XILINX
ALL PROGRAMMABLE™

Binary Networks on FPGAs

Michaela Blott, Kees Vissers, Giulio Gambardella (Xilinx Research)
Yaman Umuroglu (NTNU), Nick Fraser (Sydney Uni.), Gianluca Durelli (Politecnico Milano)

THE UNIVERSITY OF SYDNEY
NTNU Norwegian University of Science and Technology
POLITECNICO DI MILANO

Agenda

- Introductions
- Framework & Architecture
- Experimental Results
- PYNQ Overlay
- Discussions

- Introductions
  - Xilinx Research
  - Challenges with neural networks on FPGAs
  - Reduced precision neural networks
- Framework & Architecture
- Experimental Results
- PYNQ Overlay
- Discussions

Xilinx Research - Ireland

- 8 researchers + students & visiting scholars
- 2 university program
- Est. 10 years ago

Applications & Architectures:
Through application-driven technology development with customers, partners, and engineering & marketing

ICHEC Irish Centre for High-End Computing
Technische Universität Kaiserslautern
Politecnico di Milano
IBM
The University of Sydney
NTNU Norwegian University of Science and Technology
ETH
Bai du 百度
...
hp

XILINX ALL PROGRAMMABLE.

Convolutional Neural Networks

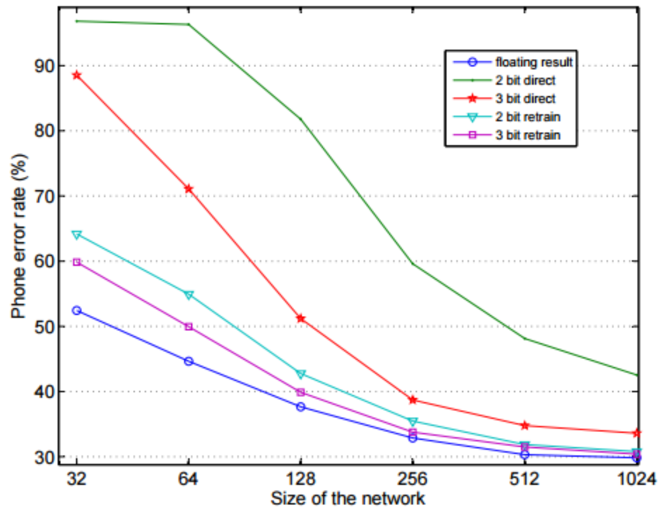
- **CNNs are the predominant machine learning algorithm**
  - Achieving superhuman accuracy since 2015
  - Use cases span image recognition, language processing, speech recognition, time series prediction, recommender systems, medical diagnosis, autonomous vehicles and many more

- **CNNs are very high in compute and memory requirements**
  - Increasing operational intensity

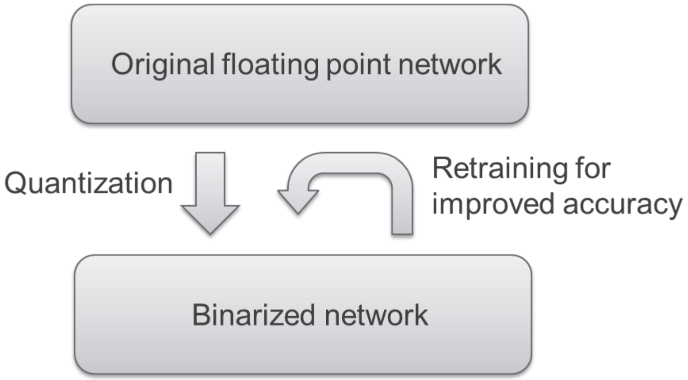
CNN for ImageNet datasets	Memory (SP) [MB]	Operations [GOPS]	Operational Intensity [OPS:B]
AlexNet – complete	244	1.5	5.97
VGG-16	552	31	55.84
GoogleNet	27.2	3.1	55.24

Increasingly Reduced Precision Networks

- **Floating point (FP) CNNs contain a lot of redundancy**
  - Even Nvidia is moving from FP, HP to 8b fixed point integer
- **Reducing precision is shown to work to 6b without loss of accuracy – Dec. 2015**
  - 50x and more reduction in model size (no external memory needed)
- **Bill Dally (Stanford), EMDNN 2016:**
  - showed TTN on par with FP for AlexNet top-1 and top-5, ResNet20,32,44,56
- **Reducing to the extreme: binary and almost binary neural networks (BNNs) – Jan 2016**
  - Possible with retraining
  - No accuracy loss for small networks
  - Small drop for large networks



(a)



XILINX ALL PROGRAMMABLE.

Potential of Binary Networks on FPGAs

<u>Roofline Assumptions</u>:
Application can fill device to 70% (fully parallelizable)
FPGA cost function:
- 1b int: 2.5LUTs (with HLS)
- 4b int: 11LUTs
- 8b int: 40LUTs
- 32b float: 178LUTs, 2DSP
- 250MHZ for KU115

KU115:
- 663k LUTs
- 5520 DSPs
- 2160 BRAM

- **Multiply accumulate becomes XNOR with bit counts**

Cost of operations	LUTs	DSPs
1b	2.5	0
4b	11	0
8b	40	0
32b	178	2

- **Today's FPGAs have a much higher peak performance for binary operations**
  - Example: KU115 offers lots of LUTs but limited DSPs for HP & SP: 5'520 DSPs and 663'360 LUTs

Peak performance	TOps/s
1b	46
4b	11
8b	3
32b	0.

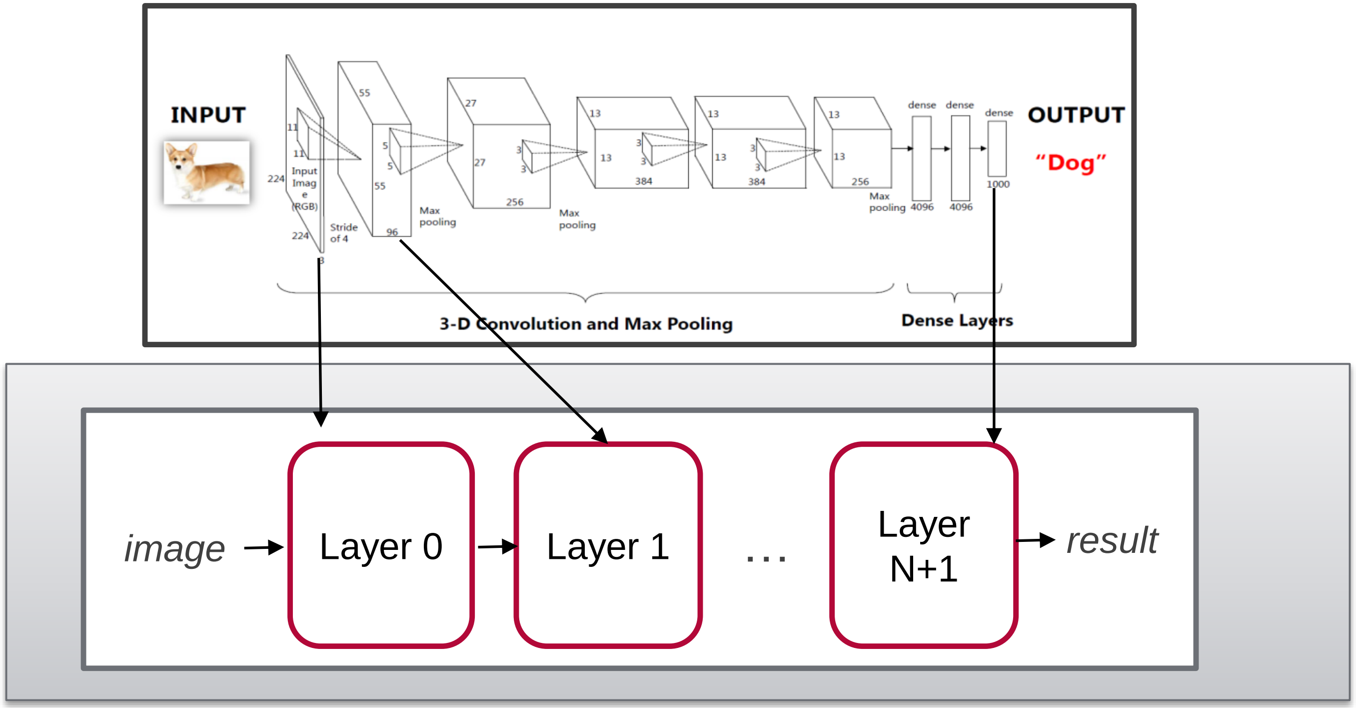
10x

Huge performance potential for low bit precision – today
No external memory needed

Nvidia today:
4.5 TOps/s
measured

- **Model sizes small enoug**

Potential of Dataflow Architectures on FPGAs



- **Binary networks can be implemented as feed-forward data flow architectures**
  - If we had enough resources to implement a full network fully parallelized
  - ⇒ classifying 1 image @ clock rate (for example 250MHz => 250Mfps)
- **Large networks need to be folded over the input stream**
  - Conceptually we have 5 orders of magnitude to play with
- **Lowest storage requirements and lowest latency**

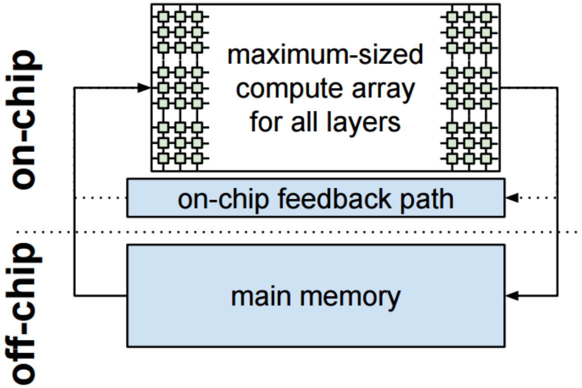
- Introductions
- Framework & Architecture
- Experimental Results
- PYNQ Overlay
- Discussions

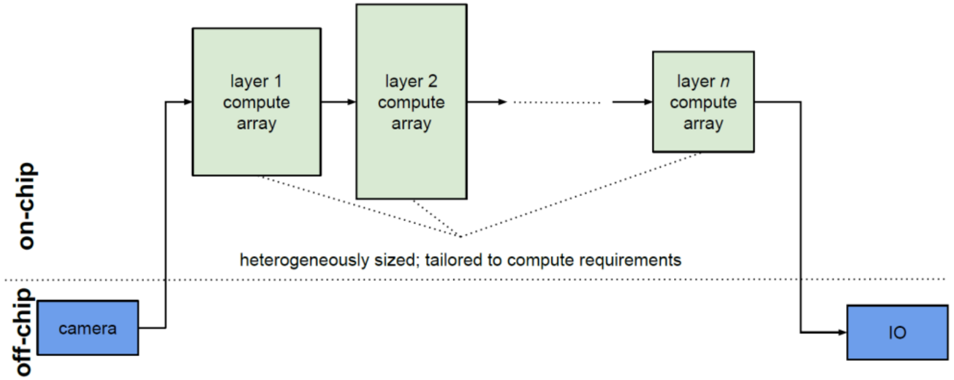
XILINX ALL PROGRAMMABLE™

Architecture

Concepts

- **Memory**
  - Weights and thresholds are contained in on-chip memory
- **Custom heterogeneous streaming architecture**
  - Not a systolic array with scheduling network on processing engines
  - Customized network where all layers coexist in a data flow architecture
  - Each layer consumes and produces in same order to minimize buffering and latency
  - Layers are different instantiations of a C++ template classes (MVTU) with equivalent throughput
- **Custom data types & BNN specific optimizations**
  - {-1/+1} maps to {0,1}
  - Xnor-popcount as cheap binary multiply-accumulates
  - Thresholds as cheap batchnorm activations
  - "OR" becomes cheap maxpool





XILINX ALL PROGRAMMABLE.

Architecture of a Matrix-Vector Threshold Unit (MVTU)

- **Fully connected layers & convolutional layers are mapped on matrix-vector multiply threshold units (MVTUs)**
- **MVTUs support folding over OFMs (neuron) and folding over weights (synaptic)**
- **Weight and output stationary (weights and popcounts are retained locally)**
- **Max pool units are optionally placed behind MVTUs**

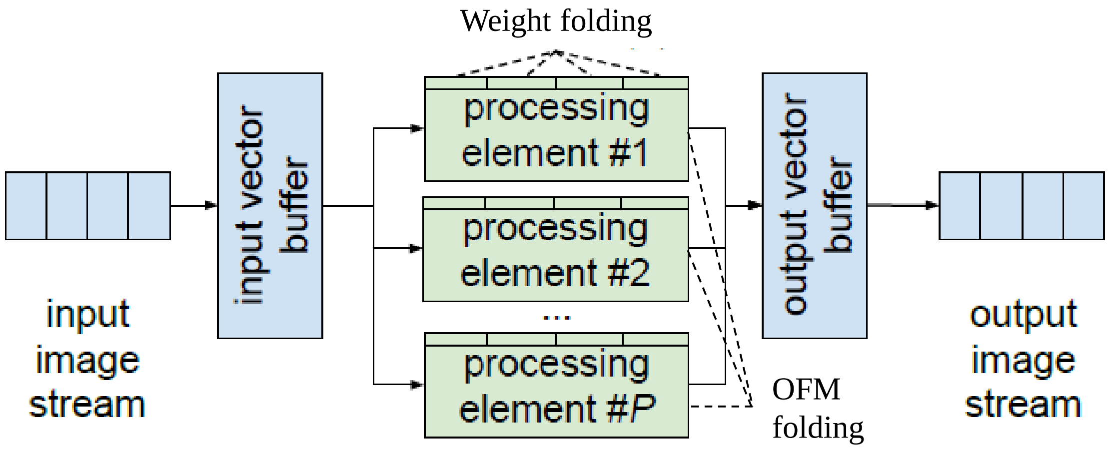


Figure 6: Overview of the MVTU.

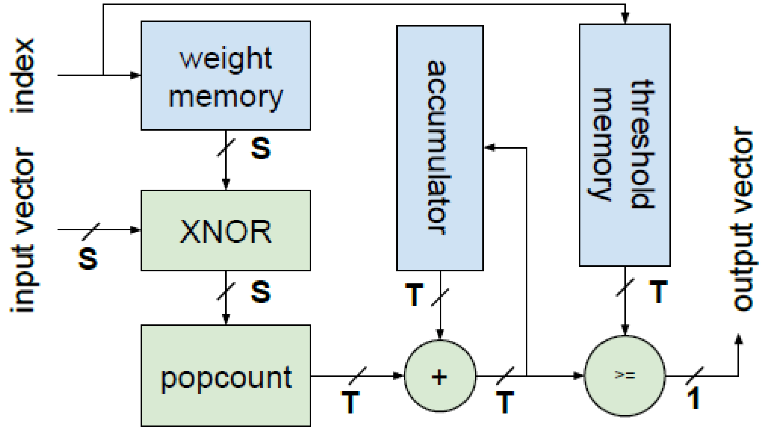
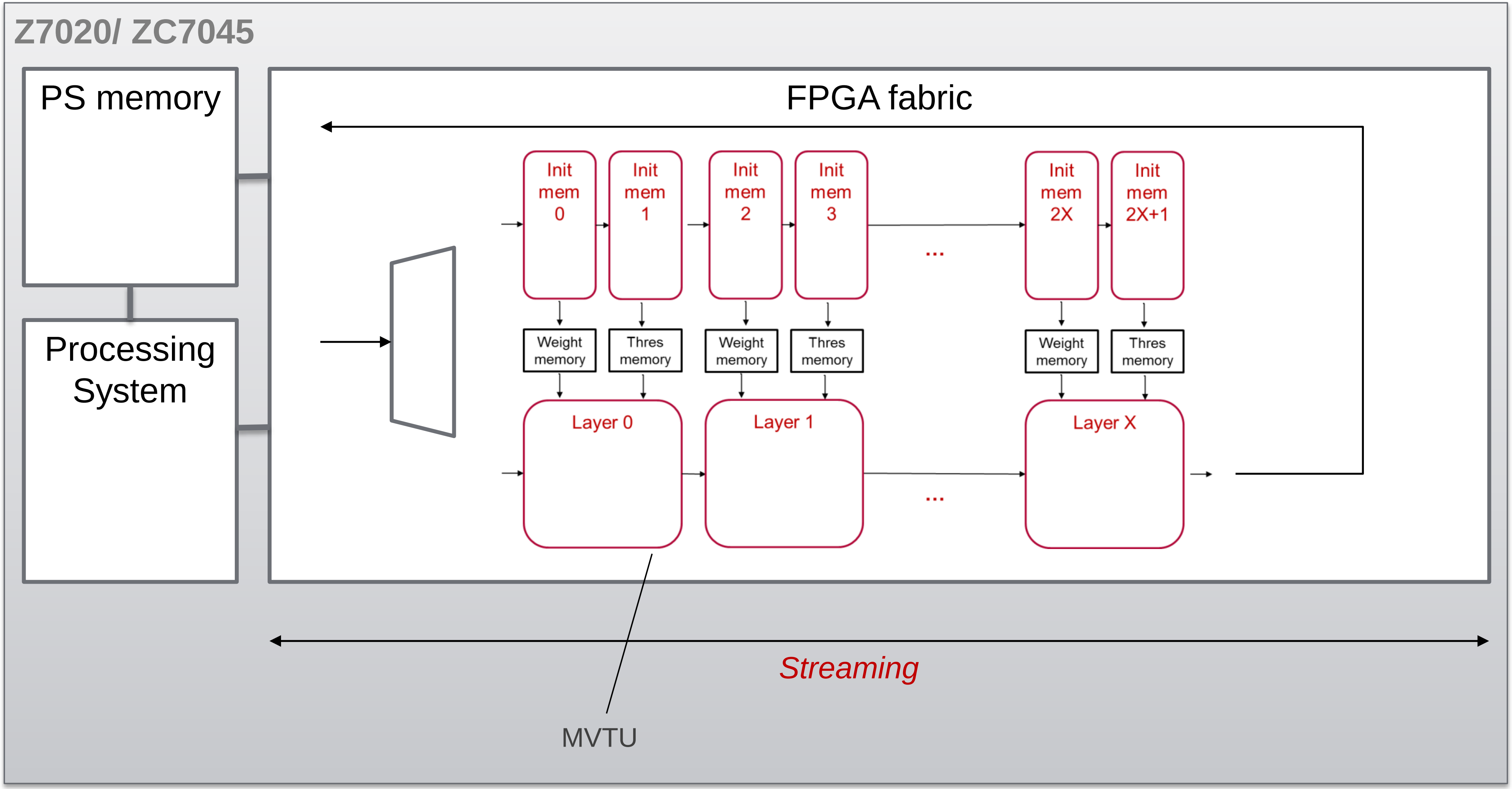


Figure 7: MVTU Processing Element (PE) datapath. **Bold** indicates bitwidth.

Architecture of Infrastructure on Zynq SOC

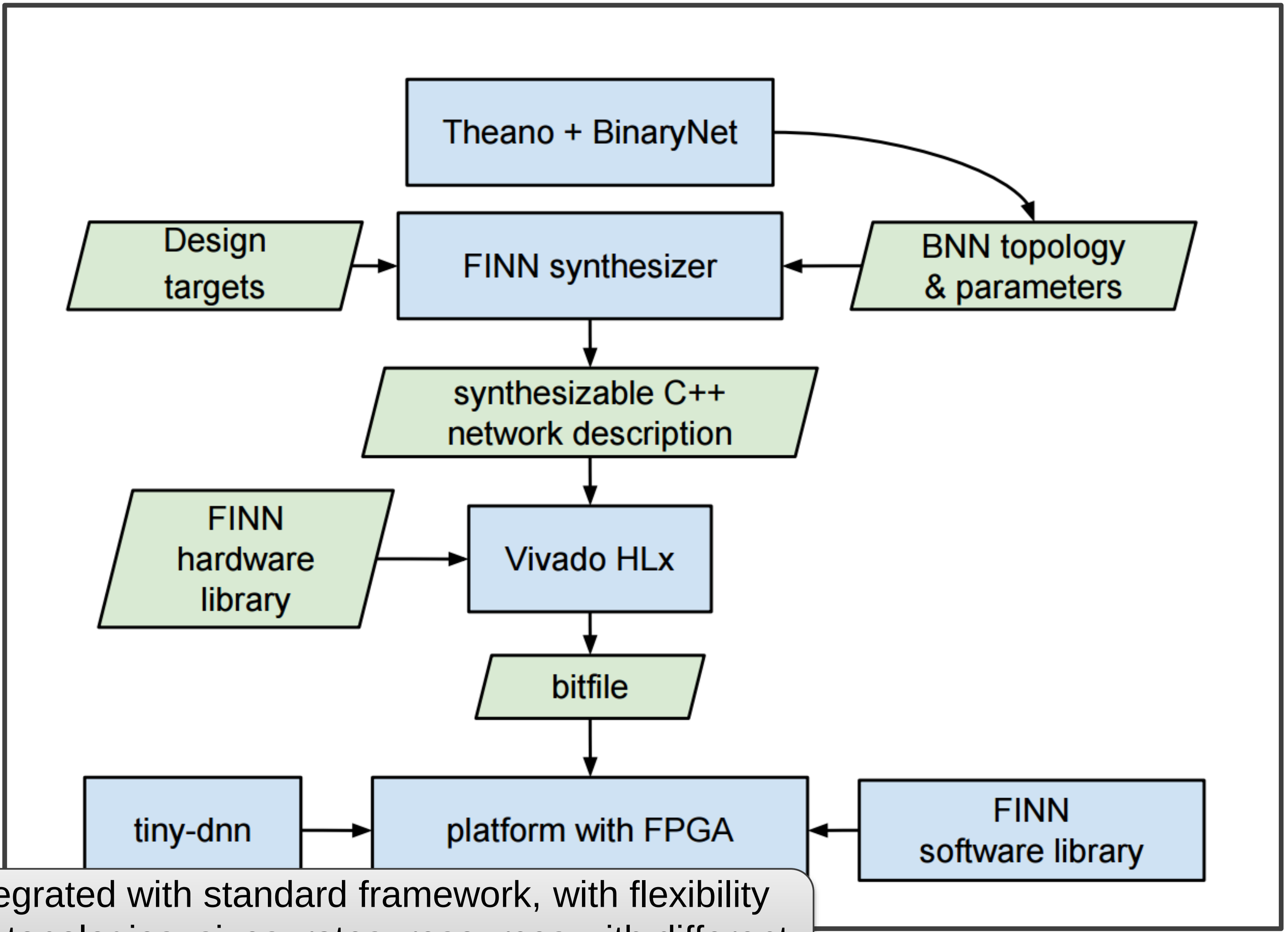


Work Flow for Exploration of BNNs

- First prototype integration with tiny-dnn and Theano (Tensorflow and Caffe in progress)

theano TensorFlow™ Caffe

- All code in C/C++
- Can execute on CPU and FPGA - No RTL needed
- Scheduler is conceptually packed into the synthesizer



Fast workflow, integrated with standard framework, with flexibility to support different topologies, sizes, rates, resources with different devices (Z7045, KU115, Z7020)

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

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)

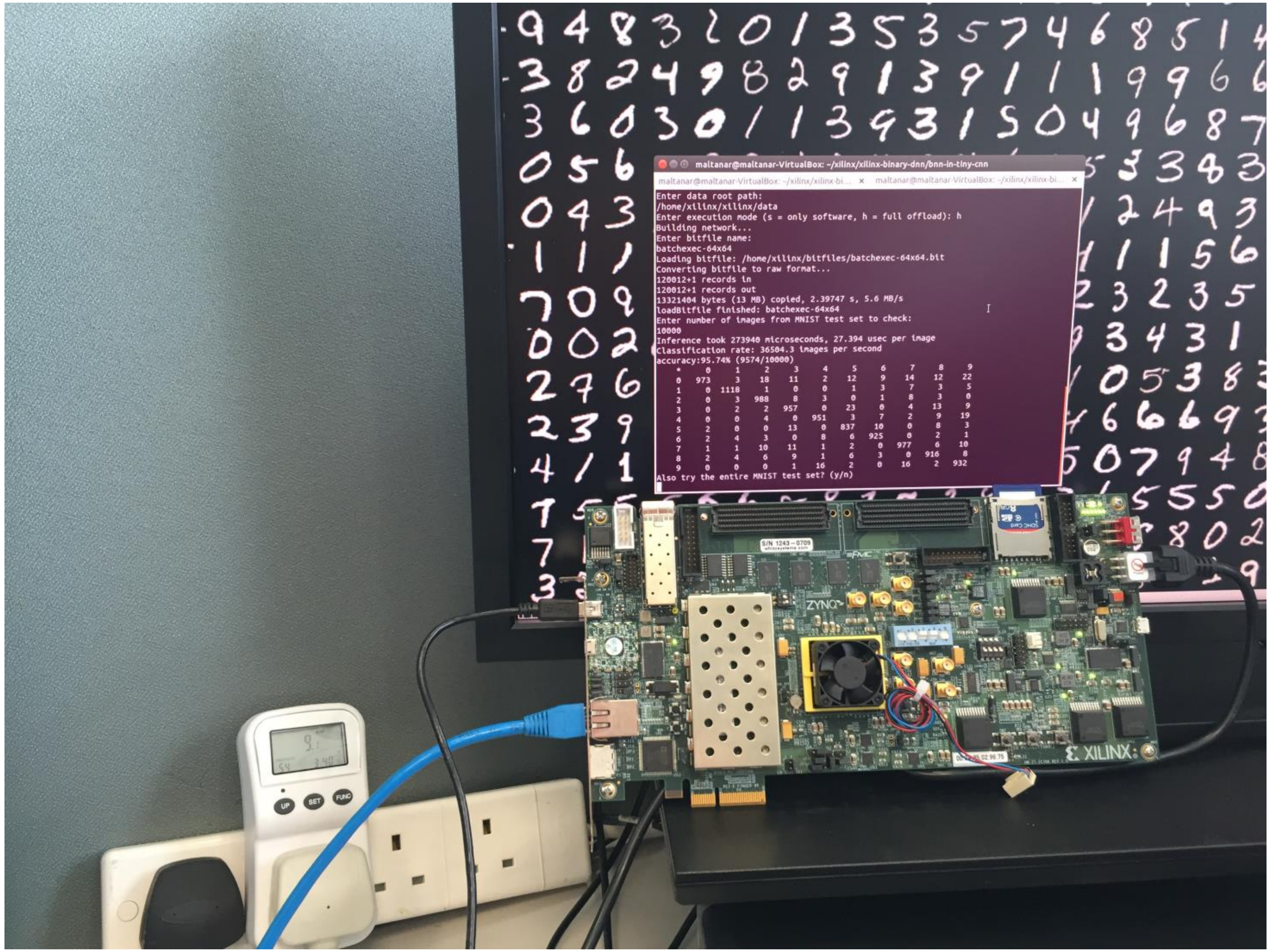
Indexing weight and
threshold memory
binary MAC

Batchnorm
activations

XILINX ALL PROGRAMMABLE.

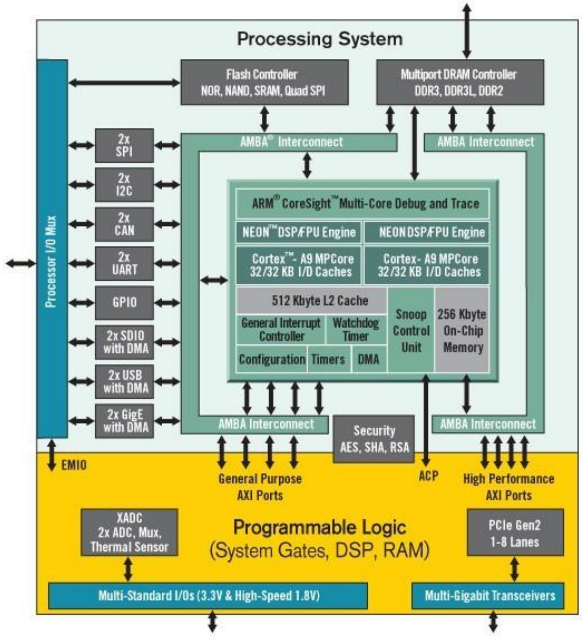
- Introductions
- Framework & Architecture
- Experimental Results
- PYNQ Overlay
- Discussions

Experimental Setup



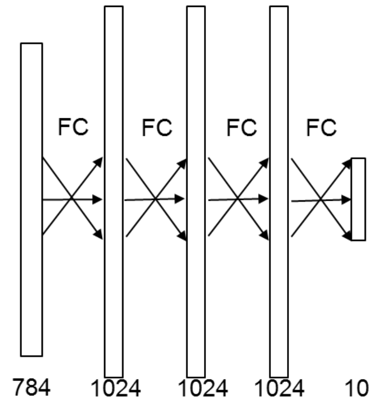
Source: Xilinx Dublin labs – BNN setup

Z706 development platform:
- Z7045
  - 2 A9 processors
  - 350k LUTs
  - 900DSPs
- 2x 1GB DDR3



XILINX ➤ ALL PROGRAMMABLE.

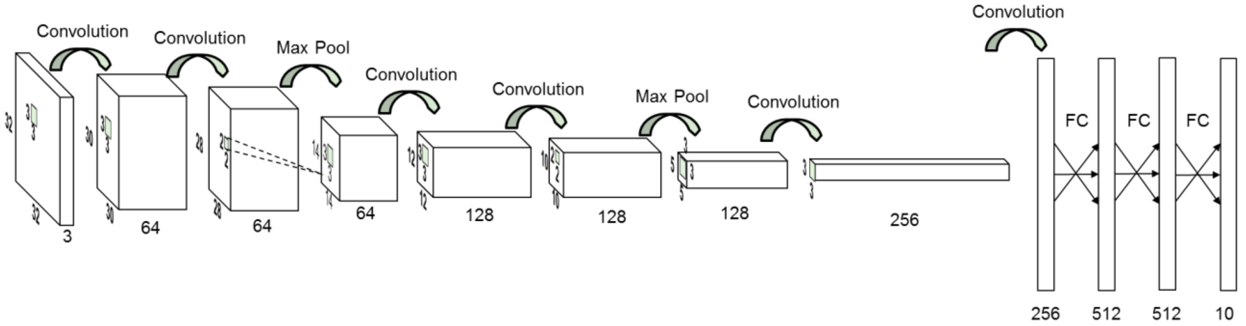
Test Networks



- **Fully connected networks**
  - Input images: 28x28 pixels, binarized MNIST
  - Number of layers: 3 FC layers, 256, 512 and 1024 neurons each
  - Compute requirement: 0.67, 1.86 and 5.8 MOPS/Frame

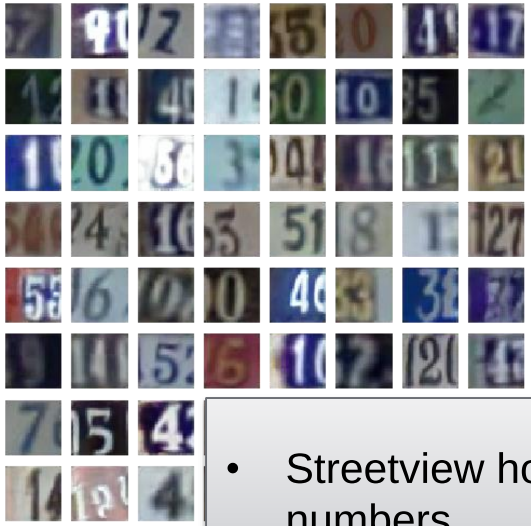
- **CNV (VGG-16 derivative)**
  - Input images: 32x32 pixels, RGB image
  - Number of layers: 2 (3x3) Conv + Max Pool + 2 (3x3) Conv + Max Pool + 2 Convolutional + 3 FC
  - Compute requirement: 0.113 and 1.2 GOPS/Frame

- **DoReFaNet (AlexNet)**
  - Reduced precision with 2b activations
  - Input Images: 227x227, RGB
  - In progress

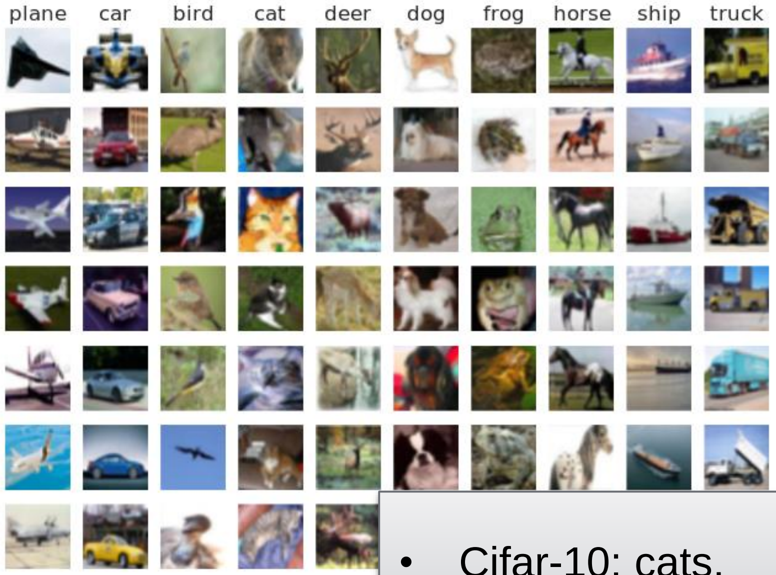


Test Networks & Input Data

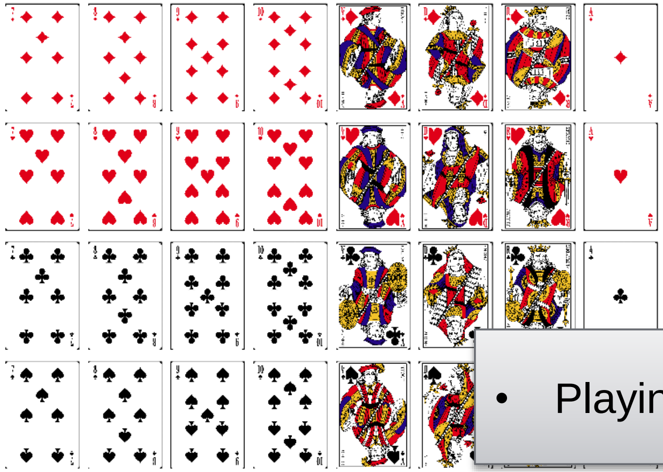
- MNIST handwritten digits

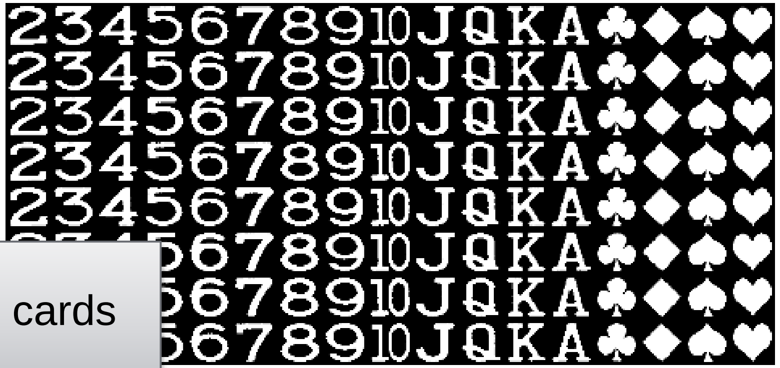


- Streetview house numbers



- Cifar-10: cats, dogs, etc





- Playing cards

XILINX ALL PROGRAMMABLE.

Results - Performance, Latency, Power & Resources

Max Throughput

Z7045	FPS	GOPS/s	BRAM	LUT	Latency [us]	Power [W]
[illegible]	**12.3M**	8’200	130.5	[illegible]	**0.31**	21.2
[illegible]	1.5M	9’085	398	[illegible]	**2.44**	22.6
CIFAR10 - small	21.9K	2’465	192	[illegible]	**283**	11.7

Unprecedented classification rates

Ultra-low latency (P4 ~11ms)
For robotics, AR, UAVs

Z7020 (PYNQ)	FPS	GOPS/s	BRAM	LUT	Latency [us]	Power [W]
MNIST – small	**307k**	203.5	64.5	23’756 (44%)	**13**	-

12K FPS target

Z7045	FPS	GOPS/s	BRAM	LUT	Latency [us]	Power [W]
MNIST - small	12.2k	[illegible]	[illegible]	**4’810 (2%)**	240	8.1
MNIST – large	12.2k	[illegible]	[illegible]	**6’156 (3%)**	282	7.9
CIFAR10 - small	11.6k	[illegible]	[illegible]	40’404 (18%)	550	10

Scalability to extremely small footprints

Comparable to AlexNet

KU115	FPS	GOPS/s	[illegible]	[illegible]	Latency [us]	Power [W]
CIFAR10 - [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

3x classification rate over best measured numbers on GPU today

- High performance, latency, low power with:
  equal accuracy on small networks and promising results for larger networks

Machine Learning Applications

Applications that require large networks and low accuracy (performance, power)

- Recommender systems
- Data analysis

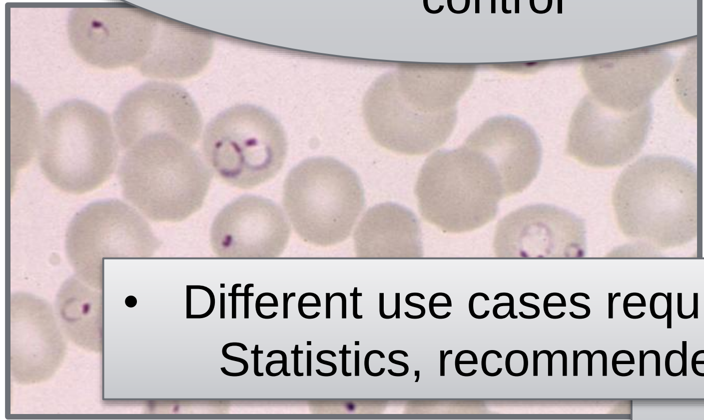
Applications that require small networks (low latency & speed)

- Wireless: channel equalization
- High Frequency Trading
- Identifying malaria cells
- Speech recognition for voice control

Applications that require large networks and high accuracy

- Autonomous driving

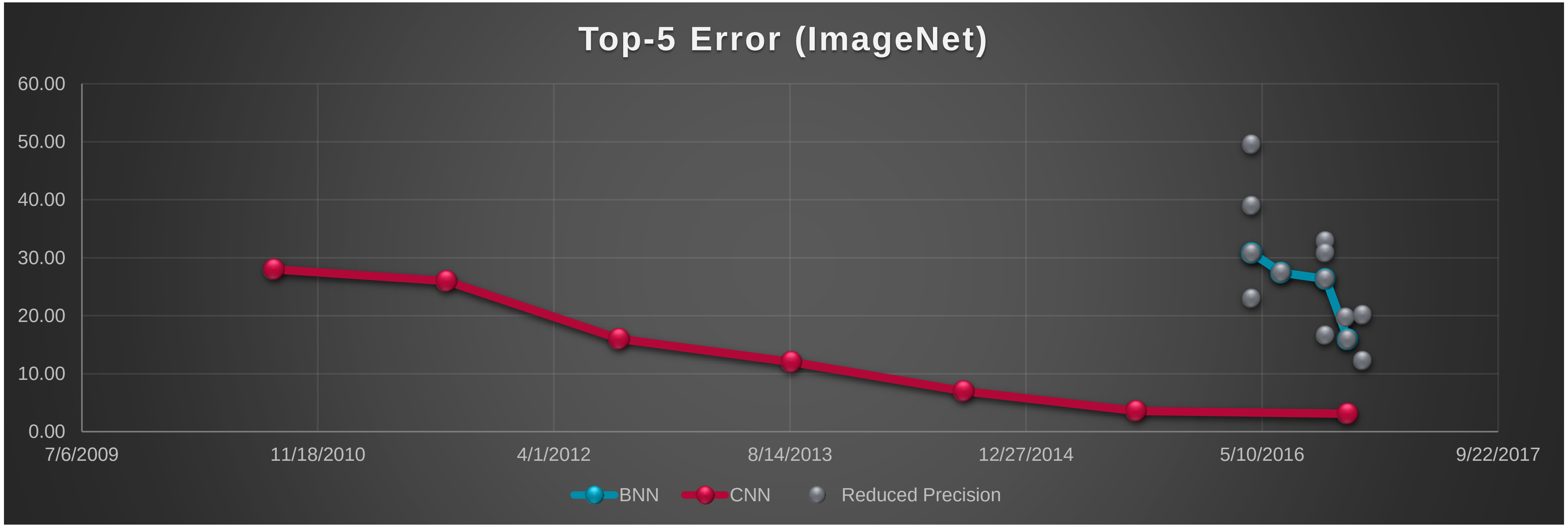
?



- Different use cases require different networks & different levels of accuracy
  Statistics, recommender systems, UAV and medical diagnosis have very different requirements

Accuracy of Binary Networks Improving

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



- BNNs are new and accuracy results are improving rapidly

XILINX ALL PROGRAMMABLE.

Others are considering it too

Facebook, Google, Intel

Soumith Chintala
@amiconfusediam
Follow

Just read the XNor-net paper. Great work, will change how we all do production convnets. arxiv.org/abs/1603.05279

Andrej Karpathy
@karpathy
Follow

It's fun watching the innovations made in binarizing ConvNets arxiv.org/abs/1603.05279 binary is the way to go eventually.

WIRED — Google Opens Montreal AI Lab to Snag Scarce Global Talent

BUSINESS | CULTURE | DESIGN | GEAR | SCIENCE | SECURITY

SHARE
SHARE 7338

CADE METZ BUSINESS 11.21.16 6:00 AM

GOOGLE OPENS MONTREAL AI LAB TO SNAG SCARCE GLOBAL TALENT

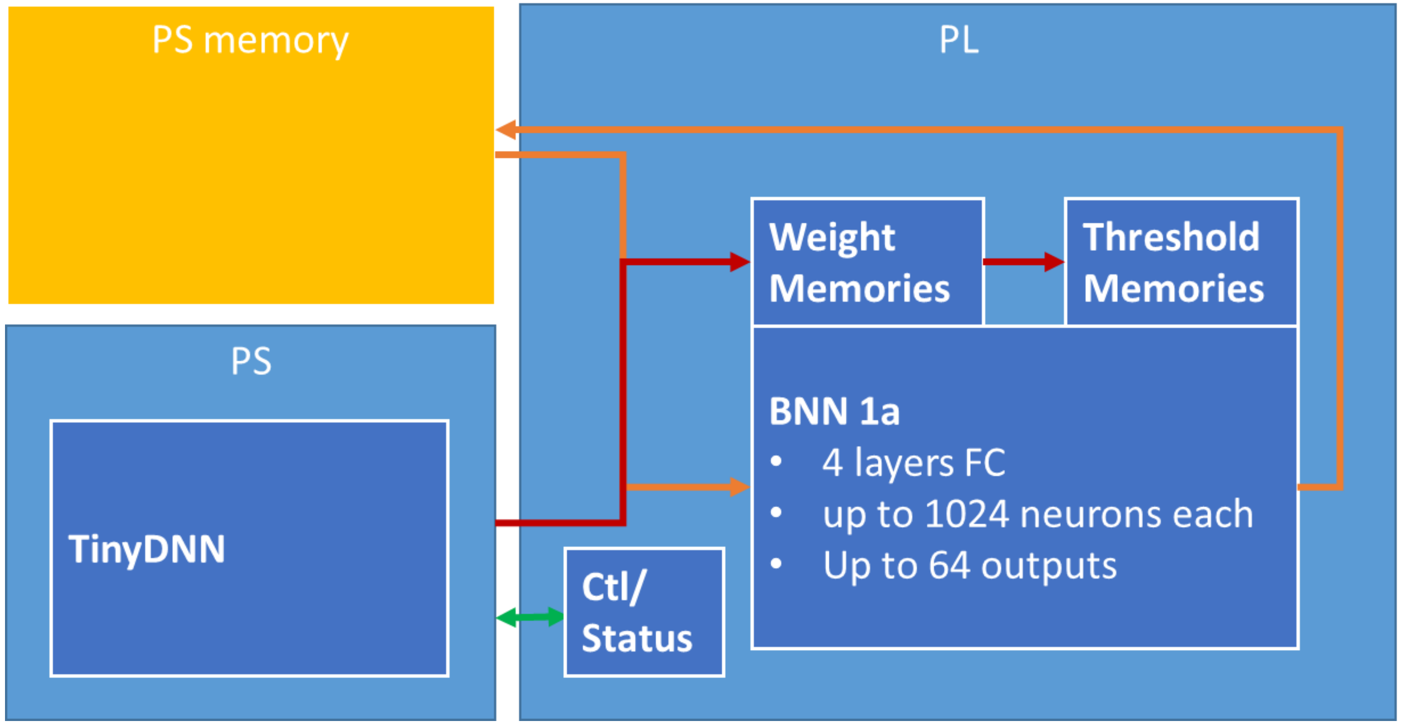
Pete Warden
@petewarden
Follow

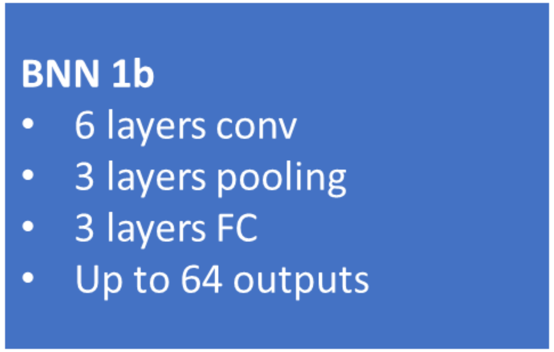
A great binary neural network implementation using XNOR that gets 66% top-1 precision on Imagenet: arxiv.org/pdf/1603.05279…

- Introductions
- Framework & Architecture
- Experimental Results
- PYNQ Overlay
- Discussions

PYNQ Overlay Architecture for BNN

First release: Rigid networks with high performance, basic tool support

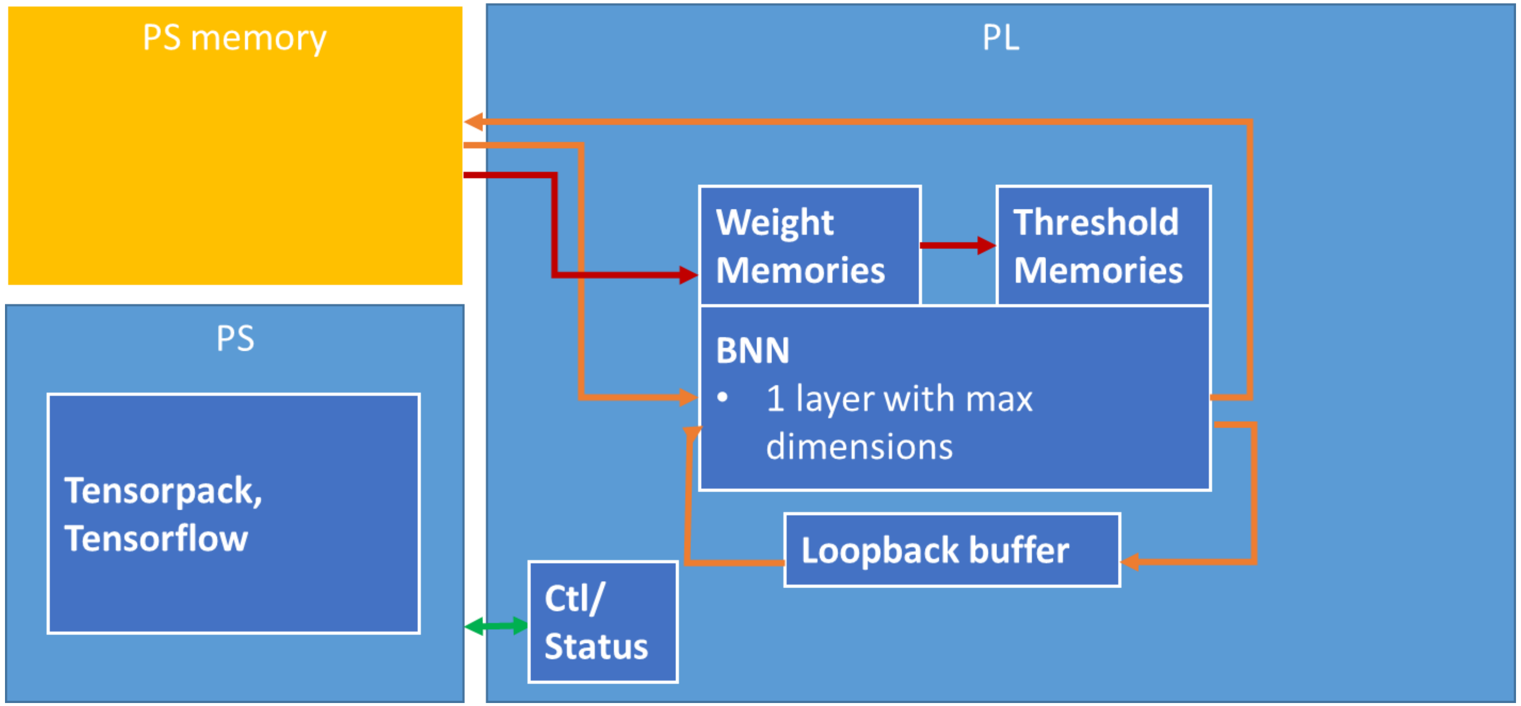




- 1a and 1b support fixed topologies that fit into the given foot print
- Classify images up to 28x28 pixels (1a) or 32x32 (1b)
- Very high classification speeds (1a => 70kfps, 1b => 6kfps?), very low latency (<1ms)
- Example use cases: solitaire, handwriting, small colour images, HFT, speech recognition (voice control for robots)
- (While smaller networks can be mapped onto this architecture, not sure it helps other than training time)

PYNQ Overlay Architecture for BNN

*Second release: **Flexible** networks, **lower** performant, high power efficiency*



- **2 supports networks that consist of convolutional, max pool,**
- **Classify images up to 32x32 pixels (1b or 24b)**
- **Value: Energy efficiency, experimental platform to get comfortable with FPGAs, gaining trust**
- **Example use cases: ImageNet or vision processing tasks**

Software Flow

theano
TensorFlow™
Caffe

Design network

Training

Train inside tensorflow on new dataset (parameterizable container with AWS scripts)

Import Data

Import data set into AWS

Export

Export weights & topology

Start network

- Generate instructions for overlay
- Configures fabric
- Writes instructions
- Reads npy, converts into internal formats and streams into weight and threshold memory

theano
TensorFlow™
Caffe

Run-time (ARM):

- Classify n images
- Visualize results
- Estimate
- performance
- Inside tensorpack?

Run-time (FPGA):

- Classify n images
- Visualize results
- Report performance
- Inside tensorpack?

XILINX ALL PROGRAMMABLE.

Provisional Timelines

- **Early release of 1a and 1b for FPGA 2017 with Theano and fixed networks**
- **Release of 2 end of March 2017 with Tensorpack and flexible network design**

- Introductions
- Framework & Architecture
- Experimental Results
- PYNQ Overlay
- Discussions

XILINX ALL PROGRAMMABLE.

Summary

- **Binary networks provide some interesting performance – resource (cost) trade-offs within the design space**
  - Extremely small footprint for slower classification rates for smallest devices (4,6k (2%) LUTs for 60fps)
  - Very high classification rates (12Mfps)
  - Low latency for applications with real-time requirements such as AR, automotive and robotics

- **Proposed architecture is flexible to support different types of neural network topologies (all in C/C++)**
  - Number of layers
  - Size and types of layers (convolutions, max pool, fully connected)
  - Experiment without hardware knowhow

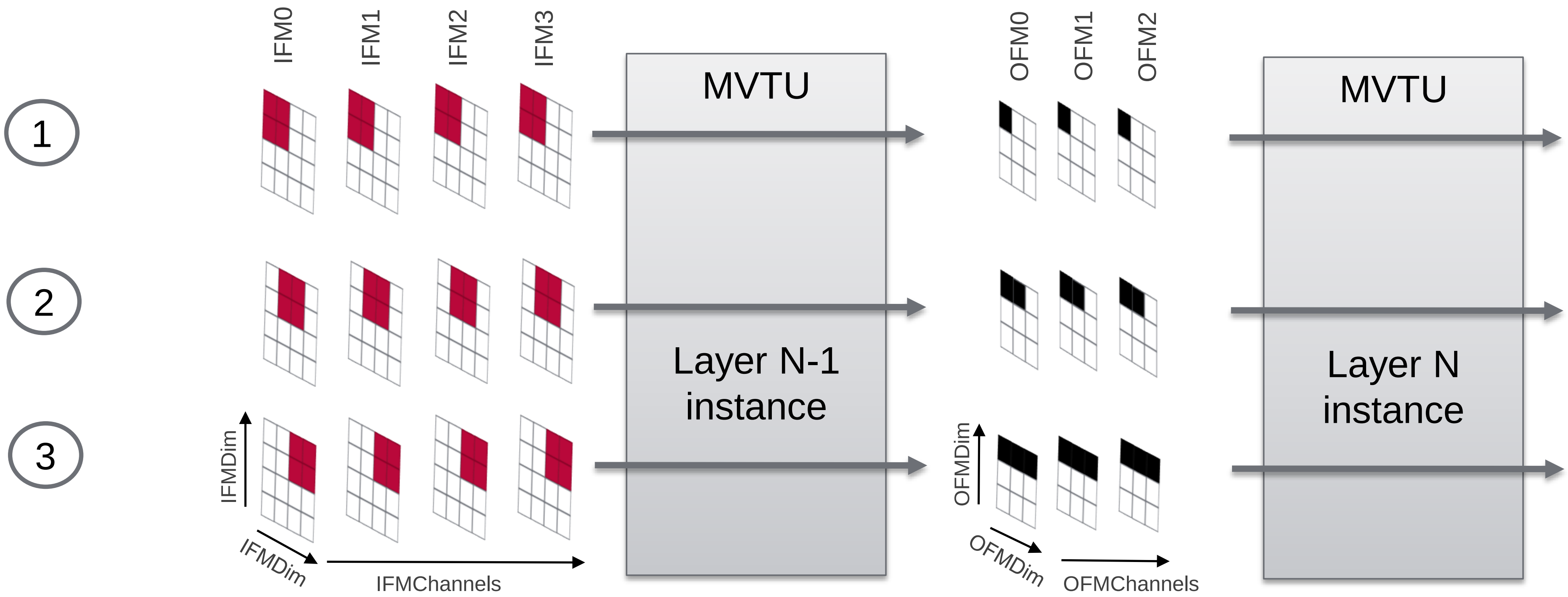
- **PYNQ provides a release mechanism that makes technology available to a wide audience**

Many Open Questions

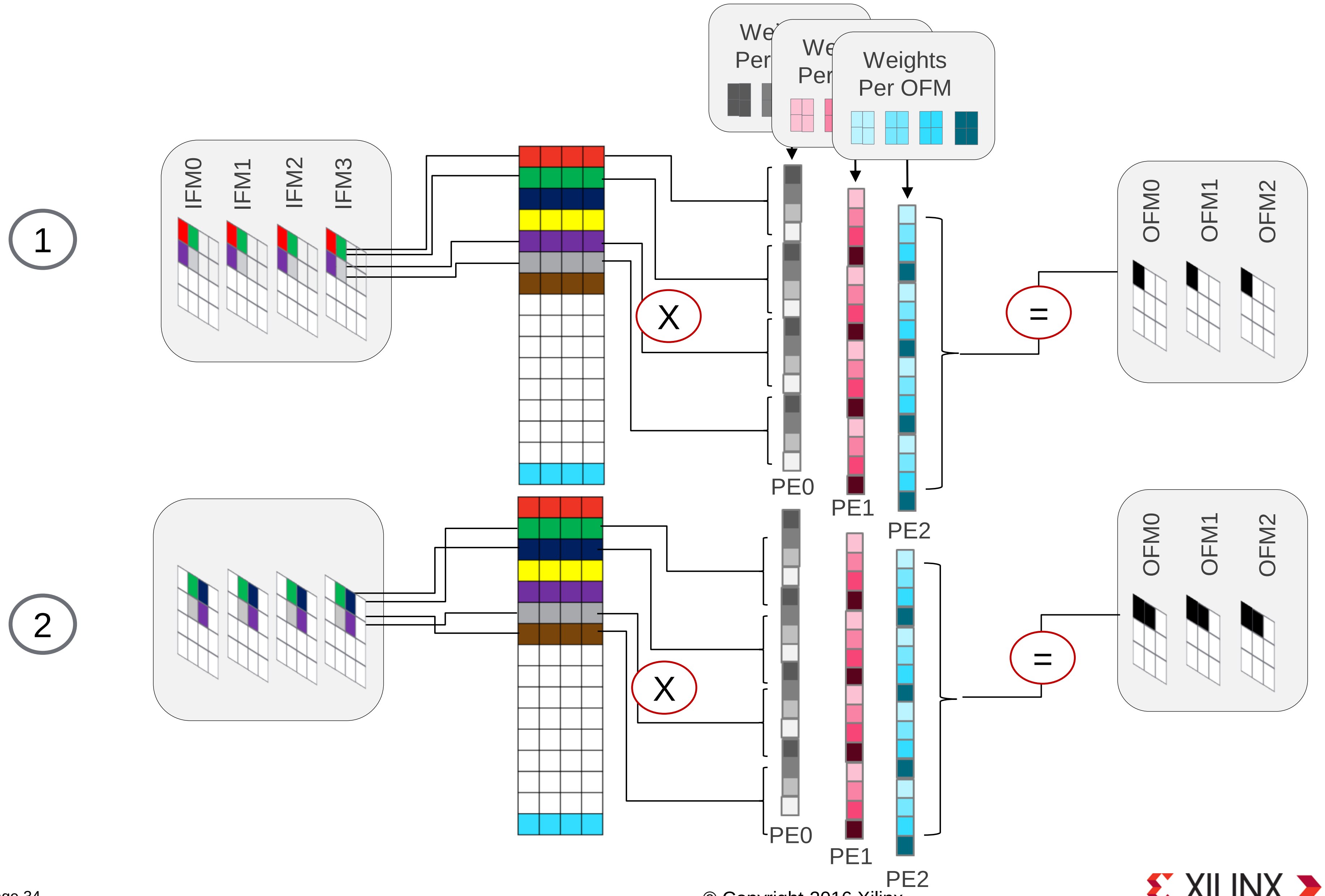
- **Real use cases**
  - – Medical images?
- **Accuracy**
  - – Research needed in large binarized neural networks with high accuracy
  - – Design space exploration/navigation accuracy- resource – frame rate
- **Adaptation & Integration & Cloud service of standard tool chains (for example Caffe)**
- **Performance comparisons with GPUs, CPUs, Phis**
  - – Microbenchmarks
- **Architecture**
  - – Improving architecture & adding more precision flexibility & adding inception & skip layers
  - – Pruning & sparse representations

- Thank You.

Consuming and Producing in Same Sequence
For Minimal Buffering & Latency



IFM & Weight Arrangements in Input Buffer & Sequencing

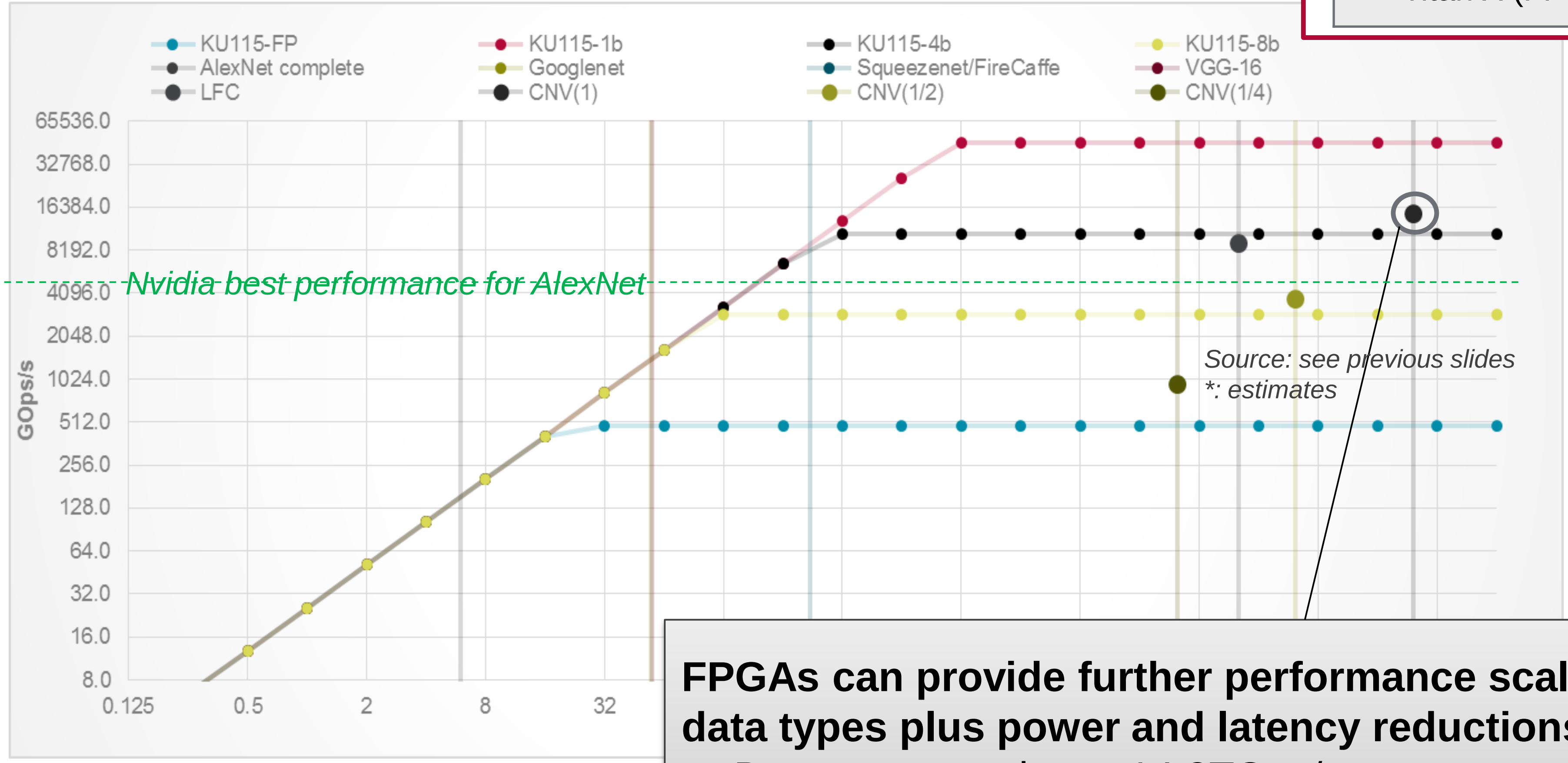


XILINX ALL PROGRAMMABLE.

Experimental Results (Server)

State of the Art - Comparison

Platform * - estimated	[GOPS/s]
Titan X (FP32)	5’750



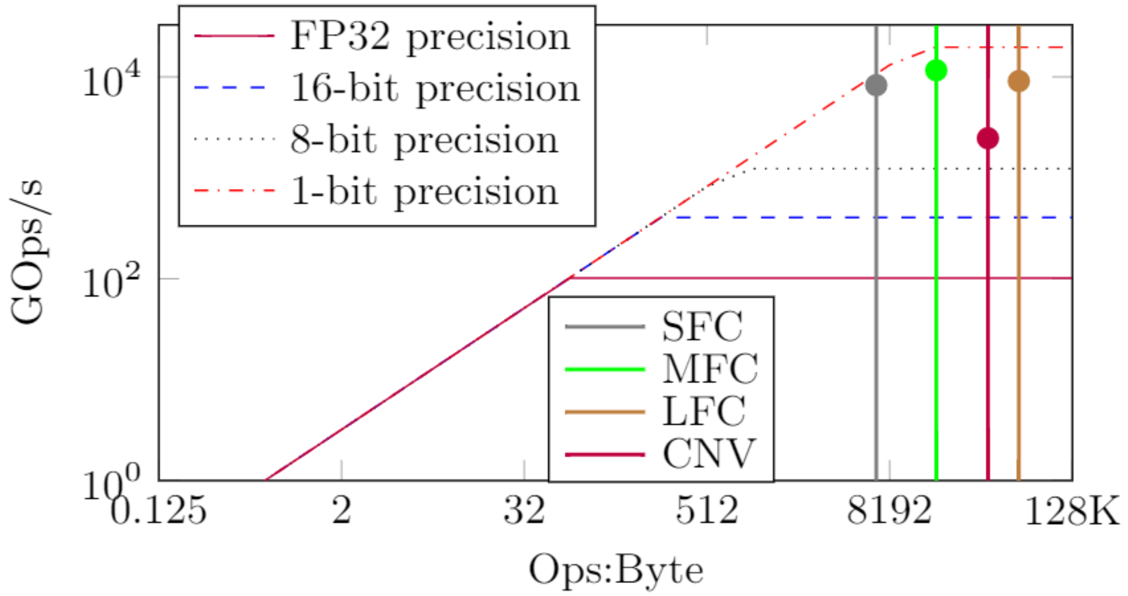
FPGAs can provide further performance scaling using custom data types plus power and latency reductions

- Demonstrator shows 14.8TOps/s
- Potential to scale up to 46 TOps/s for data center applications
- Consuming < 41Watt

XILINX ALL PROGRAMMABLE.

Experimental Results (Embedded)

Z7045 (ZC706)



State of the Art - Comparison

Platform * - estimated	[GOPS/s]
Tegra X1 (FP16)	335

Source: see previous slides
**: estimates*

Roofline Assumptions:
Z7045:
- 218k LUTs
- 900 DSPs
- 545 BRAM

Network	FPS	GOPS/s	BRAM	LUT	Latency [us]	Board Power [W]
FC	12.3M	8’200	130.5	86’110 (39%)	0.31	21.2
FC	12.2k	0.66	15.5	4’810 (2%)	240	8.1
CNV	21.9K	2’465	192	54’538 (25%)	283	11.7

Results- Power and Latency

Network	FPS	Latency [us]	Power (a) [W]	Power (b) [W]
SFC	12.3M	0.31	7.3	21.2
LFC	1.5M	2.44	8.8	22.6
CNV	21906	283	3.6	11.7

Network	FPS	Latency [us]	Power (a) [W]	Power (b) [W]
SFC	12.2k	240	0.43	8.1
LFC	12.2k	282	0.8	7.9
CNV	11.6k	550	2.3	10

Network	FPS	Latency [us]	Power (a) [W]	Power (b) [W]
SFC	996	43029	0.4	8
LFC	190	14551	0.3	7.4
CNV	6.83			

(a) PL power
(b) Board level power

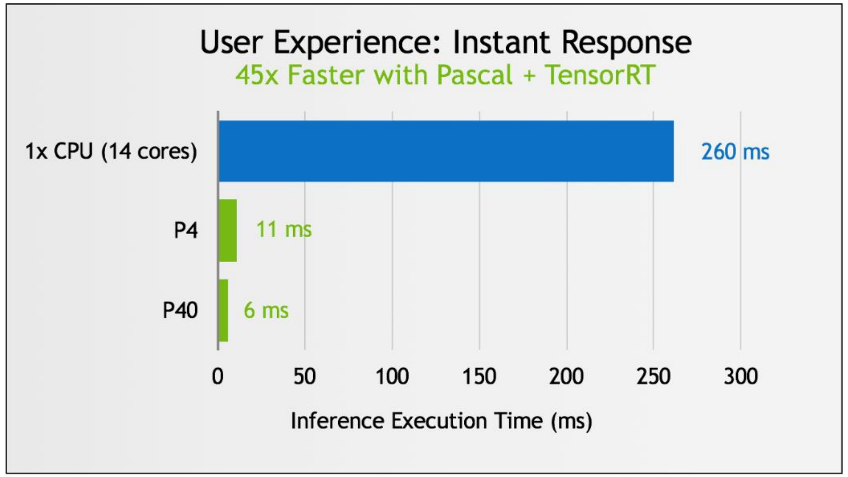


Figure 3: Inference execution time on Tesla P4 and P40 using TensorRT 2 to optimize the trained neural network, compared to IntelCaffe on CPU. (Based on VGG-19 from IntelCaffe Github. CPU: IntelCaffe, batch size = 4, Intel E5-2690v4, using Intel MKL 2017. GPU: Caffe, batch size = 4, using TensorRT internal version.)

Source:
https://devblogs.nvidia.com/parallelforall/new-pascal-gpus-accelerate-inference-in-the-data-center/

Very Low latency
- Small network: 12Mfps, 8.2TOPS/s: 310nsec latency
- Large network: 21.9Mfps, 2.5TOPS/s: 283usec latency

Required for real-time applications

XILINX ALL PROGRAMMABLE.