Bridge of Life Education

Fast, Scalable Quantized Neural Network Inference on FPGAs

Lecturer: Hua-Yang Weng

Date: 2022/08/18

[FPGA'17: FINN: A Framework for Fast, Scalable Binarized Neural Network Inference]
(https://arxiv.org/abs/1612.07119)

©BOLEDU

Outline

- Introduction to FINN
- Network Define
- NN Hardware
- Lab Description

Outline

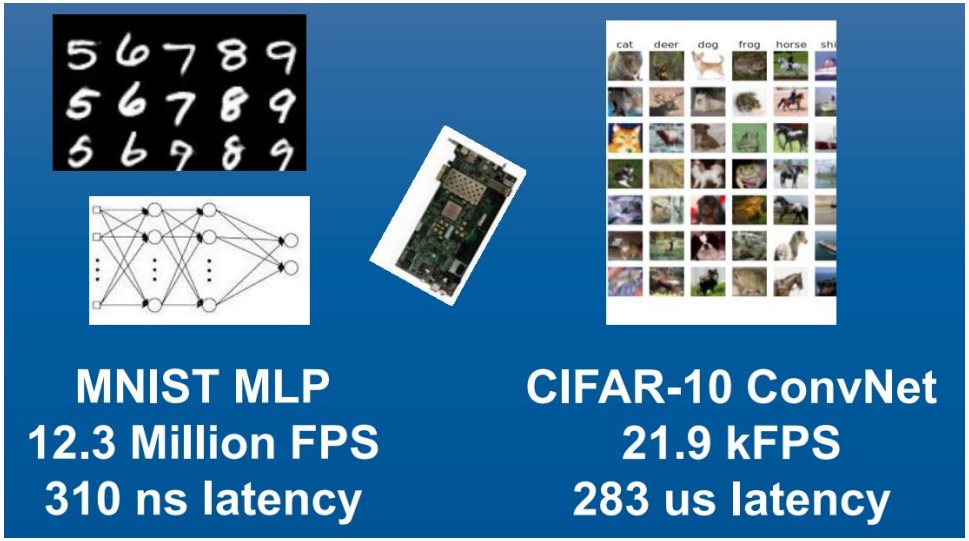
- Introduction to FINN
- Network Define
- NN Hardware
- Lab Description

FINN: The Beginning (FPGA'17)

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

Yaman Umuroglu*†, Nicholas J. Fraser*‡, Giulio Gambardella*, Michaela Blott*,
Philip Leong‡, Magnus Jahre† and Kees Vissers*
*Xilinx Research Labs; †Norwegian University of Science and Technology; ‡University of Sydney
yamanu@idi.ntnu.no

Finn: A framework for fast, scalable binarized neural network inference
Y Umuroglu, NJ Fraser, G Gambardella... - Proceedings of the ..., 2017 - dl.acm.org
Research has shown that convolutional neural networks contain significant redundancy, and high classification accuracy can be obtained even when weights and activations are reduced from floating point to binary values. In this paper, we present FINN, a framework for building fast and flexible FPGA accelerators using a flexible heterogeneous streaming architecture. By utilizing a novel set of optimizations that enable efficient mapping of binarized neural networks to hardware, we implement fully connected, convolutional and ...
☆ 儲存 99 引用 被引用 642 次 相關文章 全部共 9 個版本 »



Publications

https://xilinx.github.io/finn/publications

- ACM TRETS: Elastic-DF: Scaling Performance of DNN Inference in FPGA Clouds through Automatic Partitioning
- FPGA'21: S2N2: A Streaming Accelerator for Streaming Spiking Neural Networks and repository on GitHub
- FPT'20: Memory-Efficient Dataflow Inference for Deep CNNs on FPGA
- IEEE ToC: Evaluation of Optimized CNNs on Heterogeneous Accelerators using a Novel Benchmarking Approach
- FPL'20: LogicNets: Co-Designed Neural Networks and Circuits for Extreme-Throughput Applications
- FCCM'20: High-Throughput DNN Inference with LogicNets
- GECCO'20: Evolutionary Bin Packing for Memory-Efficient Dataflow Inference Acceleration on FPGA
- FPGA'20: Evaluation of Optimized CNNs on FPGA and non-FPGA based Accelerators using a Novel Benchmarking Approach
- ACM JETC: QuTiBench: Benchmarking neural networks on heterogeneous hardware
- ACM TRETS: Optimizing bit-serial matrix multiplication for reconfigurable computing
- FPL'18: FINN-L:Library Extensions and Design Trade-off Analysis for Variable Precision LSTM Networks on FPGAs
- FPL'18: BISMO: A Scalable Bit-Serial Matrix Multiplication Overlay for Reconfigurable Computing
- FPL'18: Customizing Low-Precision Deep Neural Networks For FPGAs
- ACM TRETS, Special Issue on Deep Learning: FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks
- ARC'18: Accuracy to Throughput Trade-Offs for Reduced Precision Neural Networks on Reconfigurable Logic
- CVPR'18: SYQ: Learning Symmetric Quantization For Efficient Deep Neural Networks
- DATE'18: Inference of quantized neural networks on heterogeneous all-programmable devices
- ICONIP'17: Compressing Low Precision Deep Neural Networks Using Sparsity-Induced Regularization in Ternary Networks
- ICCD'17: Scaling Neural Network Performance through Customized Hardware Architectures on Reconfigurable Logic
- PARMA-DITAM'17: Scaling Binarized Neural Networks on Reconfigurable Logic
- FPGA'17: FINN: A Framework for Fast, Scalable Binarized Neural Network Inference
- H2RC'16: A C++ Library for Rapid Exploration of Binary Neural Networks on Reconfigurable Logic

FINN – Project Mission

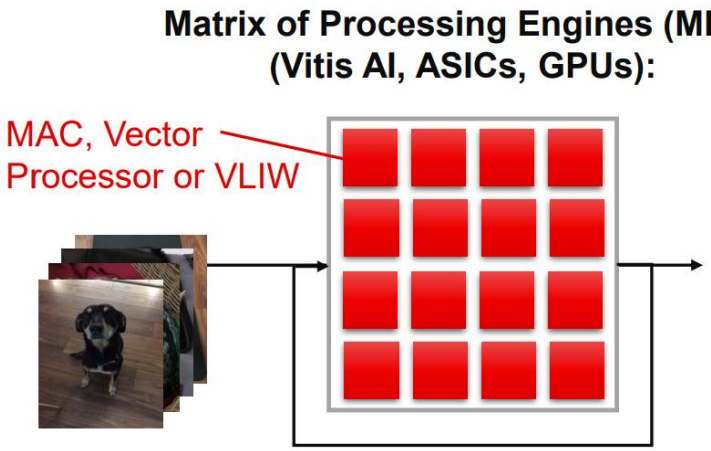
- Mission
  - Tools and platforms for creation of high throughput, ultra-low latency DNN compute architectures
- End-to-end flow
  - Users can easily create specialized hardware architectures on an FPGA and benefit from custom architectures and custom precision
- Open source
  - Transparency and flexibility to adapt to end-users' applications

Two Key Techniques for Customization in FINN

- Streaming Dataflow Architectures with FPGAs & FINN
- Custom Precision Few-bit weights & activations

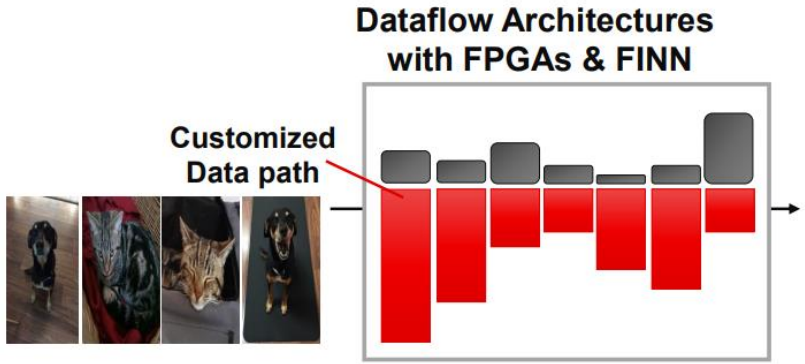
©BOLEDU

Customized Dataflow Processing versus More Generic Architectures



Customized for typical DNN operations

- multiply accumulate
- Lower throughput (~10KRps)
- Flexibility for ASICs
- Applications: CV, Speech

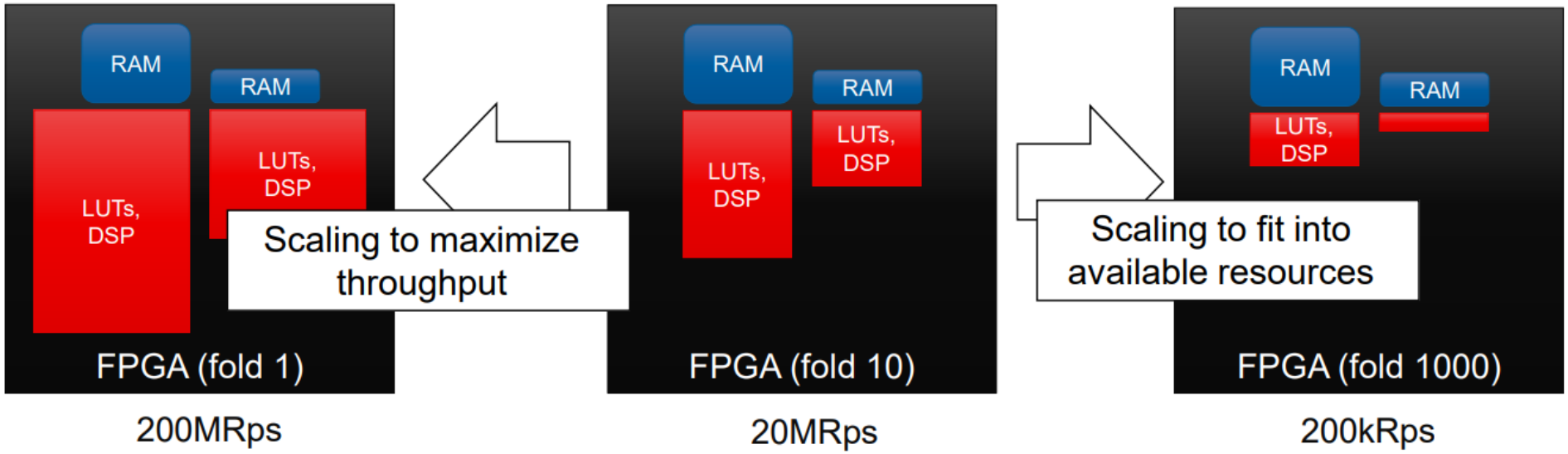


Customized/adapt for specific DNN topologies

- Streaming interfaces
- Specialization -> higher efficiency
- Lower latency (no intermediate buffering)
- Higher throughput (~100MRps)
- Flexibility through reconfiguration
- Applications: smaller DNNs

Dataflow Processing

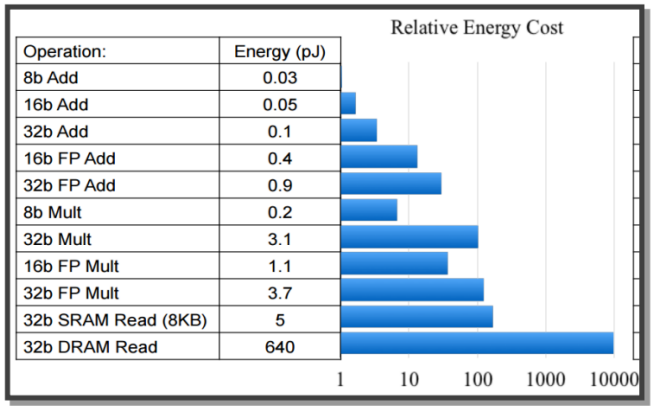
- Scale performance & resources to meet the application requirements
- If resources allow, we can completely unfold to create a circuit that inferences at clock



Customizing Arithmetic to Minimum Precision Required

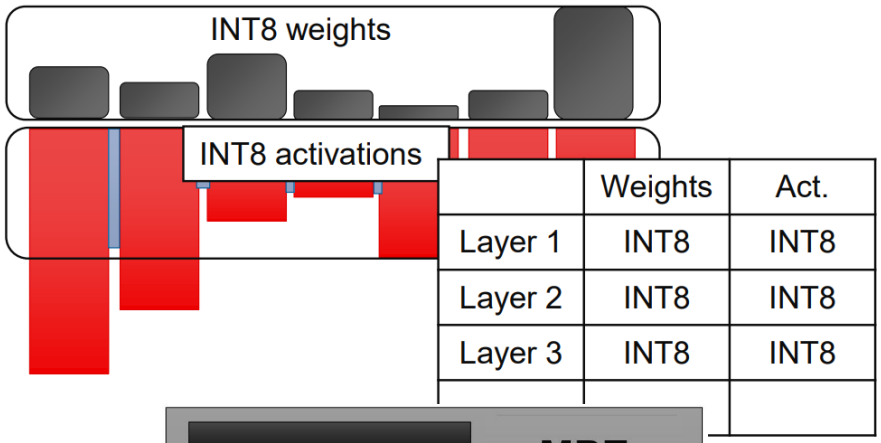
- Reducing precision shrinks hardware cost/ scales performance
  - Instantiate n-times more compute within the same fabric, thereby scale performance n-times
  - 8b/8b -> 1b/1b, RTL => 70x
- Potential to reduce memory footprint
  - NN model can stay on-chip => no memory bottlenecks
- Inherently saves power

Precision	Modelsize [MB] (ResNet50)
1b	3.2
8b	25.5
32b	102.5

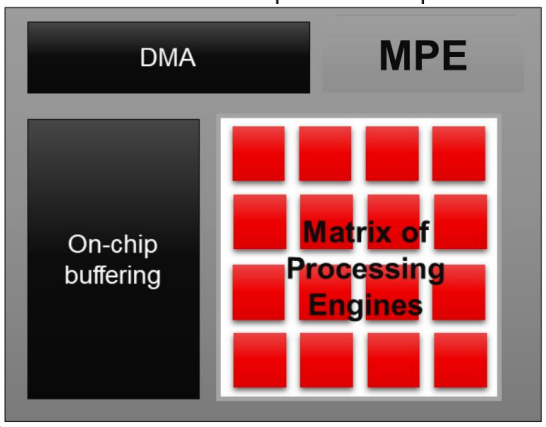


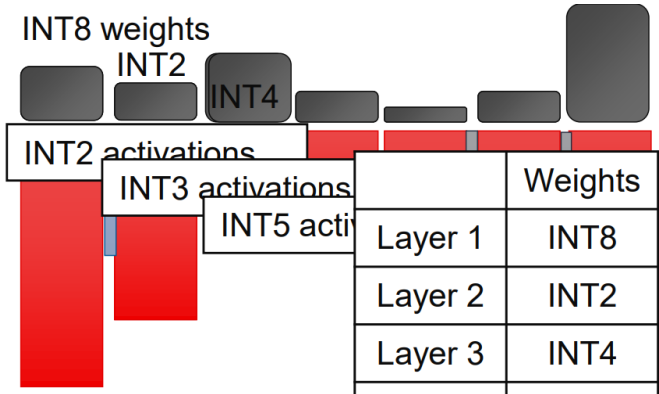
Granularity of Customizing Arithmetic

- Dataflow architectures can exploit custom arithmetic at a greater degree

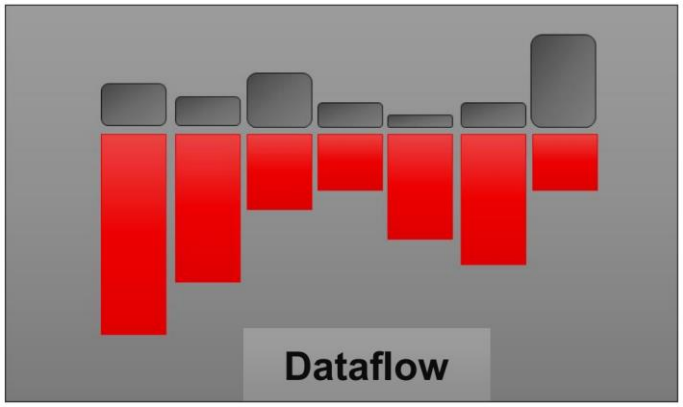


	Weights	Act.
Layer 1	INT8	INT8
Layer 2	INT8	INT8
Layer 3	INT8	INT8



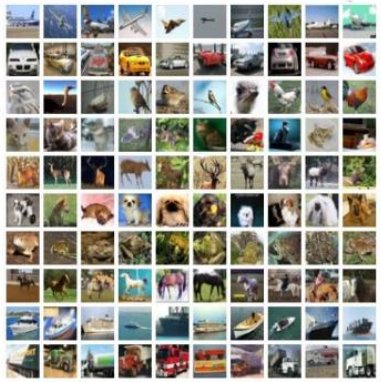


	Weights	Act.
Layer 1	INT8	INT2
Layer 2	INT2	INT3
Layer 3	INT4	INT5
…		



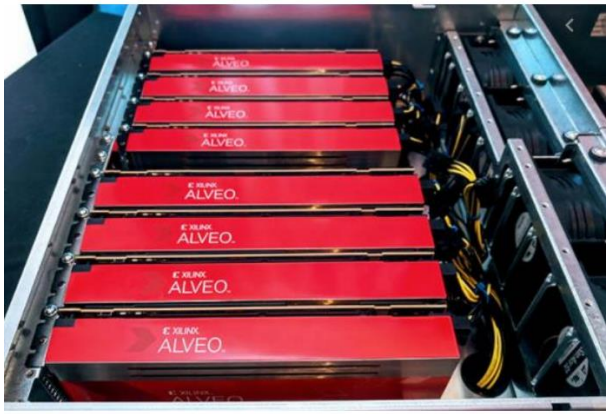
Few-bit DNNs + FPGA Dataflow: Showcases

Low-Power, Real-Time Image Classification



CIFAR-10 CNV on PYNQ-Z1
3kFPS @ 2.5 W
1ms latency

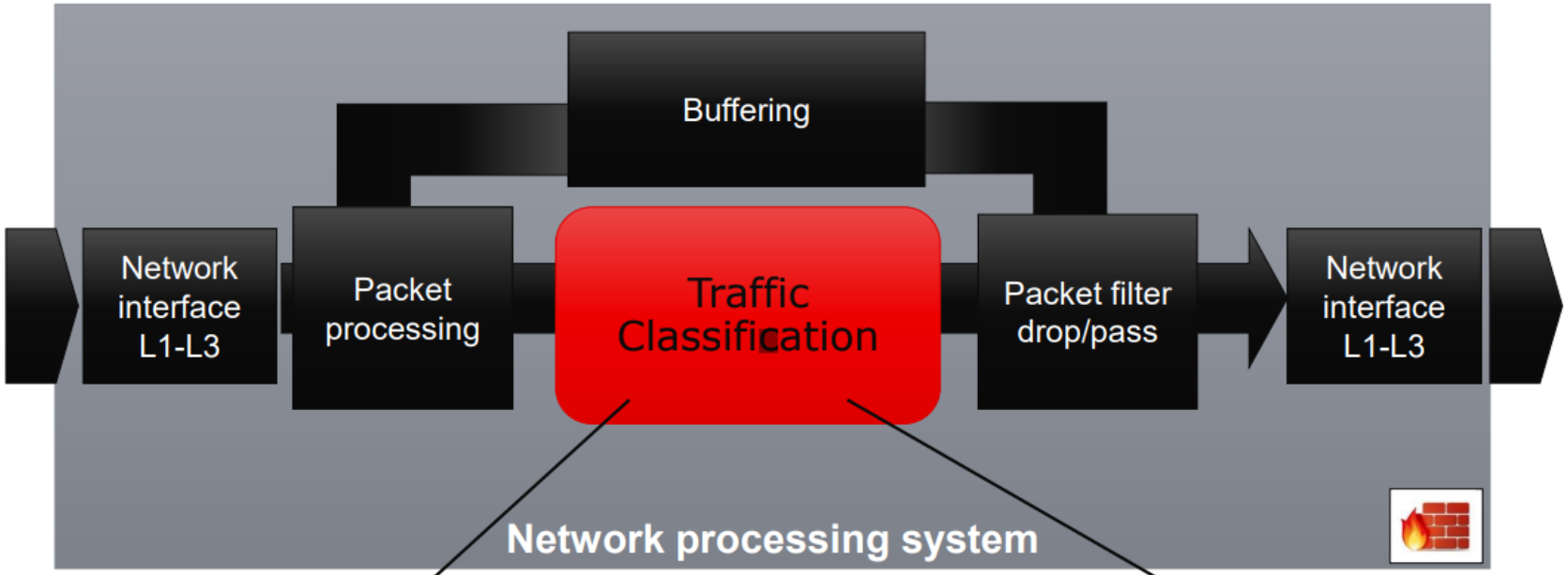
Single and multi-node ImageNet Classification



MNv1: 5.9kFPS, 2.2 msec (2x U280)
RN50: 3.1kFPS, 1.7msec (1x U250)

©BOLEDU

Deep Network Intrusion Detection System (NIDS)



UNSW-NB15 dataset

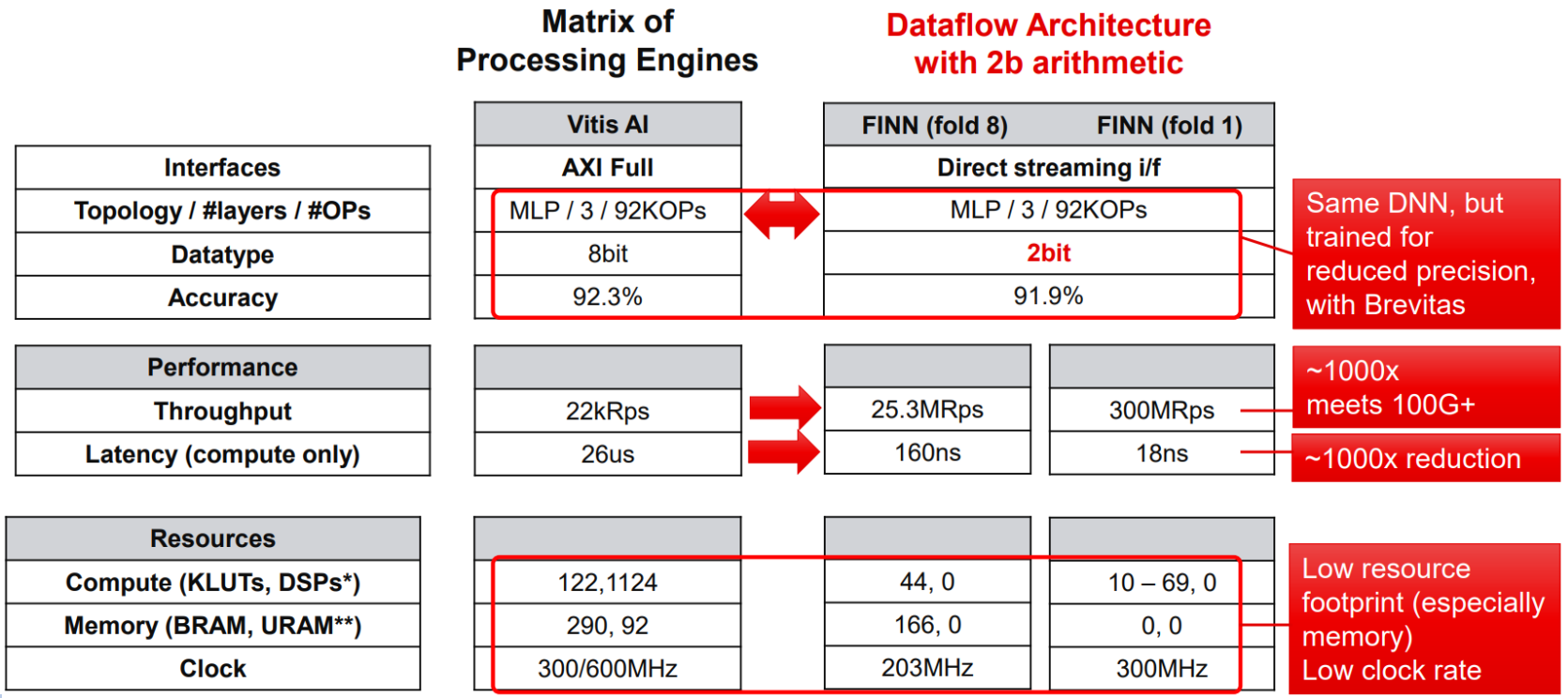
DNNs are increasingly popular:
- increased accuracy
- avoiding feature engineering

Throughput: 150MRps for 100G line rate
Latency sensitive (100Mb/msec)

MRps: Million requests per second
Assuming 64B / packet

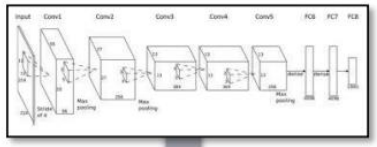
Deep Network Intrusion Detection System (NIDS) Results

- 1000x performance improvement over Vitis AI, less resources, 100Gbps line rate (150MRps)
- Through dataflow processing, reduced precision



©BOLEDU

FINN Framework: From DNN to FPGA Deployment



Brevitas
Training in pytorch
Algorithmic optimizations

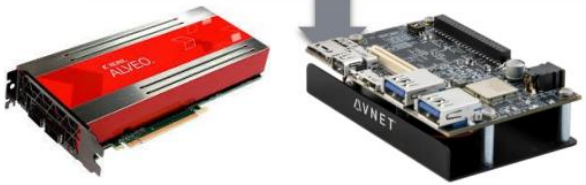
- Train or even learn reduced precision DNNs
- Library of standard layers

FINN compiler
Specializations of hardware architecture

- Perform optimizations
- Map to Vivado HLS
- Create DNN hardware IP

Deployment

- Works on embedded and Alveo platforms



Brevitas:
A PyTorch Library for Quantization-Aware Training

- Brevitas is a PyTorch research library for quantization-aware training (QAT).
- Export to ONNX
  - To import into the FINN compiler

Name	Input quantization	Weight quantization	Activation quantization	Dataset	Top1 accuracy
TFC_1W1A	1 bit	1 bit	1 bit	MNIST	93.17%
TFC_1W2A	2 bit	1 bit	2 bit	MNIST	94.79%
TFC_2W2A	2 bit	2 bit	2 bit	MNIST	96.60%
SFC_1W1A	1 bit	1 bit	1 bit	MNIST	97.81%
SFC_1W2A	2 bit	1 bit	2 bit	MNIST	98.31%
SFC_2W2A	2 bit	2 bit	2 bit	MNIST	98.66%
LFC_1W1A	1 bit	1 bit	1 bit	MNIST	98.88%
LFC_1W2A	2 bit	1 bit	2 bit	MNIST	98.99%
CNV_1W1A	8 bit	1 bit	1 bit	CIFAR10	84.22%
CNV_1W2A	8 bit	1 bit	2 bit	CIFAR10	87.80%
CNV_2W2A	8 bit	2 bit	2 bit	CIFAR10	89.03%

Name	First layer weights	Weights	Activations	Avg pool	Top1	Top5
MobileNet V1	8 bit	4 bit	4 bit	4 bit	71.14	90.10
ProxylessNAS Mobile14 w/ Hadamard classifier	8 bit	4 bit	4 bit	4 bit	73.52	91.46
ProxylessNAS Mobile14	8 bit	4 bit	4 bit	4 bit	74.42	92.04
ProxylessNAS Mobile14	8 bit	4 bit, 5 bit	4 bit, 5 bit	4 bit	75.01	92.3

Open Neural Network Exchange (ONNX)

- ONNX provides an open source format for AI models, both deep learning and traditional ML.
- ONNX is widely supported and can be found in many frameworks, tools, and hardware.
- Enabling interoperability between different frameworks and streamlining the path from research to production helps increase the speed of innovation in the AI community.

FINN Compiler

- Transform DNN into Custom Dataflow Architecture

Input is ONNX description of the quantized DNN

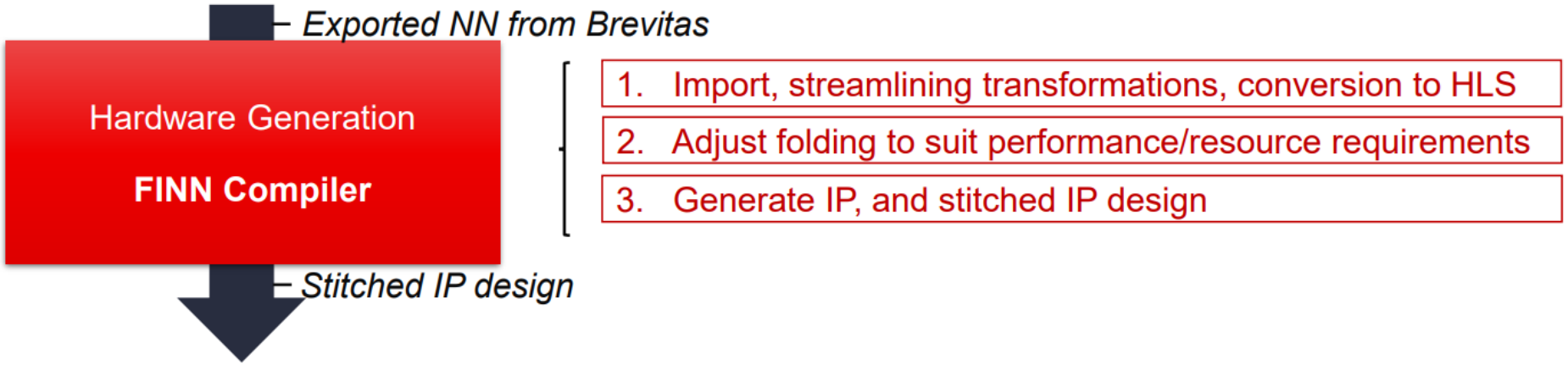
Output is the stitched DNN accelerator IP

FINN Compiler

- FINN uses the ONNX-based intermediate representation as intermediate representation (IR)
- FINN is a python library of graph transformations
- Synthesizable description of each layer is produced (in HLS)
- After synthesis each layer as IP block
  - AXI stream inputs and outputs

FINN Compiler for Hardware Generation

- In 3 Steps



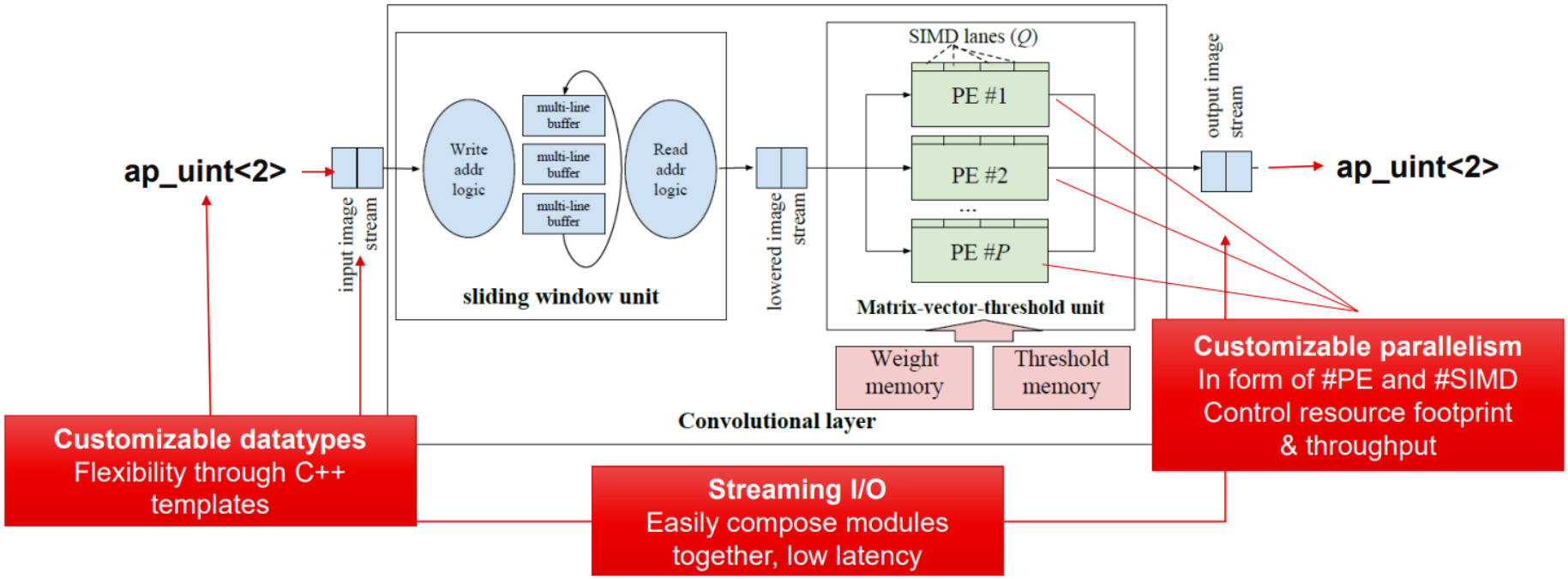
FINN Compiler: HLS Generation

- Generate calls to a pre-optimized Vivado HLS C++ library
- Support arbitrary-precision datatypes via templates
- Synthesizable to RTL


```
hls::stream<ap_int<185>> in
hls::stream<ap_int<100>> inter0, inter1, ...
...
StreamingFCLayer<BINARY, BINARY, ..>(in, inter0, ...)
StreamingFCLayer<BINARY, BINARY, ..>(inter0, inter1, ..)
...
```

The FINN HLS Library

- Key component: MVTU (Matrix Vector Threshold Unit)



The FINN HLS Library

- An optimized, templated Vivado HLS C++ library of 10+ common DNN layers

activations.hpp	Fixed documentation for thresholding blocks	4 months ago
bnn-library.h	Add UpsampleNearest for square IFM	6 months ago
convlayer.h	correct convlayer	7 months ago
dma.h	Change PE streaming weights Endianess from ...	2 years ago
fclayer.h	Fixed bug on VVAU and small cosmetic changes	2 years ago
gen-python-data.sh	refactor: pregen data, extract from Jenkinsfile i...	12 days ago
interpret.hpp	Fixed bug in slice and slice_mmv	2 years ago
mac.hpp	Added MMV support in MVAU and convolutio...	2 years ago
maxpool.h	Add tb to Jenkinsfile	6 months ago
mmv.hpp	Added MMV support in MVAU and convolutio...	2 years ago
mvau.hpp	Added explicit unroll on PE loop	17 months ago
pool.hpp	Add QuantAvgPoolFunction to implement Aver...	17 months ago
requirements.txt	Merge pull request #62 from Xilinx/dependab...	12 days ago
slidingwindow.h	Rename, add documentation	last month
streamtools.h	Added support for non-square images padding	14 months ago

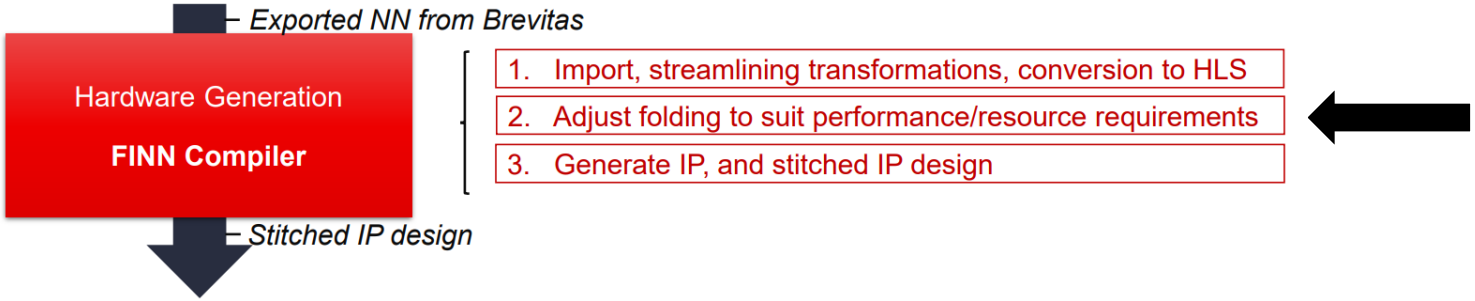
GitHub

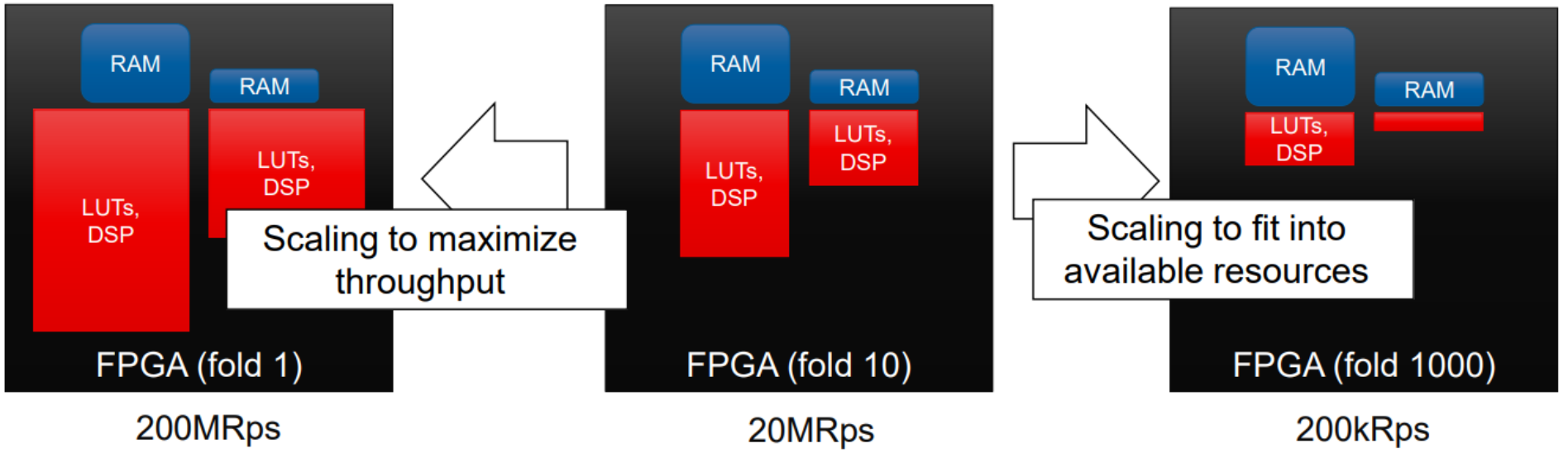
More details next week!

https://github.com/Xilinx/finn-hlslib

©BOLEDU

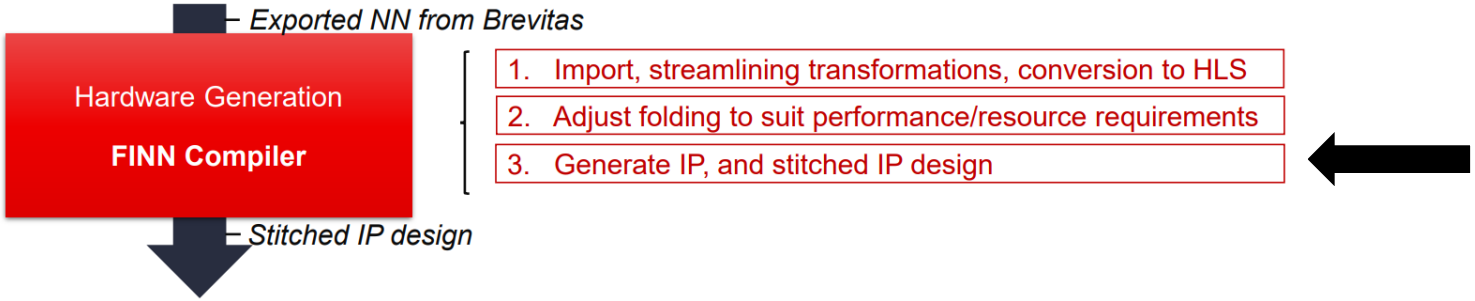
FINN Compiler: Adjusting Performance/Resources



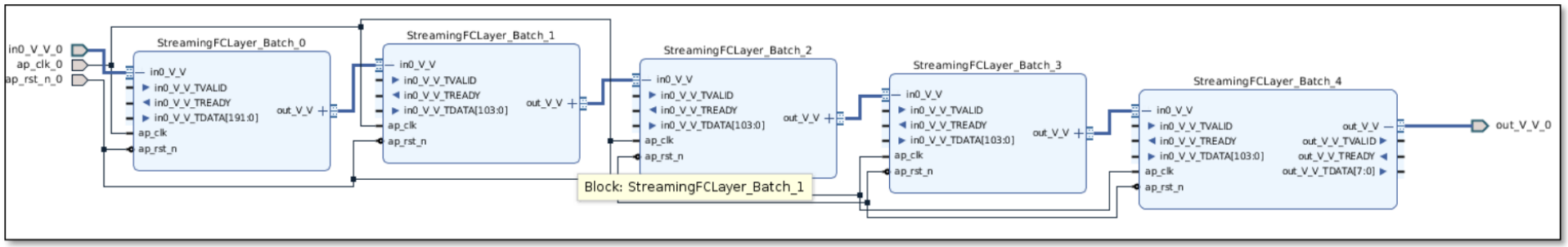


200MRps 20MRps 200kRps

FINN Compiler: IP Generation Flow



- Stream-in, stream-out FPGA IP block
  - Easy "bump-in-the-wire" integration into streaming systems
  - Simple data movement, fully deterministic



Overview of the FINN software stack

©BOLEDU

FINN-Examples: Prebuilt Dataflow Accelerators

- Dataflow accelerators for MNIST, CIFAR-10, ImageNet
  - Bitfiles for PYNQ boards and Alveo U250
- Jupyter notebook example to run each accelerator
  - Based on PYNQ Python driver

```
# on your PYNQ board or Alveo U250
pip3 install finn-examples
pynq get-notebooks --from-package finn-examples -p .
```

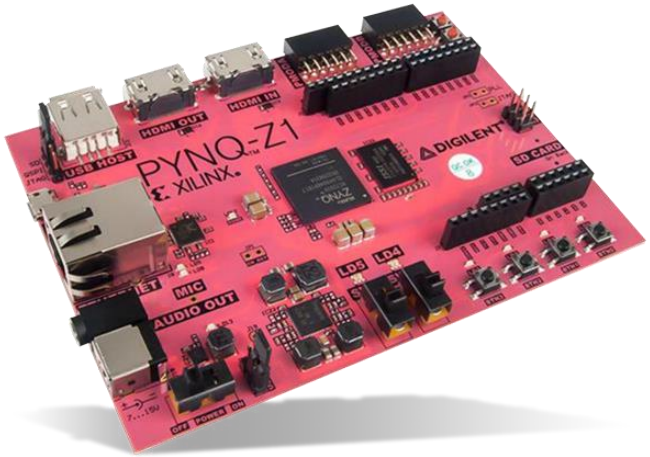
- More examples
  - ResNet-50 toolflow (bitfiles already on Xilinx/ResNet50-PYNQ)
  - speech recognition & keyword spotting

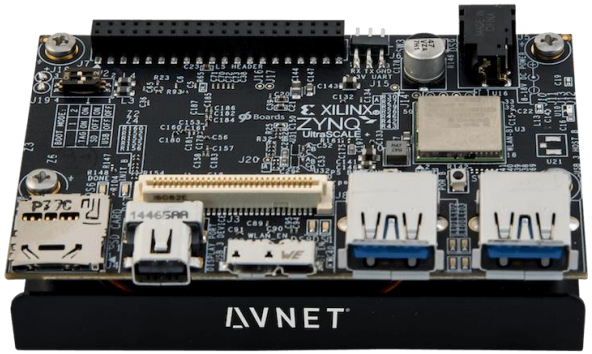
GitHub https://github.com/Xilinx/finn-examples

©BOLEDU

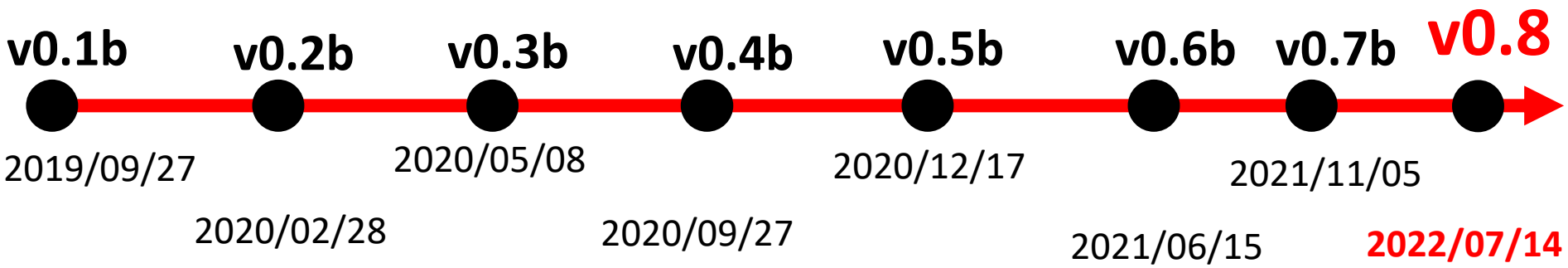
Supported Boards

- Edge: Pynq-Z1, Pynq-Z2, Ultra96 and ZCU104
- Datacenter: Alveo U250



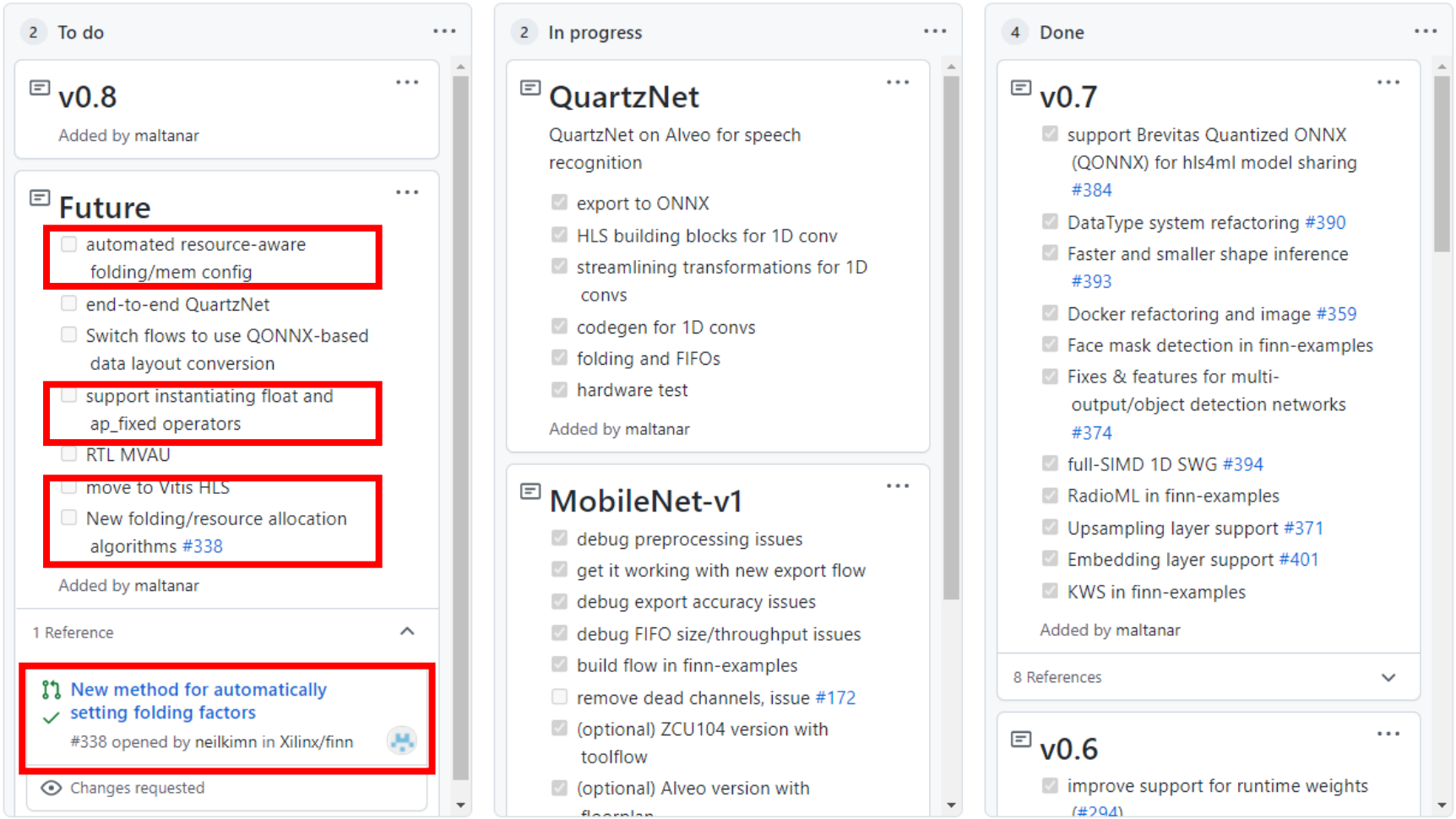


FINN Release Verion



https://xilinx.github.io/finn/blog
https://github.com/Xilinx/finn/discussions/categories/announcements

FINN Road Map

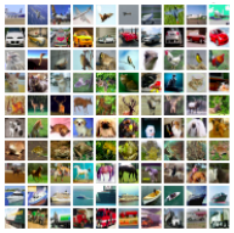


FINN Application Example

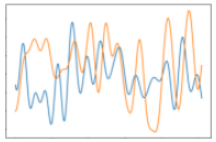
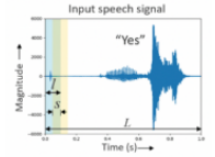
- MNIST Handwritten Digit Classification
- Image Classification using CIFAR-10dataset
- Image Classification using ImageNet dataset
- Face mask wear and positioning
  - Low-power BNN classifier for Pynq-Z1 for correct face mask wear and positioning
- Radio signal modulation
  - Classify RadioML 2018.1 at 250k inferences/second on a ZCU104
- Keyword spotting
  - Trained on the Google Speech Commands v2 dataset

https://github.com/Xilinx/finn-examples/tree/main/finn_examples/notebooks

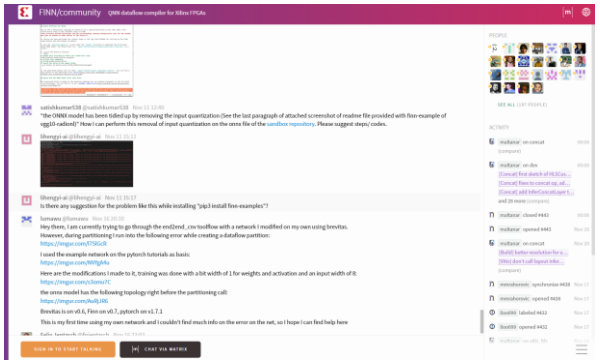
Example Neural Network Accelerators(1/2)

Dataset	Topology	Quantization	Supported boards
 CIFAR-10	CNV (VGG-11-like)	several variants: 1/2-bit weights/activations	all
 MNIST	3-layer fully-connected	several variants: 1/2-bit weights/activations	all
 ImageNet	MobileNet-v1	4-bit weights and activations 8-bit first layer weights	Alveo U250 ZCU104
 ImageNet	ResNet-50	1-bit weights 2-bit activations 4-bit residuals 8-bit first/last layer weights	Alveo U250

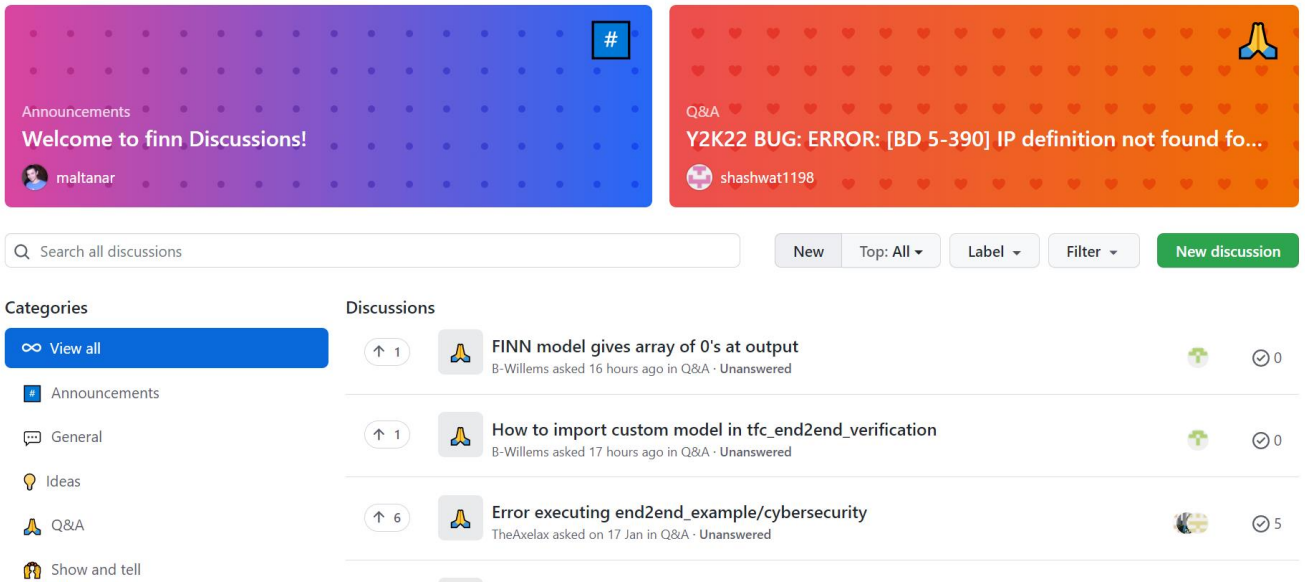
Example Neural Network Accelerators(2/2)

 ImageNet	ResNet-50	1-bit weights 2-bit activations 4-bit residuals 8-bit first/last layer weights	Alveo U250
 RadioML 2018	1D CNN (VGG10)	4-bit weights and activations	ZCU104
 MaskedFace-Net	BinaryCoP *Contributed by TU Munich+BMW*	1-bit weights and activations	Pynq-Z1
 Google Speech Commands v2	3-layer fully-connected	3-bit weights and activations	Pynq-Z1

FINN Community



- Old community :
  - **https://gitter.im/xilinx-finn/community**
- New community (Current) :
  - https://github.com/Xilinx/finn/discussions



FINN Resource

- Xilinx Official Github Pages: https://xilinx.github.io/finn/
- Xilinx Official Documents: https://finn.readthedocs.io/en/latest/
- Xilinx FINN Repositories:
  - FINN framework: https://github.com/Xilinx/finn
  - finn-base: https://github.com/Xilinx/finn-base
  - finn-hls: https://github.com/Xilinx/finn-hlslib
- Xilinx Brevitas: https://github.com/Xilinx/brevitas
- Xilinx FINN Examples:
  - FINN Examples: https://github.com/Xilinx/finn-examples
  - BNN-PYNQ: https://github.com/Xilinx/BNN-PYNQ
- Xilinx FINN Publications: https://xilinx.github.io/finn/publications
  - FINN: A Framework for Fast, Scalable Binarized Neural Network Inference
  - FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks
- Xilinx Vitis HLS Pragmas: https://www.xilinx.com/html_docs/xilinx2021_1/vitis_doc/hls_pragmas.html

FINN Textbook

- Chapter 1: Getting Started

In this chapter, we will show how to take a simple, binarized, fully-connected network trained on the MNIST data set and take it all the way down to a customized bitfile running on a PYNQ board.

- Chapter 2: Network Define

In this chapter, we are going to introduce how to use Brevitas which is a PyTorch research library for quantization-aware training (QAT) to define the network and do the quantization-aware training.

- Chapter 3: Compiler

In chapter 3, we are going to take a deeper look at the FINN compiler part. Readers can safely jump over this chapter if the compiler part is not of your interest. We will give some guides to anyone interested in adding custom hardware operations into FINN compiler. Note that we assume readers have already gone through Chapter 1 and Chapter 2.

- Chapter 4: Verification

In chapter 4, we are going to talk about design verifications. This chapter contains three sections. The first section talks more about how verification flow is executed within FINN compiler. The second section is the simulation for C/C++ executed code. The third section is rtl simulation, performing cycle accurate tests and verifies the final hardware HDL implementation.

- Chapter 5: NN Hardware

In this chapter, we are going to explain the binarized neural network hardware part. This is based on the Xilinx paper "FINN: A Framework for Fast, Scalable, Binarized Neural Network Inference."

- Chapter 6: HLS

In chapter 6, we explain the detailed HLS source code of the hardware library. Here, readers should be familiar to the FINN hardware architexture explained in chapter 5 as well as High-Level-Synthesis.

- Chapter 7: Case Study

In chapter 7, we go through a case study of end-to-end VGG9 neuaral network from network define, training, to the deployment on Pynq-Z2 edge device.

Outline

- Introduction to FINN
- Network Define
- NN Hardware
- Lab Description