© Copyright 2016 Xilinx

Scenario: 9k FPS target

12 kFPS with ~1-3 W over idle power

Prototype	FPS	GOPS	BRAM	LUT	Latency [us]	Power [W]
SFC-fix	12.2 k	8	16	**5 155 (3%)**	240	**8.1**
LFC-fix	12.2 k	71	114.5	**5 636 (3%)**	282	7.9
CNV-fix	11.6 k	1 306	152.5	29 274 (13%)	550	**10**

FPS goal exceeded (integer folding factors)

Scalability to small footprints

© Copyright 2016 Xilinx

XILINX ALL PROGRAMMABLE™

Scenario: Maximum Throughput

Slow down clock to 250 kHz: 0.2W of power, 9k FPS

Prototype	FPS	GOPS	BRAM	LUT	Latency [us]	Power [W]
SFC-max	**12.3 M**	8 265	4.5	91 131 (42%)	**0.31**	21.2
LFC-max	**1.5 M**	9 085	396	82 988 (38%)	**2.44**	22.6
CNV-max	**21.9 k**	2 465	186	46 253 (21%)	**283**	11.7

Unprecedented classification rates

Ultra-low latency For robotics, AR, UAVs

© Copyright 2016 Xilinx

Comparison to Prior Work

- **How to compare neural network accelerators across precisions and devices?**
  - Accuracy, images per second, energy efficiency

		Accuracy	FPS	Power (chip)	Power (wall)	kFPS / Watt (chip)	kFPS / Watt (wall)	Precision
FINN	MNIST, SFC-max	95.8%	12.3 M	7.3 W	21.2 W	1693	583	1
	MNIST, LFC-max	98.4%	1.5 M	8.8 W	22.6 W	177	269	1
	CIFAR-10, CNV-max	80.1%	21.9 k	3.6 W	11.7 W	6	2	1
	SVHN, CNV-max	94.9%	21.9 k	3.6 W	11.7 W	6	2	1
Prior Work	MNIST, Alemdar et al.	97.8%	255.1 k	0.3 W	-	806	-	2
	CIFAR-10, TrueNorth	83.4%	1.2 k	0.2 W	-	6	-	1
	SVHN, TrueNorth	96.7%	2.5 k	0.3 W	-	10	-	1

Max accuracy loss: ~3%

10 – 100x better performance

CIFAR-10/SVHN energy efficiency comparable to TrueNorth ASIC

© Copyright 2016 Xilinx

- Introduction
- BNNs: Framework & Architecture
- BNNs: Experimental Evaluation
- Looking Ahead: QNNs and Bit Serial
- Conclusions

© Copyright 2016 Xilinx

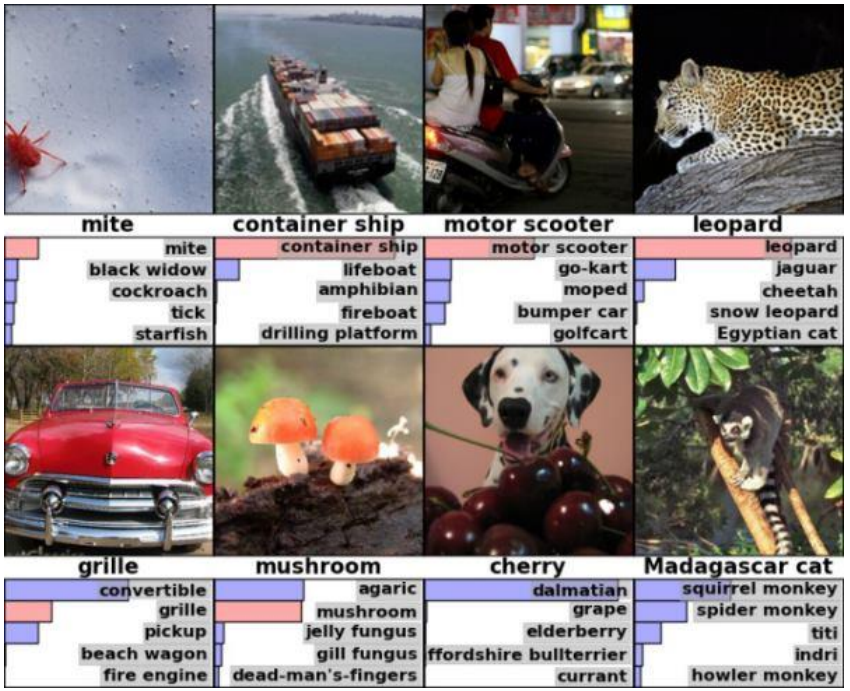
Quantized Neural Networks (QNNs)

- **BNNs work well for *smaller* problems, but...**
  - Significant (~15%) accuracy loss for e.g. ImageNet-1K

- **Solution: a few more bits of precision**
  - *W*-bit weights, *A*-bit activations
  - Still only integer ops (no floating point)
  - Top-5 accuracy loss down to ~**5%** with W1A2

Network	Float	W1A2 HWGQ	Accuracy Loss
AlexNet	81.5%	76.3%	-5.2%
GoogLeNet	90.5%	84.9%	-5.6%
VGG-like	89.3%	85.6%	-3.7%

Cai et al. «Deep Learning With Low Precision by Half-wave Gaussian Quantization». CVPR'17



© Copyright 2016 Xilinx

Few-bit Integer Matrix Multiplication

- **Key operation in QNNs: multiply *few-bit* integer matrices**
  - Number of bits could be different in each layer
  - Also the key operation for reduced-precision training!

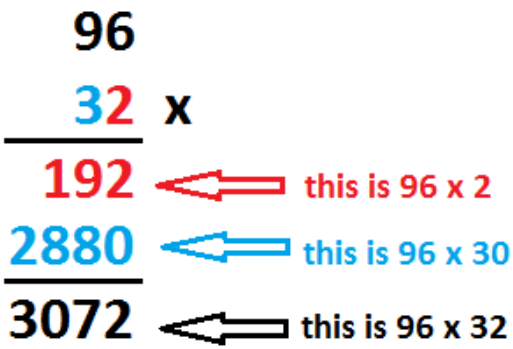
- **Could cast everything to lowest supported precision (i.e. 8-bit), but...**
  - Loses arithmetic efficiency (upper bits are always zeroes)
  - Loses memory footprint advantages (e.g. 4x larger for 2-bit operands)

- **How do we formulate *efficient* few-bit integer matrix multiplication on a *fixed* datapath?**
  - Fixed datapath = CPU, GPU, ASIC BNN accelerator, FPGA overlay...

© Copyright 2016 Xilinx

QNN as Bit-Serial BNN

- ***w-bit* times *a*-bit matrix multiplication = sum of weighted binary matrix multiplications**
  - «Primary school» way of doing multiplications – just in binary
- **Performance is approximately <binary matrix mul performance> / (*w* * *a*)**



function BITSERIALGEMM(W, A, res)
  for $i \leftarrow 0 \ldots w-1$ **do**
    for $j \leftarrow 0 \ldots a-1$ **do**
      sgnW $\leftarrow (i == w-1\ ?\ -1 : 1)$
      sgnA $\leftarrow (j == a-1\ ?\ -1 : 1)$
      BINARYGEMM($W[i]$, $A[j]$, res, sgnW $\cdot$ sgnA $\cdot\ 2^{i+j}$)
    end for
  end for
end function

© Copyright 2016 Xilinx

QNN/BNN operations on commodity CPUs

- **Binary GEMM: multiply matrices of {-1, +1} values**
  - Bitwise XNOR followed by popcount
  - ARM NEON, x86 SSE 4.1, NVIDIA and AMD GPUs have popcount instructions

- **Peak performance assuming 1 AND + 1 popcount every cycle:**
  - 1.8 GHz ARM Cortex-A57, 128-bit NEON: **460 binary GOPS per core (versus 22 int8 GOPS)**
  - 3.4 GHz Intel i7-3770K, 64-bit scalars: **435 binary GOPS per core (versus 50 int8 GOPS)**

- **If w=1 and a=2, up to 10x faster than using int8 operations for ARM**
  - Easier to achieve higher performance due to higher arithetic intensity
  - Fewer cache misses = fewer off-chip accesses = less energy

© Copyright 2016 Xilinx

- Introduction
- BNNs: Framework & Architecture
- BNNs: Experimental Evaluation
- Looking ahead: QNNs and Bit Serial
- Conclusions

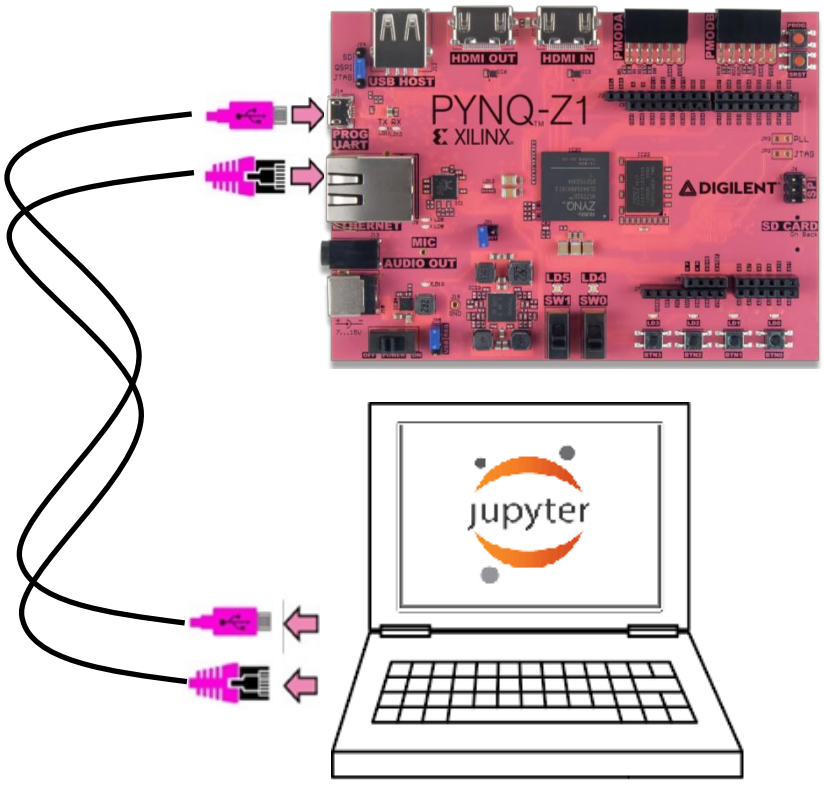
© Copyright 2016 Xilinx

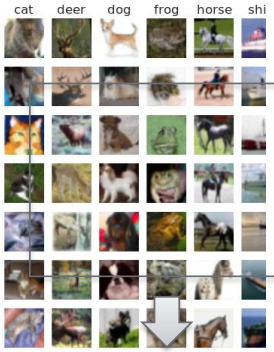
Conclusions and *Future Work*

- FPGAs can do **trillions of binary operations per second.**
  - *How high can we go?*
- FINN can build BNN inference FPGA accelerators that classify **10Ks to Ms of images per second**, at < 25 W, on **today's hardware.**
  - *What if the neural network doesn't fit on-chip?*
- Bit-serial is a promising way to implement **general few-bit integer matrix multiplication** for both FPGAs and CPUs.
  - *Deep neural nets on microcontrollers / IoT nodes?*
  - *Low-precision on-device training*
  - *Approximate computing by only considering higher-order bits*

© Copyright 2016 Xilinx

Open Source BNNs on Xilinx's Python Productivity Kit PYNQ





Trained datasets: CIFAR10, traffic signs, SVHN

python

Image preprocessing in Python

Binary Neural Network in FPGA fabric & on ARM processor

"cat"

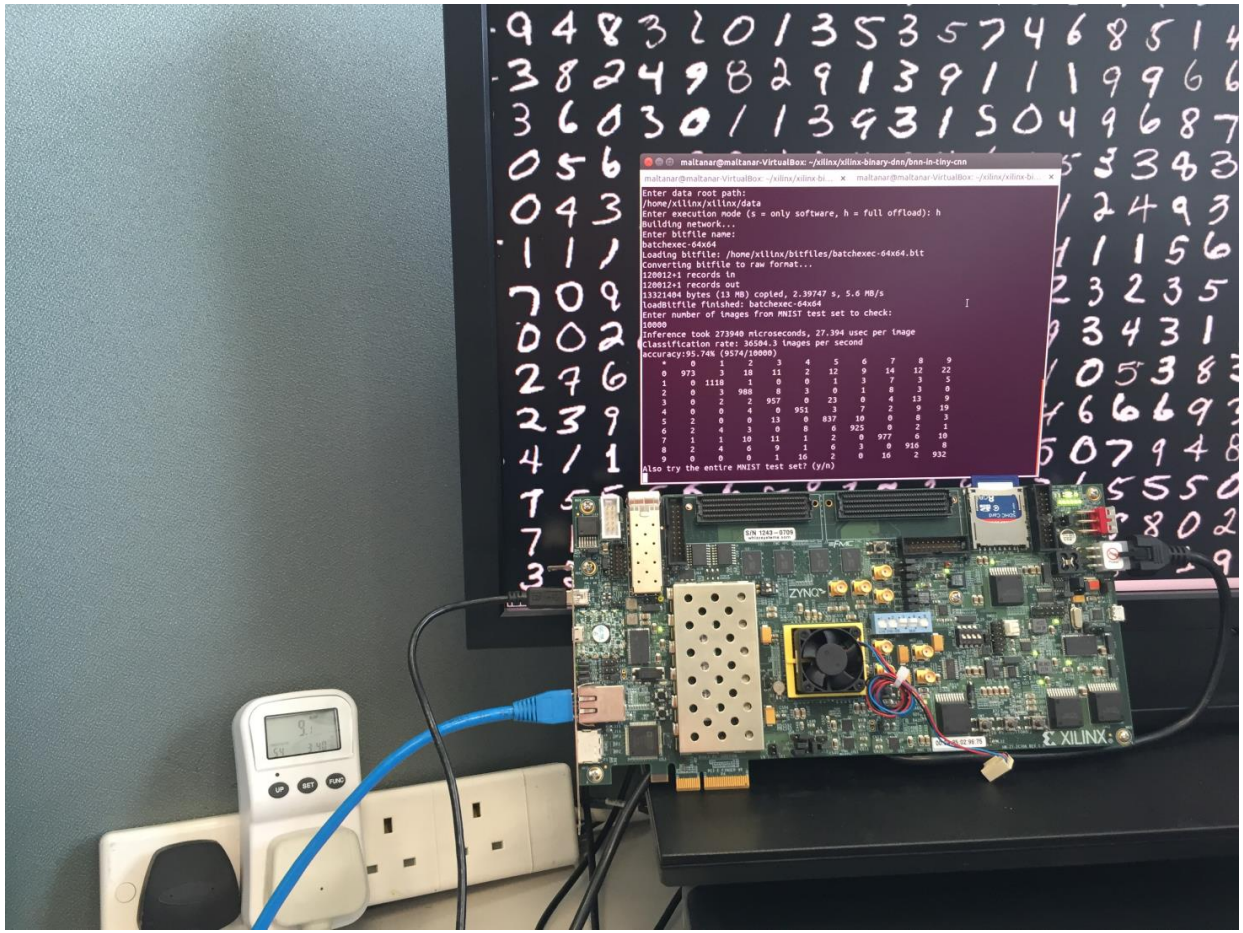
https://github.com/Xilinx/BNN-PYNQ

© Copyright 2016 Xilinx

Thank You!

© Copyright 2016 Xilinx

Experimental Setup



ZC706 development platform:
Z7045 All-Programmable SoC
2 ARM Cortex-A9 cores
218k LUTs, 545 BRAMs

- **10000 test images in PS DDR**
  - Streamed in-out via DMA
- **FINN-generated accelerator on PL**
  - Running at 200 MHz
- **ARM core:**
  - launches accelerator
  - measures time
  - verifies results
- **PMBus and wall power monitoring**
  - Idle wall power ~7 W

© Copyright 2016 Xilinx

Neurons versus Accuracy – Float and Binarized

Neurons/layer	Binary Err. (%)	Float Err. (%)	# Params	Ops/frame
128	6.58	2.70	134,794	268,800
256	4.17	1.78	335,114	668,672
512	2.31	1.25	932,362	1,861,632
1024	1.60	1.13	2,913,290	5,820,416
2048	1.32	0.97	10,020,874	20,029,440
4096	1.17	0.91	36,818,954	73,613,312

~2x binary neurons give approximately the same accuracy (for MNIST)

© Copyright 2016 Xilinx
XILINX ALL PROGRAMMABLE™

Test Networks & Scenarios

- **BNN Topology:**
  - **SFC:** small fully-connected, 0.6 MOP per image
  - **LFC:** large fully-connected, 5.8 MOP per image
  - **CNV:** convolutional, 112.5 MOP per image

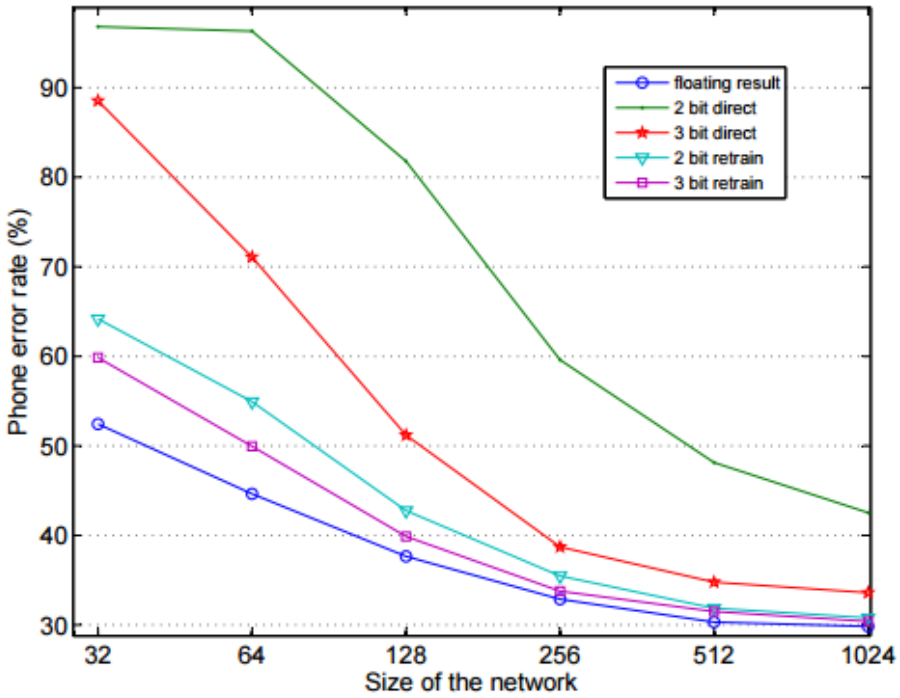
  - SFC & LFC on MNIST, binarized inputs and outputs
  - CNV on CIFAR-10 and SVHN, 8-bit inputs, 16-bit outputs

- **Scenario:**
  - **fix:** assume I/O bound, achieve 9000 FPS
    - **max:** achieve as high FPS as possible

© Copyright 2016 Xilinx

Redundancy and Quantization

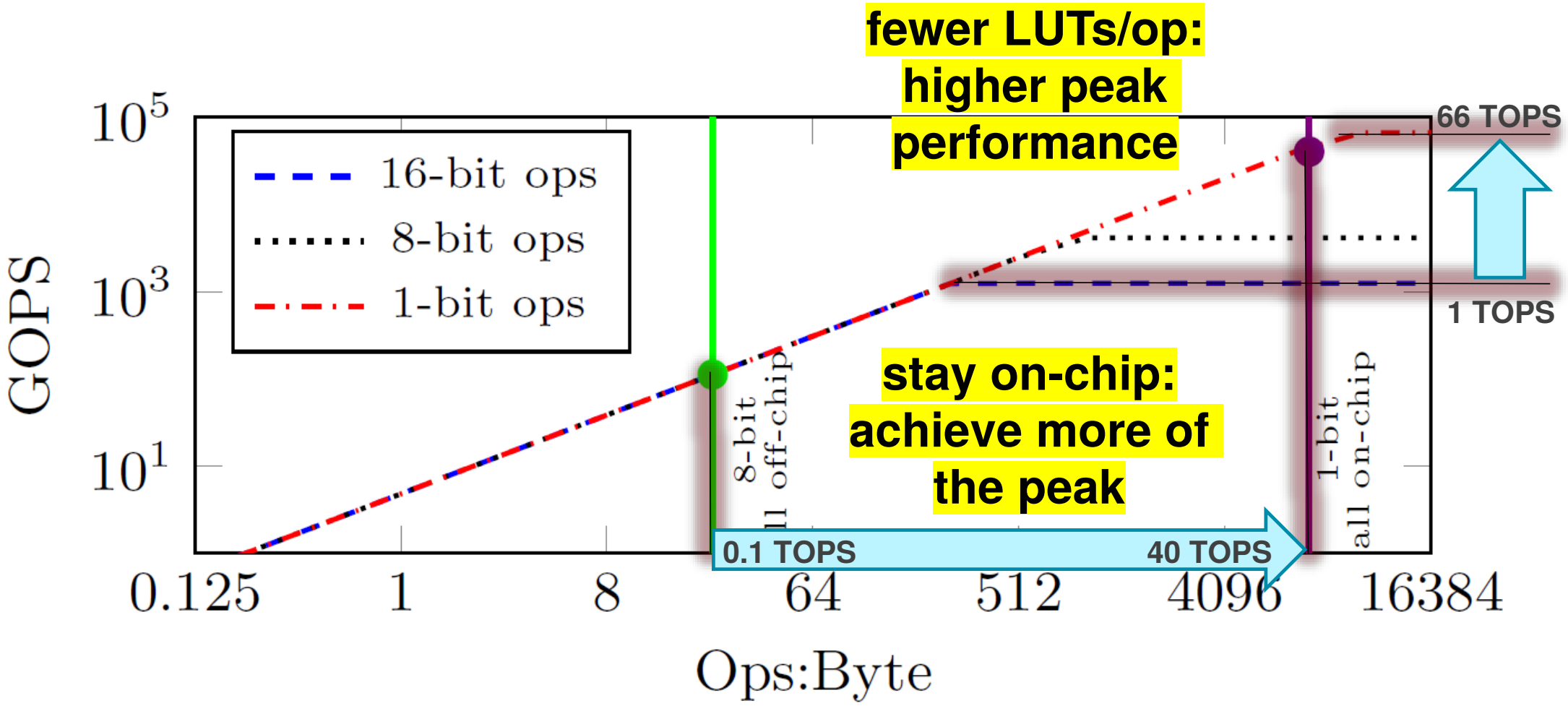
- **Evidence of redundancy in trained networks**
  - sparsification, low-rank approximations, fault tolerance…
- **Reduced precision (quantization)**
  - Restrict weights and/or activations to *Q*-bit values
  - HW benefits: Low-bitwidth datapaths, regular compute
- **Sung et al: Quantization works well when…**
  - …the network is "big enough"
  - …the network is aware of quantization during (re)training



"(…) the performance gap between the floating-point and the retrain-based ternary (+1, 0, -1) weight neural networks (…) almost vanishes in fully complex networks (…)"
(Sung et al, Resiliency of Deep NNs Under Quantization)

© Copyright 2016 Xilinx

Potential of BNNs on FPGAs



© Copyright 2016 Xilinx

FINN Synthesizer («FINNthesizer»)

- **Inputs:**
  - BNN topology (JSON) and trained parameters (NPZ)
  - Desired frames per second (FPS)
- **Apply BNN-specific compute transformations**
  - Simplifications enabled by the value-constrained nature of BNNs
  - Popcount, batchnorm-activation as threshold, maxpool as OR (details in paper)
- **Compute «folding factors» to meet FPS goal**
- **Output:**
  - C++ (Vivado HLS) description of desired architecture

© Copyright 2016 Xilinx

Top Level

```
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);
}
```

Stream definitions

Move image in from PS memory

Layer instantiation connected by streams

Move results to PS memory

© Copyright 2016 Xilinx

XILINX ALL PROGRAMMABLE.

MVTU

```
for (unsigned int nm = 0; nm < neuronFold; nm++) {
    for (unsigned int sf = 0; sf < synapseFold; sf++) {
#pragma HLS PIPELINE II=1
          ap_uint<SIMDWidth> inElem;
          if (nm == 0) {
             inElem = in.read();
             inputBuf[sf] = inElem;
          } else {
             inElem = inputBuf[sf];
          }
          for (unsigned int pe = 0; pe < PECount; pe++) {
#pragma HLS UNROLL
             ap_uint<SIMDWidth> weight = weightMem[pe][nm * synapseFold + sf];
             ap_uint<SIMDWidth> masked = ~(weight ^ inElem);
             accPopCount[pe] += NaivePopCount<SIMDWidth, PopCountWidth>(masked);
          }
    }
    ap_uint<PECount> outElem = 0;
    for (unsigned int pe = 0; pe < PECount; pe++) {
#pragma HLS UNROLL
          outElem(pe, pe) = accPopCount[pe] > thresMem[pe][nm] ? 1 : 0;
          accPopCount[pe] = 0;          // clear the accumulator
    }
```

Folding

Reading
Inputs or consume
internal (when folded)

Indexing weight and
threshold memory
binary MAC

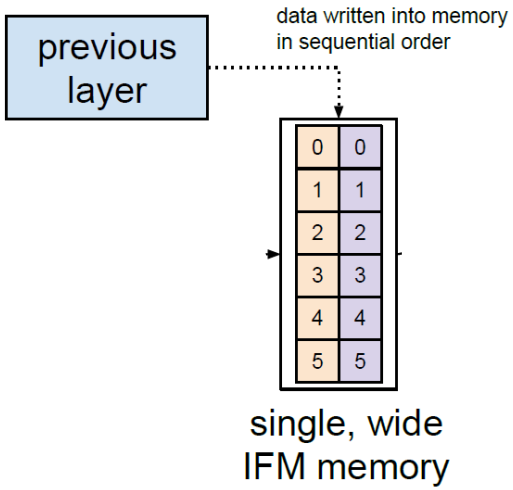
Batchnorm & activation
as threshold

© Copyright 2016 Xilinx

XILINX ALL PROGRAMMABLE.

Convolution: Sliding Window Unit (SWU)

- **Buffer incoming images in a single, #IFM-wide memory**
- **Read out addresses corresponding to sliding window location**
- **Preserve produce-consume order to minimize buffering**

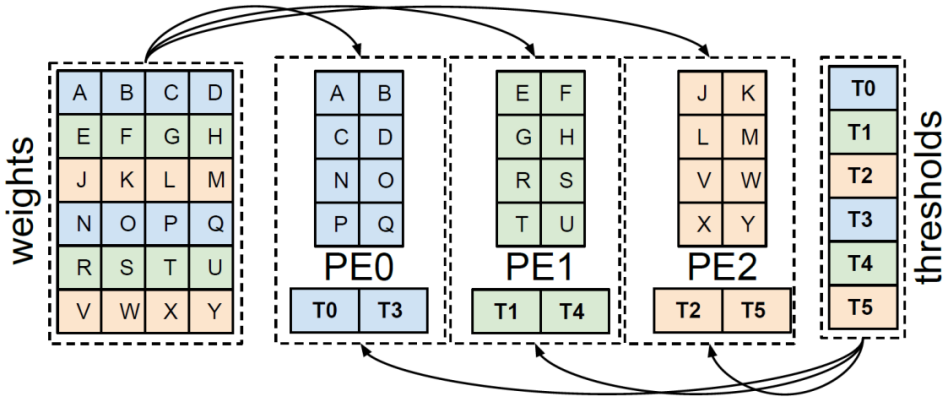


© Copyright 2016 Xilinx
XILINX ALL PROGRAMMABLE

Folding

- **Time-multiplex (or *fold*) real neurons onto hardware neurons**
  - Control folding via number of PEs and SIMD lanes in each layer

- **Folding computed by FINNthesizer to satisfy FPS requirements**
  - FPS for one layer = clock frequency / folding factor
  - FPS of streaming system = minimum FPS of any layer
  - FINNthesizer will balance folding factors to match FPS across layers

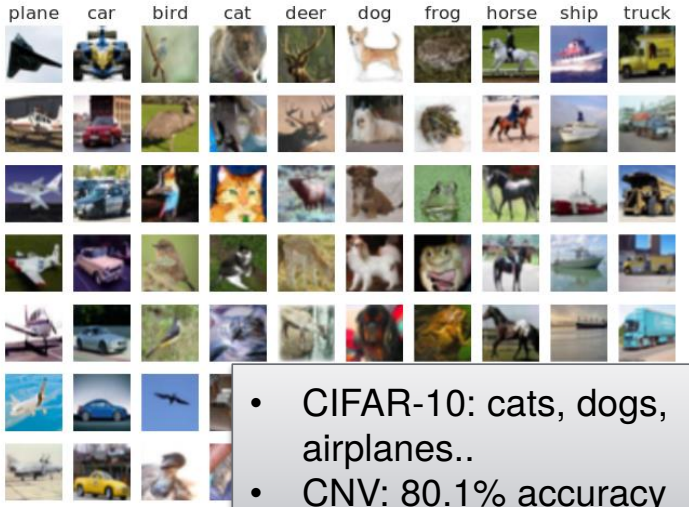


© Copyright 2016 Xilinx
XILINX ALL PROGRAMMABLE

Input Data

- MNIST handwritten digits
- LFC: 98.4% accuracy

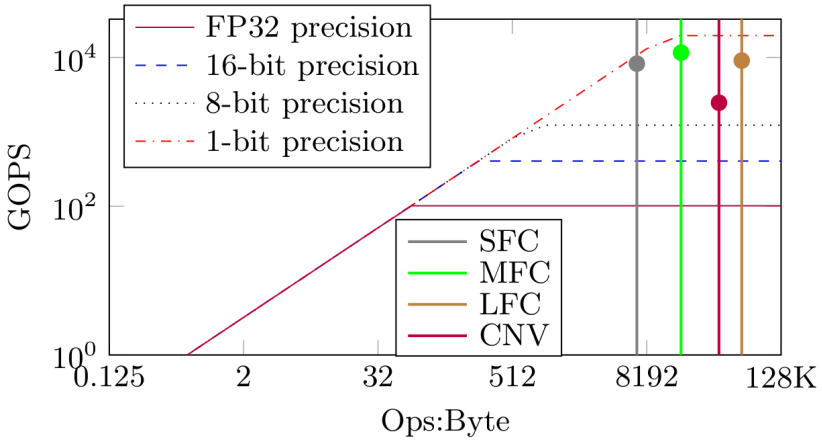
- Street View House Numbers
- CNV: 94.9% accuracy

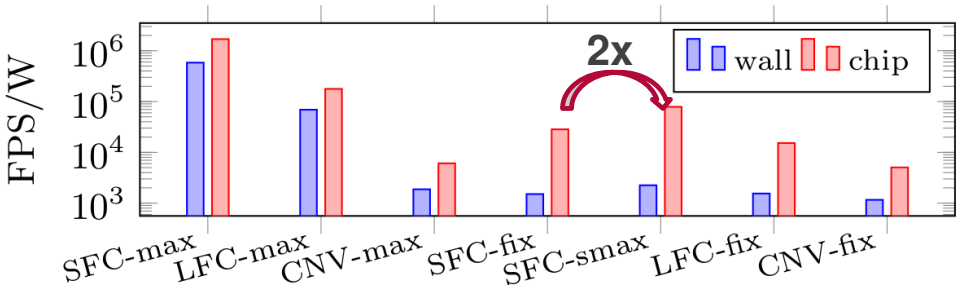


- CIFAR-10: cats, dogs, airplanes..
- CNV: 80.1% accuracy

© Copyright 2016 Xilinx

Results – Other Highlights





- **Up to 58% of roofline performance estimate**
  - SFC-max: DRAM bandwidth-bound
  - LFC-max: resource bound (BRAM)
  - CNV-max: architecture bound (SWU)

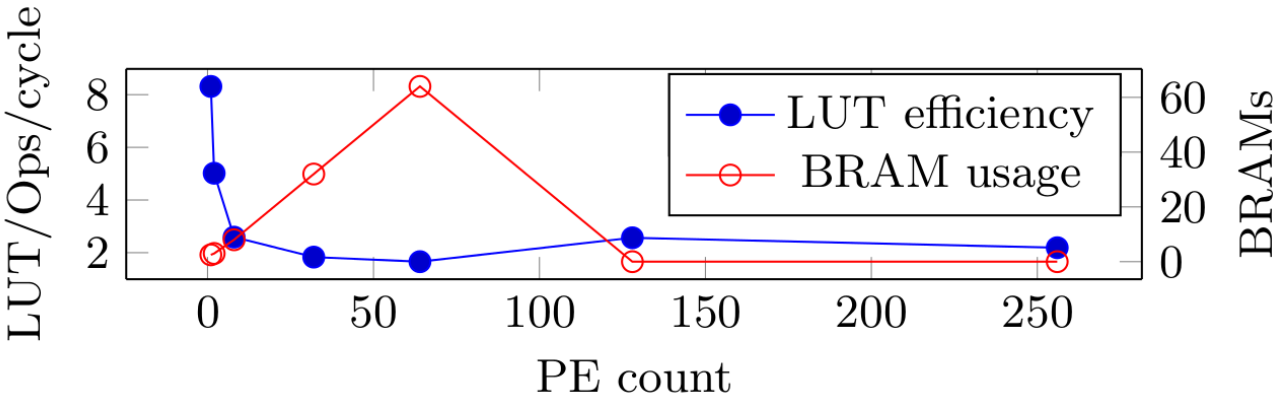
- **Massive but slow-clock parallelism: good energy efficiency**
  - Use 250 kHz clock for 12M FPS prototype: 15 kFPS on MNIST with 0.2 W chip power
  - Observed that slowed-down SFC-max 2x more energy efficient than SFC-fix

© Copyright 2016 Xilinx

Results – Efficiency

- **Runtime utilization: Operators busy 70-90% of the time**

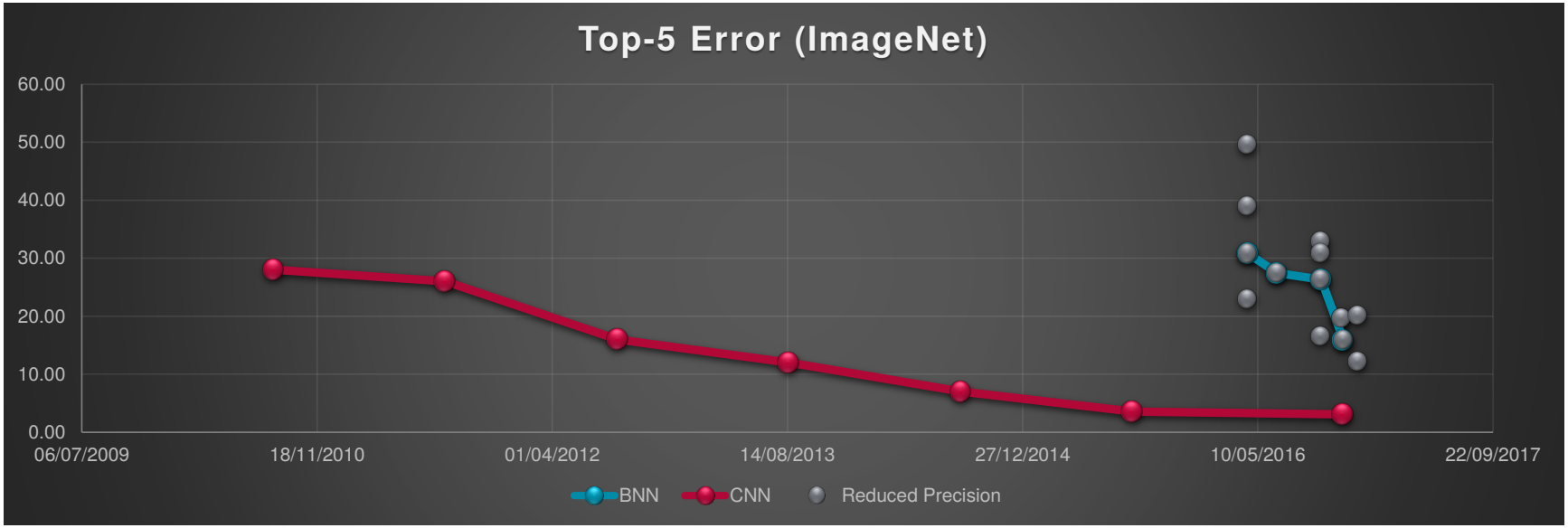
- **LUT (instead of BRAM) storage if many PEs**
  - Fixed amount of work divided between more workers
  - Complex mapping problem, multi-dimensional tradeoff between performance/area



© Copyright 2016 Xilinx
XILINX ALL PROGRAMMABLE

Accuracy of BNNs on ImageNet

Published Results for FP CNNs, BNNs and Extreme Reduced Precision NNs



- BNNs are new and accuracy results are improving rapidly

© Copyright 2016 Xilinx

XILINX ALL PROGRAMMABLE™