

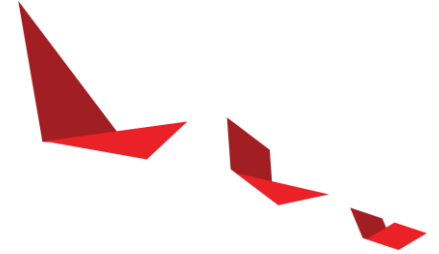


Tutorial: Neural Network Accelerator Co-Design with **FINN**

Xilinx Research Labs, TU Delft, Heidelberg University

2021-03-24

Tutorial Agenda



- ▶ Introduction to FINN (~30m)
- ▶ A tour of the repositories and the flow (~10m)
- ▶ FINN community update (~15m)

----- Break + Q&A (~10m)

- ▶ Hands-on part (~1.5h)
 - Training a quantized MLP with Brevitas
 - Exporting the MLP and verification
 - From MLP to custom hardware with the FINN compiler



Part of recording from the FINN FPGA'21 tutorial
<https://www.youtube.com/watch?v=zw2aG4PhzmA>



Introduction to FINN

Michaela Blott
Distinguished Engineer, Xilinx Research Labs

A Tour of the Repositories

Yaman Umuroglu
Senior Research Scientist, Xilinx Research Labs

Introduction to FINN

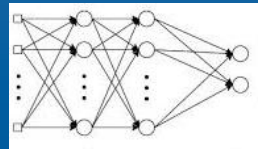
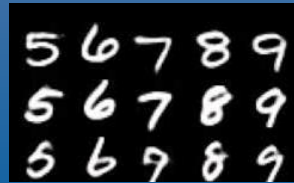
*Michaela Blott
Distinguished Engineer, Xilinx Research Labs*

FINN: The Beginning (FPGA'17)

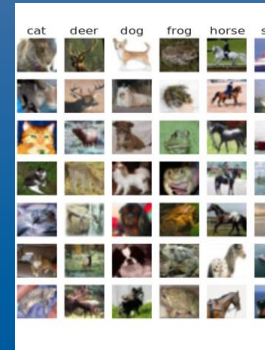
FINN: A Framework for Fast, Scalable Binarized Neural Network Inference

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



MNIST MLP
12.3 Million FPS
310 ns latency



CIFAR-10 ConvNet
21.9 kFPS
283 us latency

FINN – Project Mission



▶ Mission

- Tools and platforms for creation of high throughput, ultra-low latency DNN compute architectures

▶ End-to-end flow

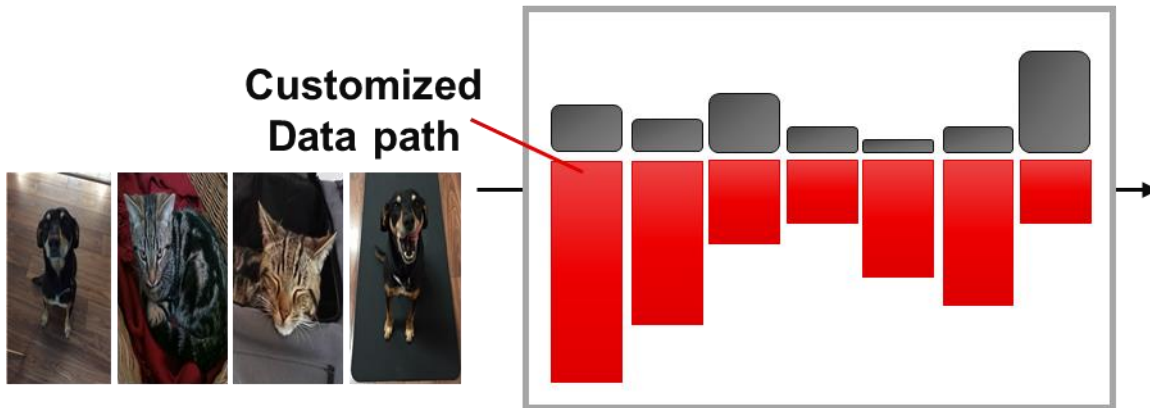
- Users can easily create specialized hardware architectures on an FPGA and benefit from custom architectures and custom precision

▶ Open source

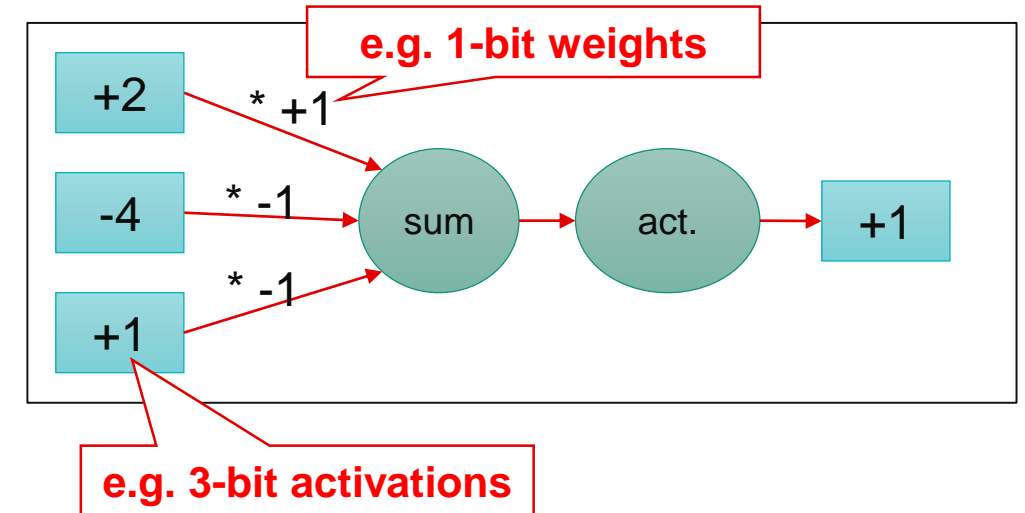
- Transparency and flexibility to adapt to end-users' applications

Two Key Techniques for Customization in FINN

Streaming Dataflow Architectures with FPGAs & FINN



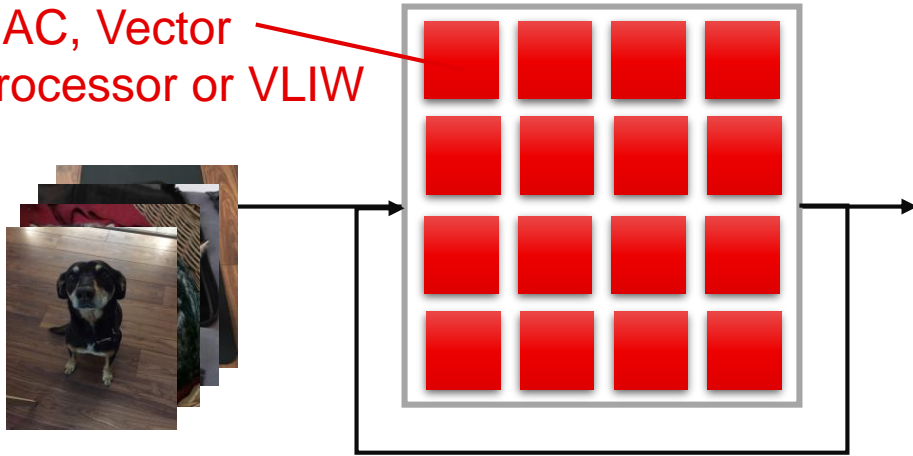
Custom Precision Few-bit weights & activations



Customized Dataflow Processing versus More Generic Architectures

Matrix of Processing Engines (MPE) (Vitis AI, ASICs, GPUs):

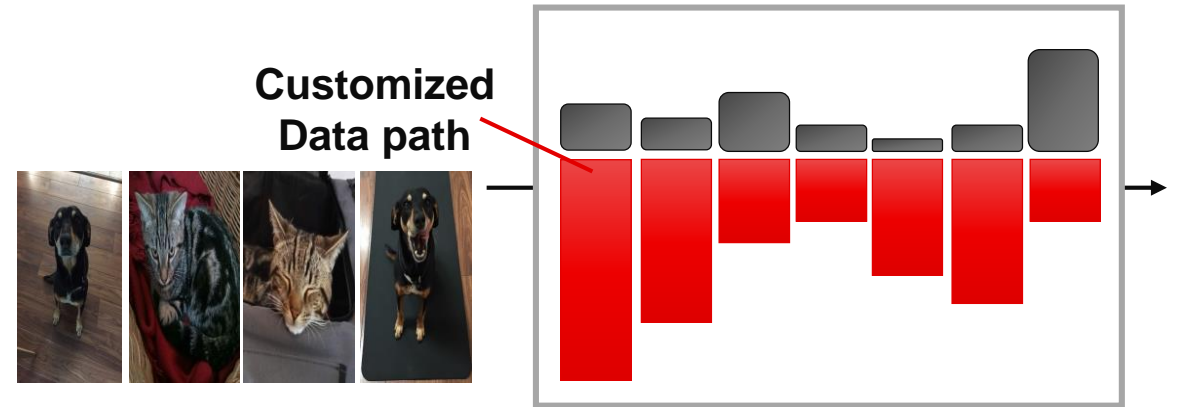
MAC, Vector
Processor or VLIW



- Customized for typical DNN operations
 - for example multiply accumulate
- Lower throughput (~10KRps)
- Flexibility for ASICs
- Applications: CV, Speech

Dataflow Architectures with FPGAs & FINN

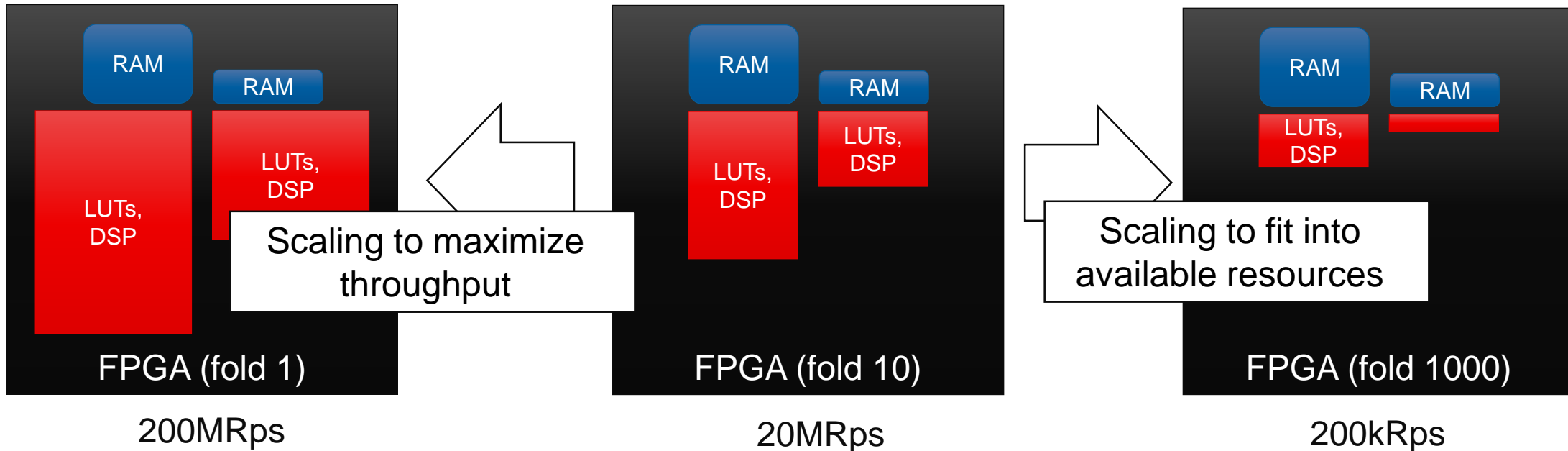
Customized
Data path



- Customized/adapt for specific DNN topologies
- Streaming interfaces
- Specialization -> higher efficiency
- Lower latency (no intermediate buffering)
- Higher throughput (~100MRps)
- Flexibility through reconfiguration
- Applications: TBD, networking, material science, particle physics – smaller DNNs

Dataflow Processing:

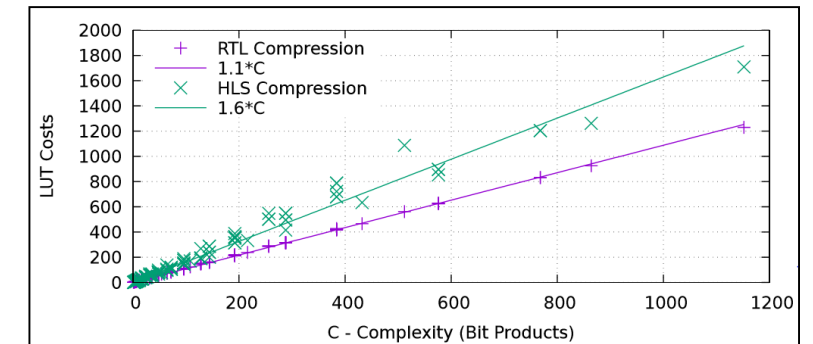
Scaling to Meet Performance & Resource Requirements



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

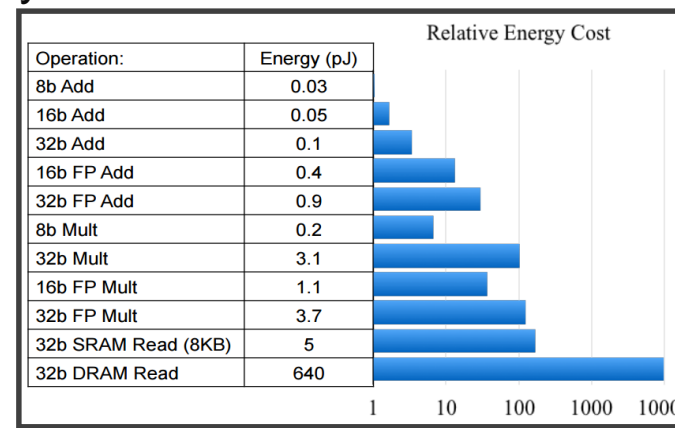
Customizing Arithmetic to Minimum Precision Required

- ▶ 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

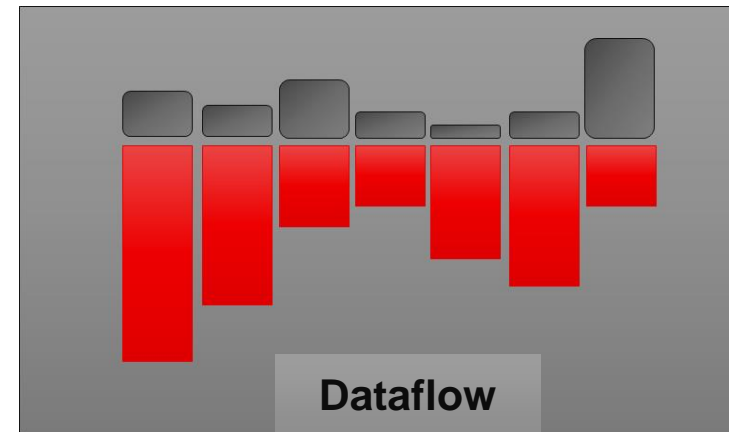
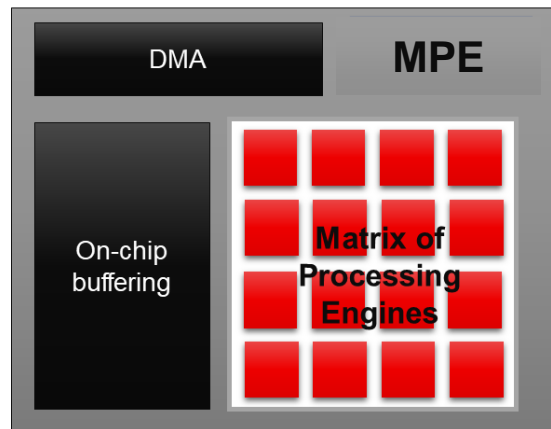
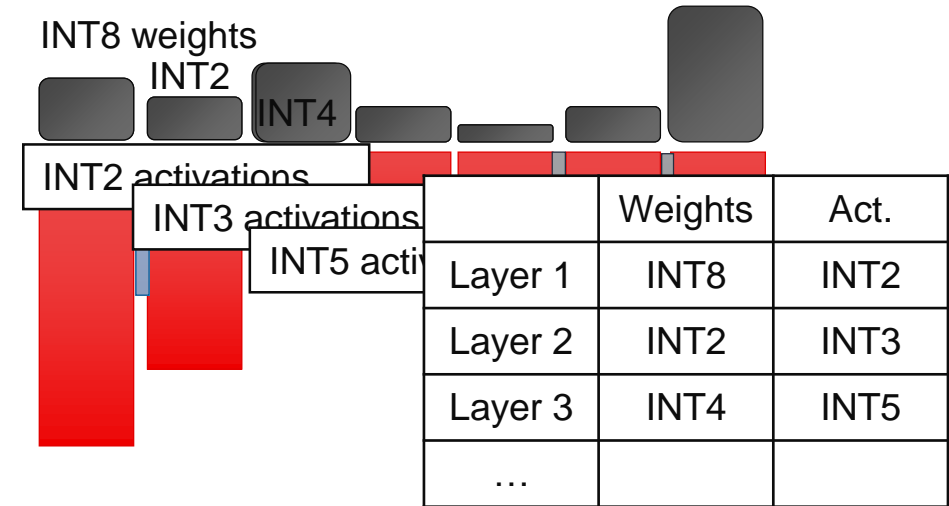
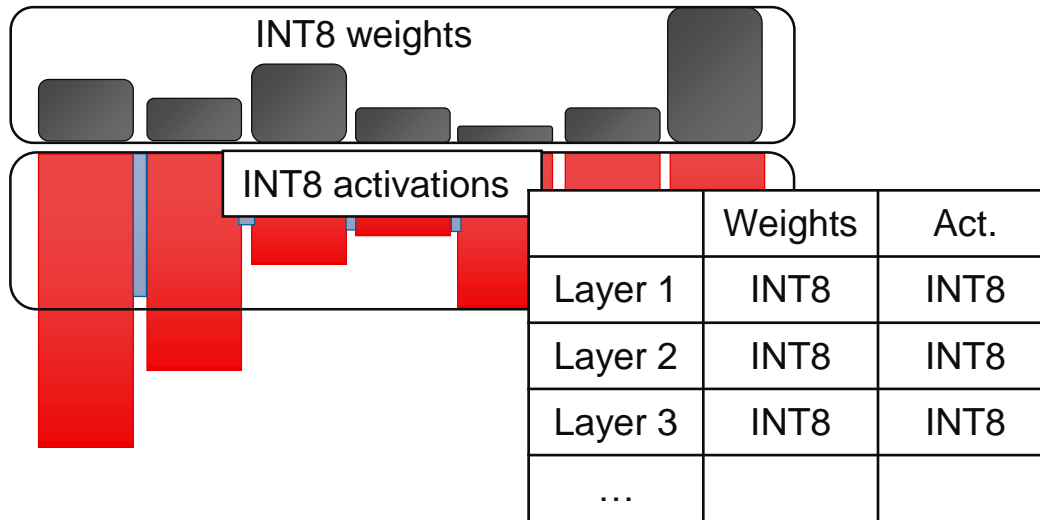


$C = \text{size of accumulator} * \text{size of weight} * \text{size of activation}$

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



Granularity of Customizing Arithmetic

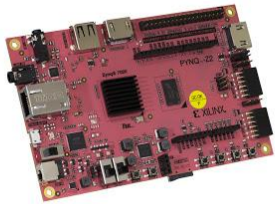
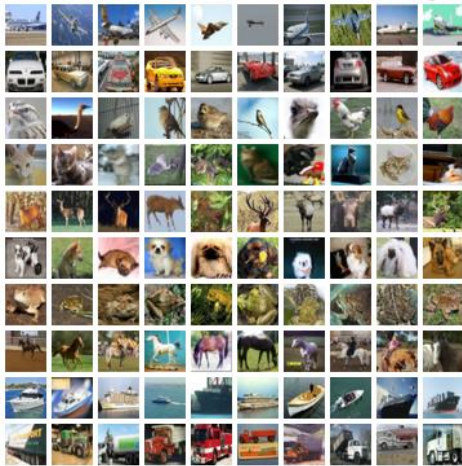


Dataflow architectures can exploit custom arithmetic at a greater degree

Results

Few-bit DNNs + FPGA Dataflow: Showcases

Low-Power, Real-Time Image Classification



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

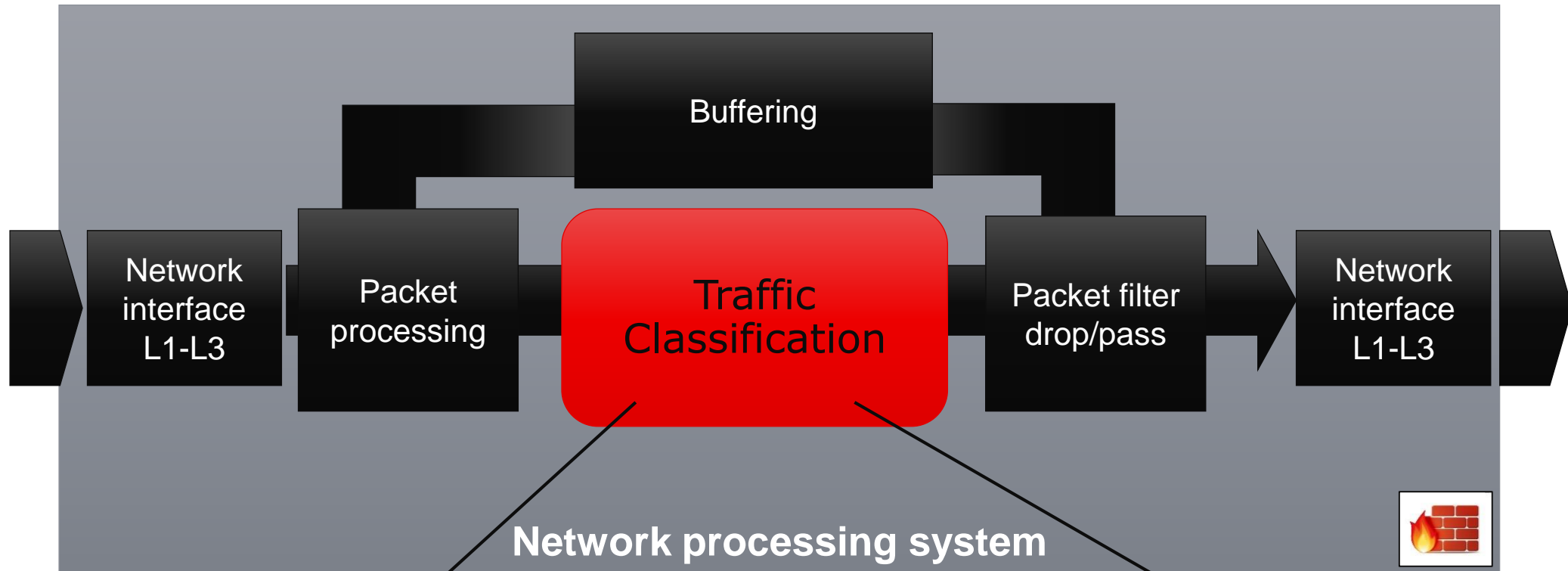
Single and multi-node ImageNet Classification (on XACC)

We'll tell you more
about this later...



ResNet-50, MobileNet on
Alveo U280 & U250
MNv1: 5.9kFPS, 2.2 msec
(2x U280)
RN50: 3.1kFPS, 1.7msec
(1x U250)

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

NIDS Results

Matrix of Processing Engines

Dataflow Architecture with 2b arithmetic

Interfaces
Topology / #layers / #OPs
Datatype
Accuracy

Vitis AI
AXI Full
MLP / 3 / 92KOPs
8bit
92.3%



FINN (fold 8)	FINN (fold 1)
Direct streaming i/f	
MLP / 3 / 92KOPs	
2bit	
91.9%	

Same DNN, but trained for reduced precision, with Brevitas

Performance
Throughput
Latency (compute only)

22kRps
26us



25.3MRps
160ns

300MRps
18ns

~1000x meets 100G+

~1000x reduction

Resources
Compute (KLUTs, DSPs*)
Memory (BRAM, URAM**)
Clock

122,1124
290, 92
300/600MHz

44, 0
166, 0
203MHz

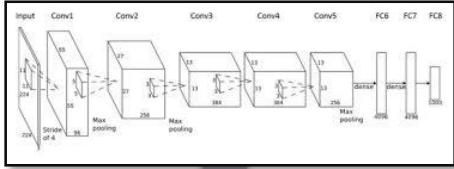
10 – 69, 0
0, 0
300MHz

Low resource footprint (especially memory)
Low clock rate

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

The FINN Framework

FINN Framework: From DNN to FPGA Deployment



Brevitas
Training in pytorch
Algorithmic optimizations

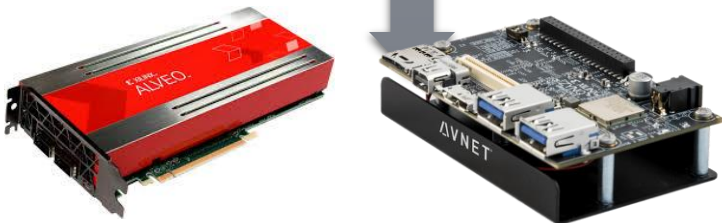
- Train or even learn reduced precision DNNs
- Library of standard layers
- Pretrained examples

FINN compiler
Specializations of
hardware architecture

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

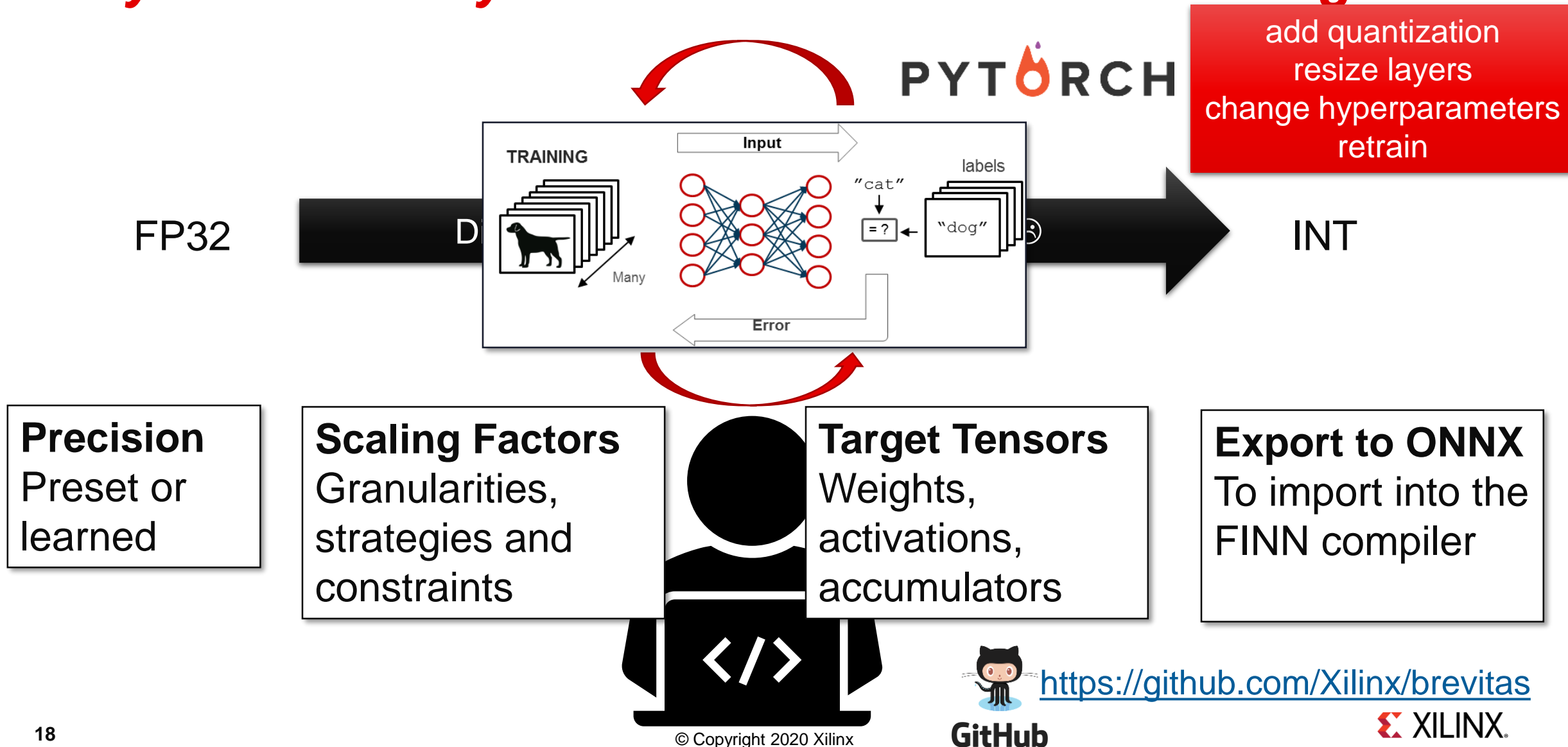
Deployment

- Embeds the DNN IP into an infrastructure design
- Generates Python run-time
- Enables integration with your application
- Works on embedded and Alveo platforms
 - Including XACC



Brevitas:

A PyTorch Library for Quantization-Aware Training



FINN Compiler

Transform DNN into Custom Dataflow Architecture

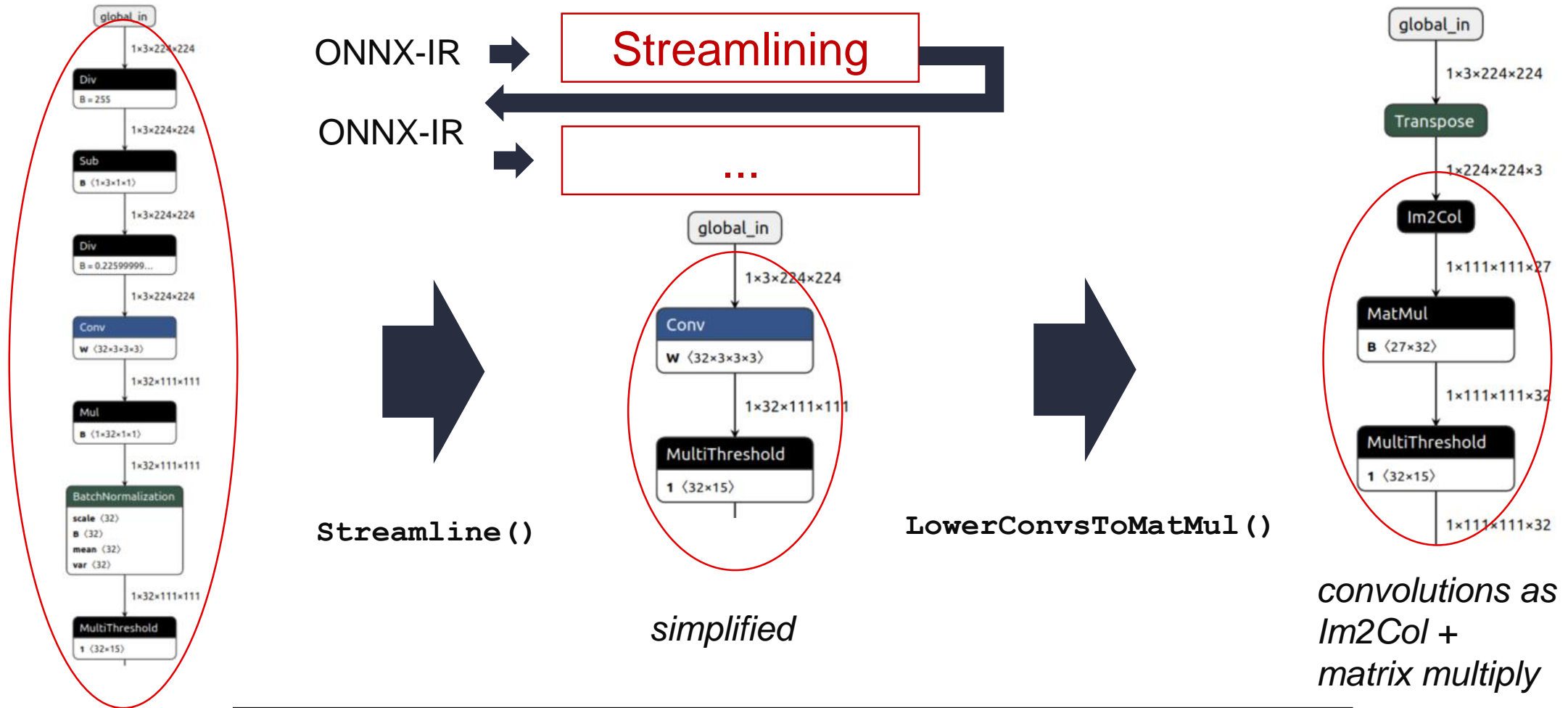
Input is ONNX description of the quantized DNN

- ▶ 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

Output is the stitched DNN accelerator IP

FINN Flows

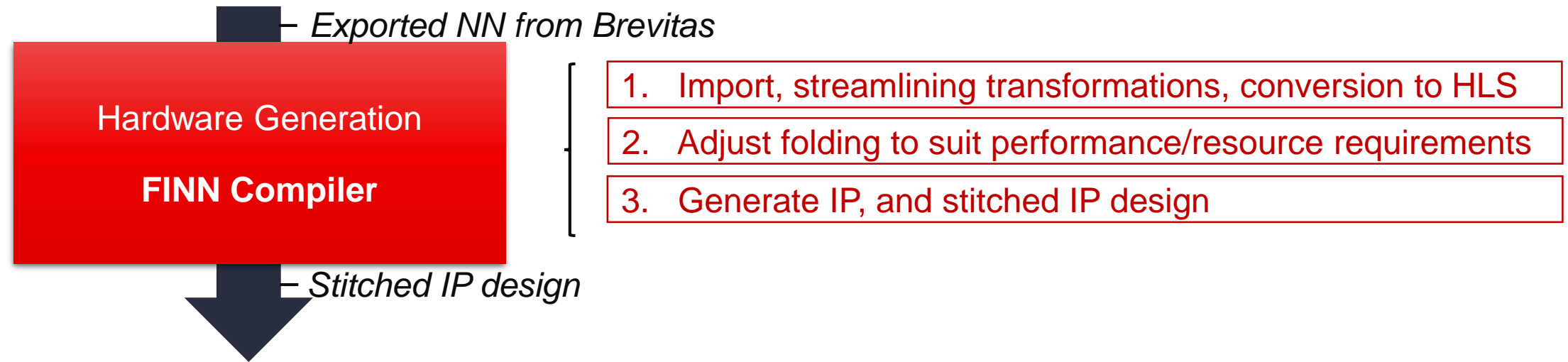
Every Step is a ONNX Graph Transformations



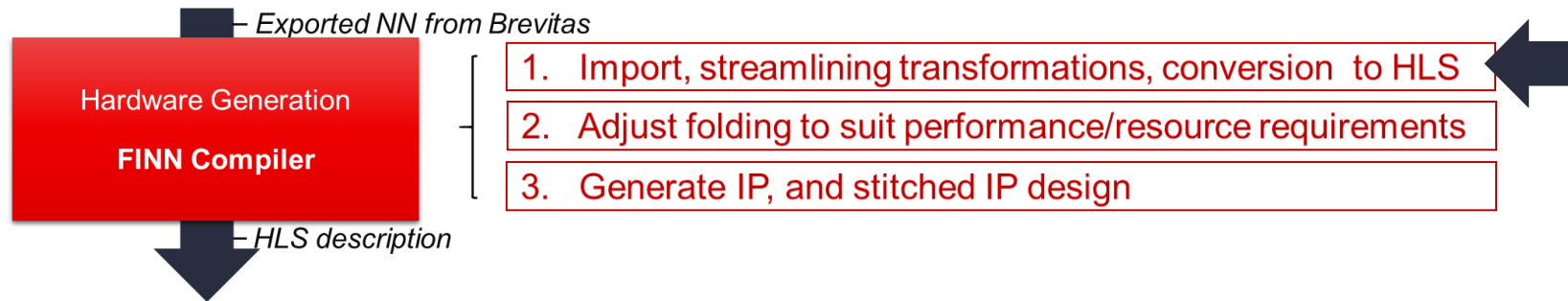
Optimization, lowering, code generation... are all transformations

FINN Compiler for Hardware Generation

In 3 Steps



FINN Compiler: Import, Optimization & HLS Generation



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



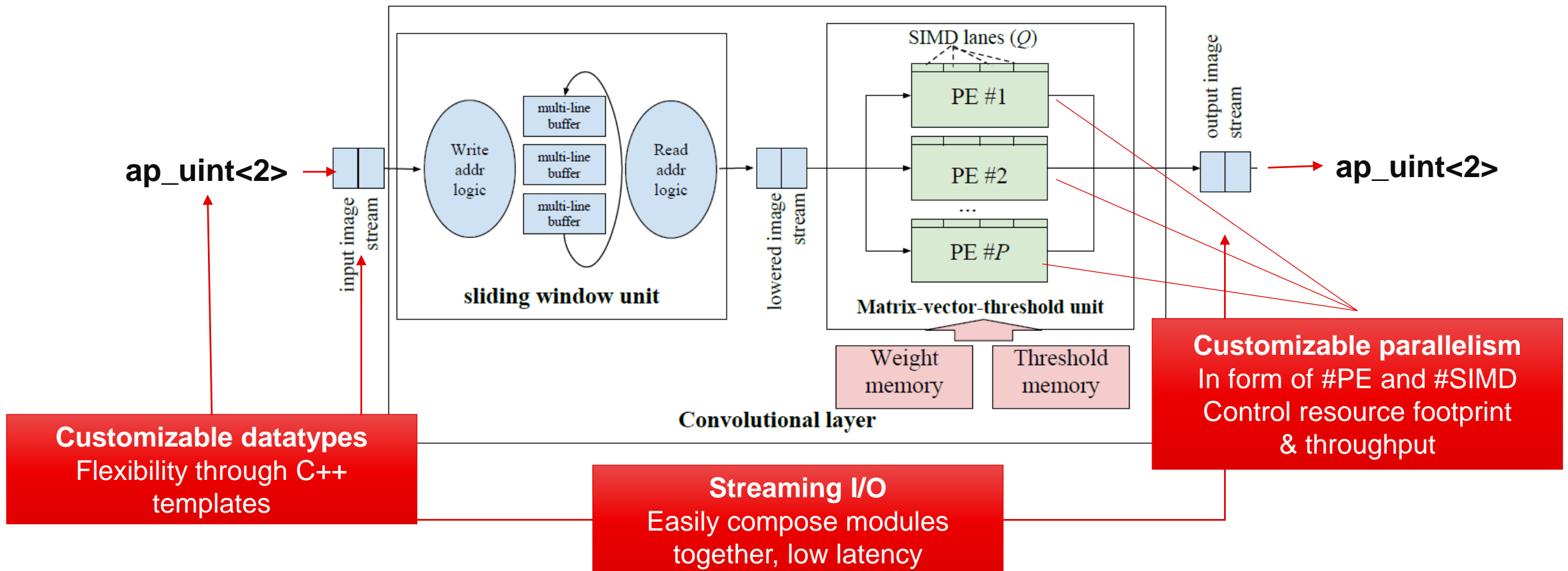
- ▶ Generate calls to a pre-optimized Vivado HLS C++ library
- ▶ Support arbitrary-precision datatypes via templates
- ▶ Synthesizable to RTL

The FINN HLS Library

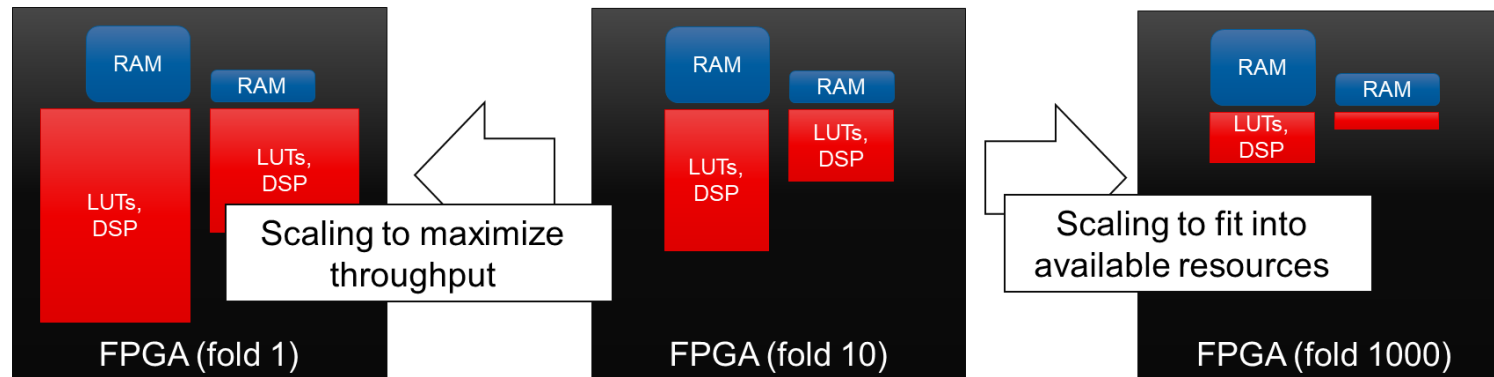
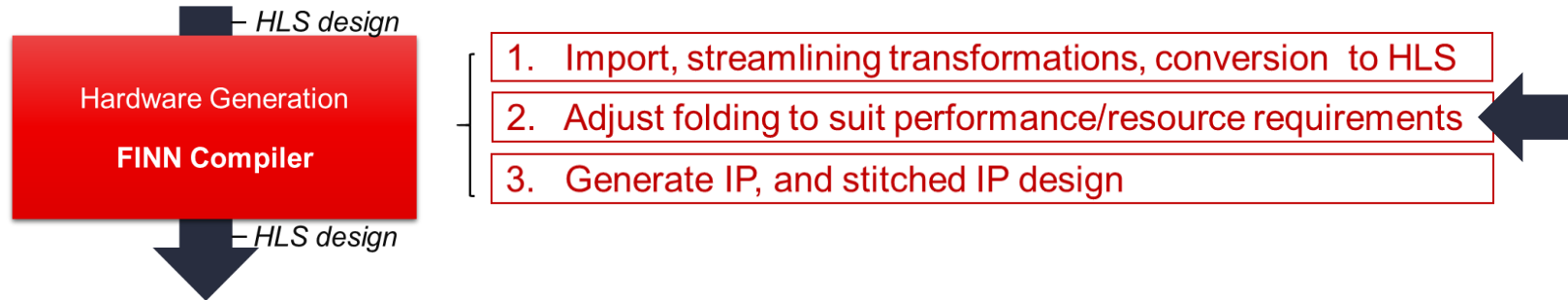


<https://github.com/Xilinx/finn-hlslib>

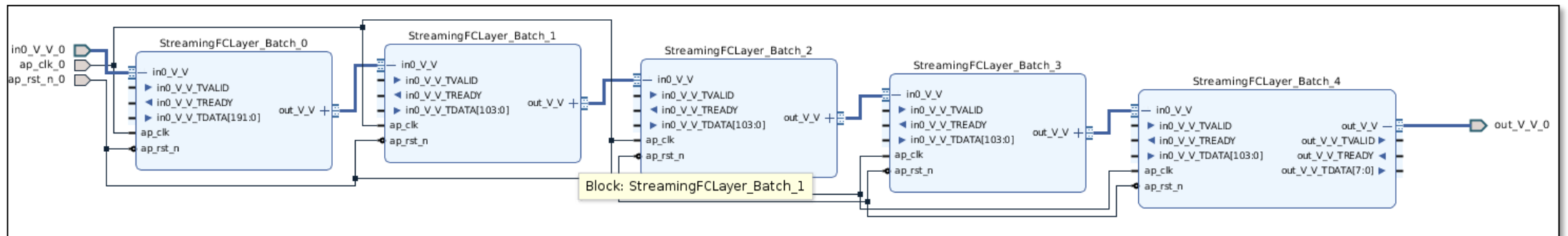
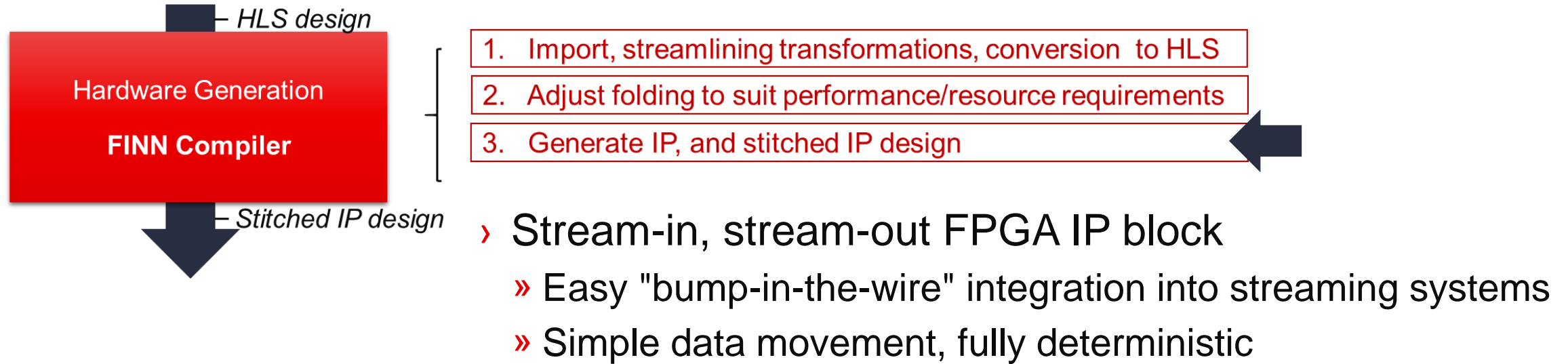
- › An optimized, templated Vivado HLS C++ library of 10+ common DNN layers
- › Key component: MVTU (Matrix Vector Threshold Unit)



FINN Compiler: Adjusting Performance/Resources

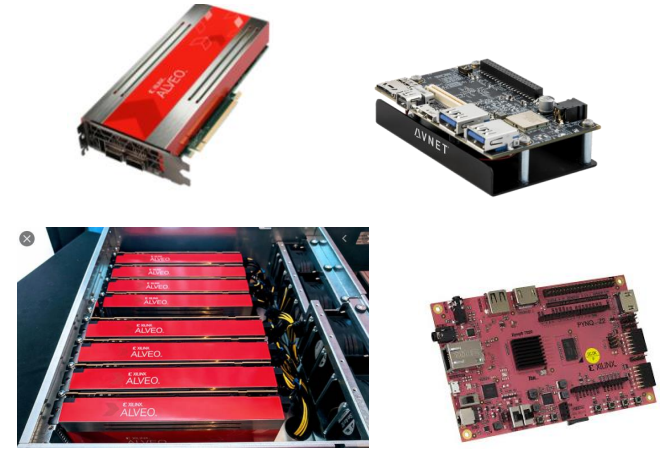


FINN Compiler: IP Generation Flow



Deployment with PYNQ™ for Python Productivity

```
# instantiate the accelerator
accel = models.cnv_w2a2_cifar10()
# generate an empty numpy array to use as input
dummy_in = np.empty(accel.ishape_normal,
dtype=np.uint8)
# perform inference and get output
dummy_out = accel.execute(dummy_in)
```



- ▶ Use PYNQ-provided Python abstractions and drivers
- ▶ User provides Numpy array in, calls driver, gets Numpy array out
 - Internally use PYNQ DMA driver to wr/rd NumPy arrays into I/O streams



GitHub

<https://github.com/Xilinx/PYNQ>

<https://github.com/Xilinx/finn-examples>

Infrastructure for Experimentation & Collaboration

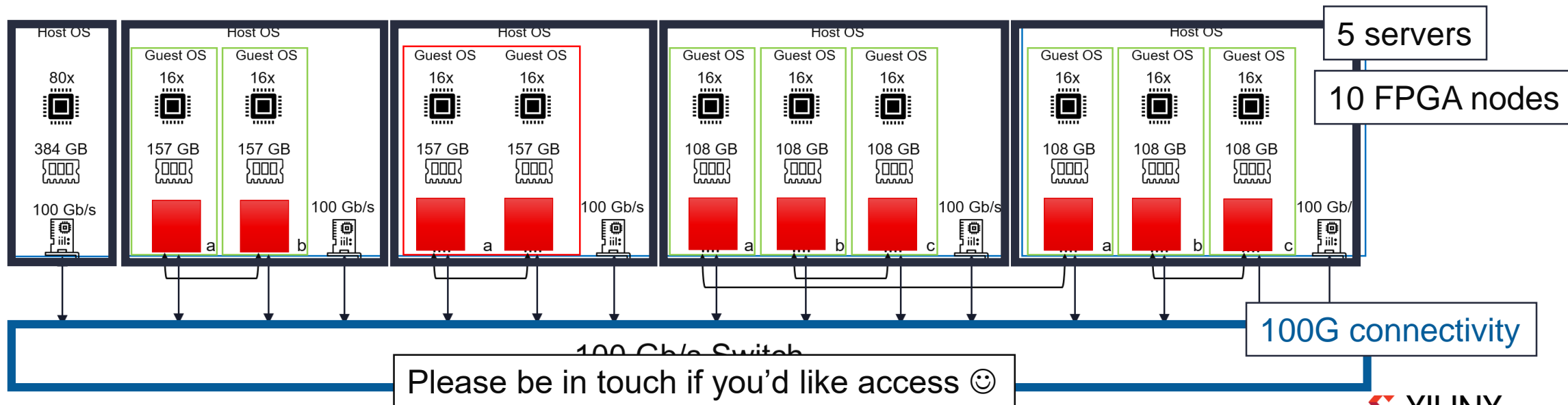
► Xilinx academic compute clusters (XACC)

- 4 centres world-wide
- Free to use
- Enabling research community



► Flexibility, shared hardware, networked FPGAs

► Not only for FINN



FINN Status

- ▶ **Many example designs available at [github/finn](https://github.com/finn)**
 - Increasing application & feature & platform support
- ▶ **Ongoing development** (3 researchers + community)
- ▶ **Training material**
 - Tutorials (more coming!)
 - University classes with FINN @ Stanford, Charlotte, NTNU
 - EPFL and Technion in preparation
- ▶ **Looking to build-up community, applications and functionality**



**If you're interested in collaborating,
please be in touch 😊**



A Tour of the Repositories

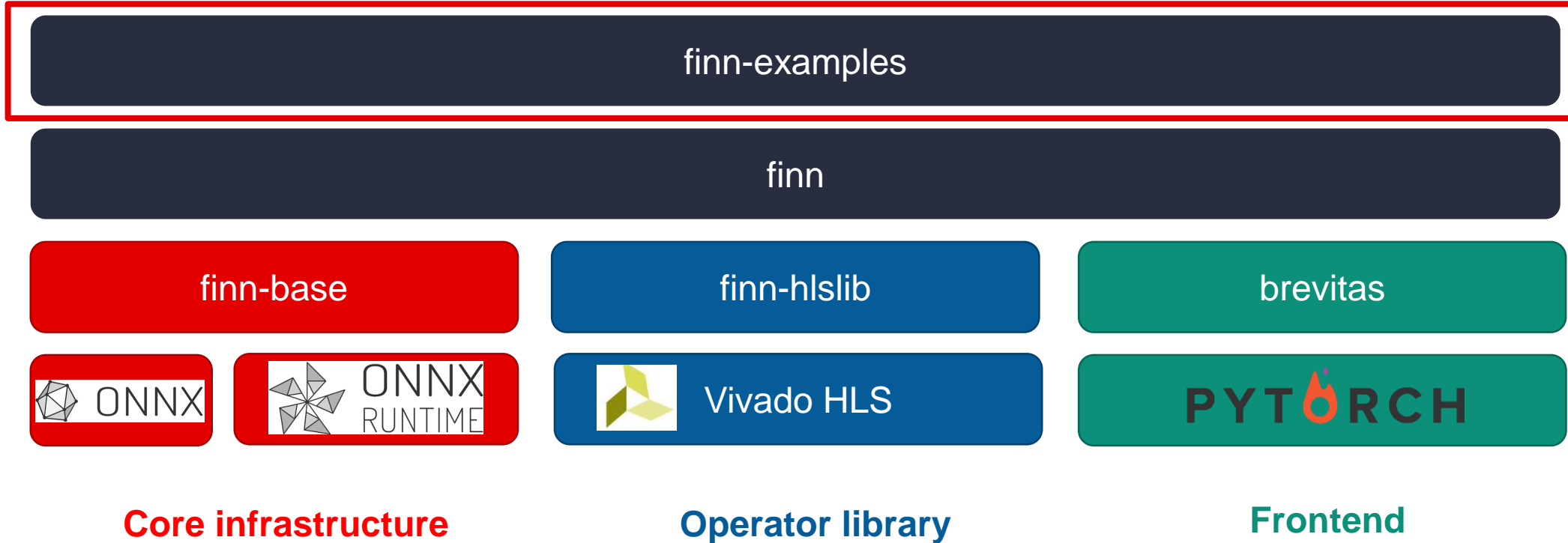
Yaman Umuroglu
Senior Research Scientist, Xilinx Research Labs

Overview of the FINN software stack



 project landing page: <https://xilinx.github.io/finn>

Tour of the FINN software stack



✓ **FINN** project landing page: <https://xilinx.github.io/finn>

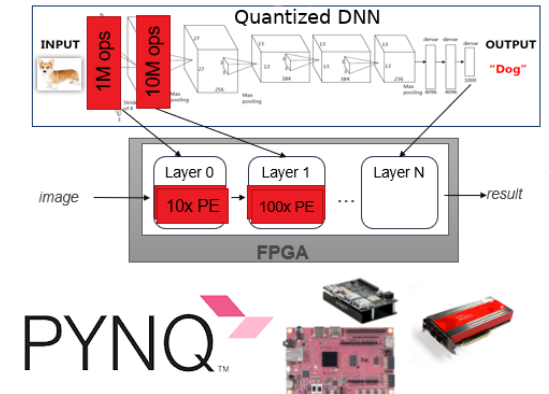
finn-examples: prebuilt dataflow accelerators

- ▶ 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

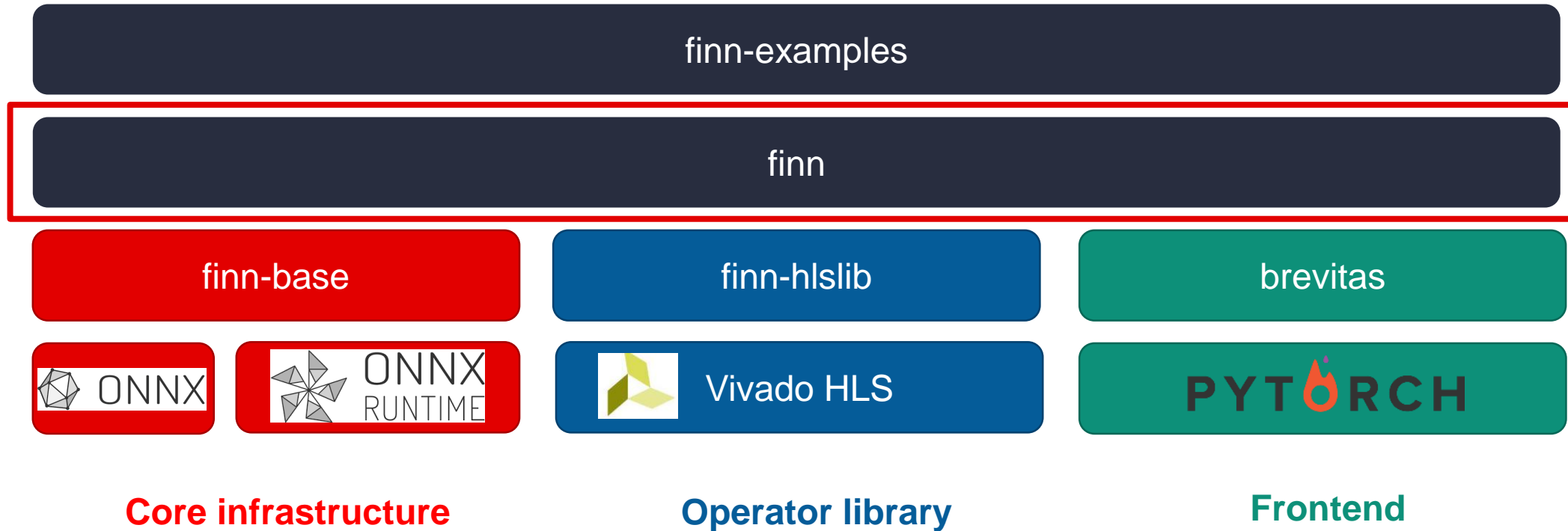


```
# on your PYNQ board or Alveo U250
pip3 install finn-examples
pynq get-notebooks --from-package finn-examples -p .
```

- ▶ Scripts to rebuild the examples
- ▶ More examples on the way
 - ResNet-50 toolflow (bitfiles already on Xilinx/ResNet50-PYNQ)
 - speech recognition & keyword spotting
 - <your cool dataflow accelerator example here!>



Tour of the FINN software stack



 project landing page: <https://xilinx.github.io/finn>

finn: dataflow compiler

- ▶ ONNX -> bitfile (or IPI design) build automation
- ▶ Docker environment with all dependencies
- ▶ Large library of graph transformations
 - Streamlining to remove floating point scaling factors
 - Lowering to finn-hlslib ops
 - Stitching generated IPs in Vivado IPI
- ▶ Custom ops corresponding to finn-hlslib
 - Including code generation and verification
- ▶ Jupyter notebook tutorials (basic, advanced, end2end)



Xilinx/finn



Read the Docs

finn.readthedocs.io



Tour of the FINN software stack



 project landing page: <https://xilinx.github.io/finn>

brevitas: quantization-aware training in PyTorch

- ▶ Train NNs with quantized weights and activations
- ▶ Quantized versions of many PyTorch layers
 - e.g. `brevitas.nn.QuantConv2D` instead of `torch.nn.Conv2D`
- ▶ Flexible quantization schemes
 - Mixed precision with fixed or learnable bitwidths
- ▶ Example pretrained models + training scripts
 - image classification, speech-to-text, text-to-speech
 - **<your quantized DNN model contribution here!>**
- ▶ Different ONNX export flows for different backends in progress
 - FINN, Xilinx DPU, standard ONNX



Xilinx/brevitas

PYTORCH

Tour of the FINN software stack



✓ **FINN** project landing page: <https://xilinx.github.io/finn>

finn-hlslib: library of Vivado HLS components

- ▶ 10+ common DNN layer types in Vivado HLS
 - <your HLS layer contribution here!>
- ▶ Highly configurable via C++ template arguments
 - Input, weight, output datatypes (precision)
 - Parallelism along different axes
 - Mapping to FPGA resources (LUT or DSP, LUTRAM or BRAM...)
- ▶ Easily composable components
 - AXI stream-in, AXI stream-out



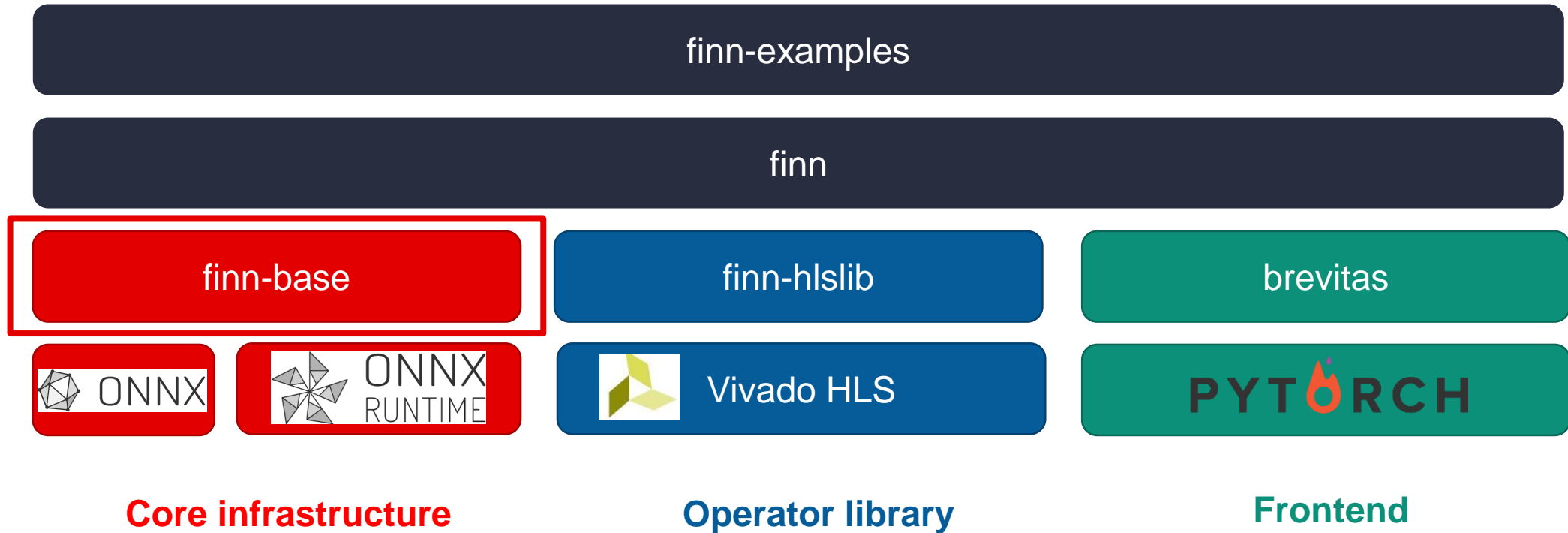
Xilinx/finn-hlslib



Read *the* Docs

finn-hlslib.
readthedocs.io

Tour of the FINN software stack



✓ **FINN** project landing page: <https://xilinx.github.io/finn>

finn-base: ONNX compiler infrastructure

- ▶ Infrastructure for manipulating + verifying custom ONNX graphs
 - Not tied to FINN's HLS op implementations or lowering flow
 - Useful for exploring DNN compilation without full FINN
- ▶ Three key parts
 - Wrapper around ONNX protobuf with helper functions
 - Defining + executing (for verification) custom ops
 - Defining + applying graph transformations
- ▶ Various other utilities
 - e.g. execute Verilog as part of ONNX custom op (with pyverilator)



Xilinx/finn-base



Read *the* Docs

finn-base.

readthedocs.io

Putting it all together: a FINN end-to-end flow

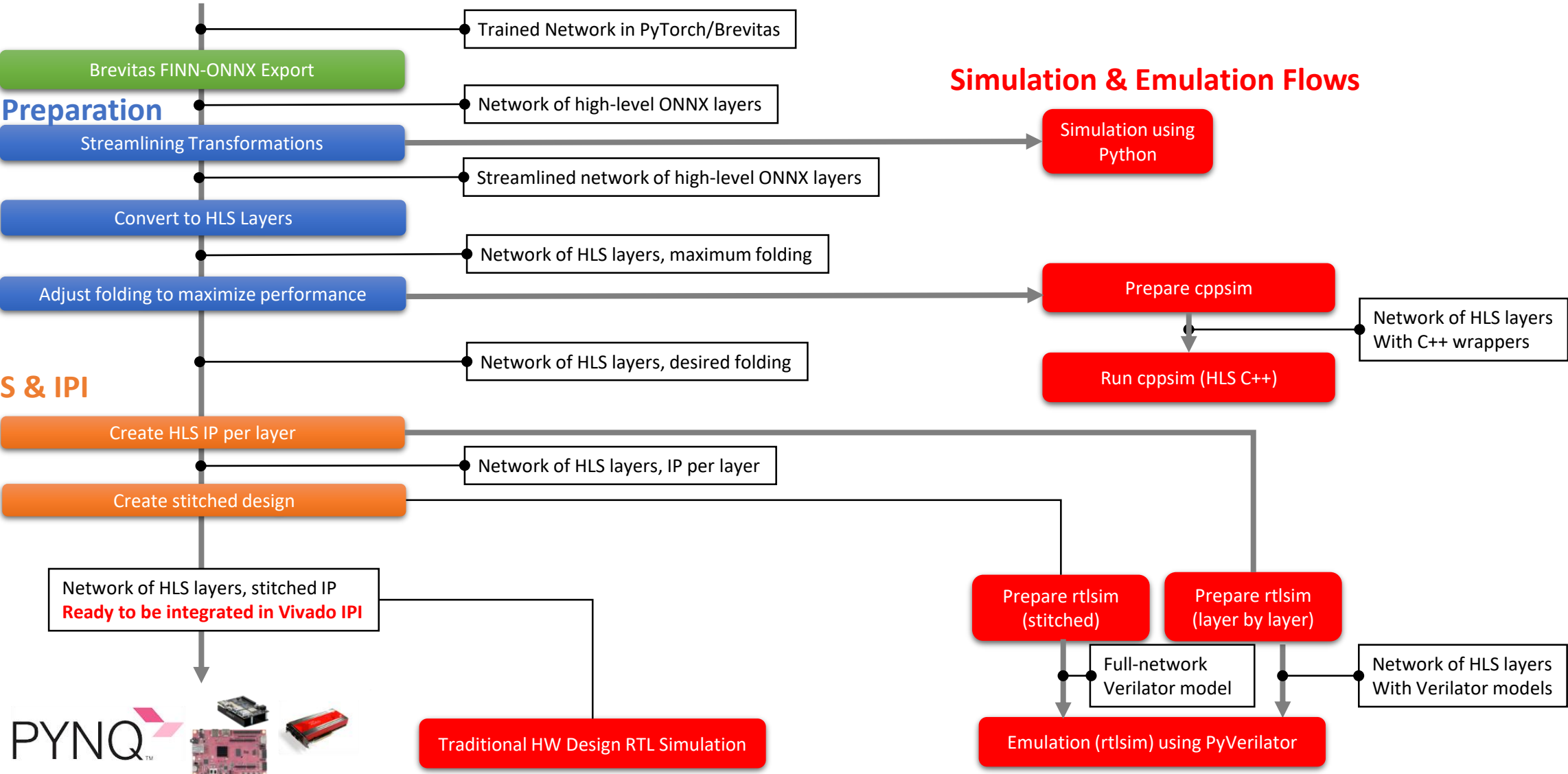
Brevitas

Network Preparation



Vivado HLS & IPI

Simulation & Emulation Flows





The FINN Community

Zaid Al-Ars
Associate Professor, TU Delft

Call for community building

1300+ stars on GitHub across
repos, 600+ citations across
papers

There are many other users +
use-cases we don't know about
-- we want to hear from you!

We welcome partners to
work together to build and
extend FINN!

We enable community
efforts and provide
support

Various types of engagement with FINN

Contribution to
FINN framework

Using FINN in
research

Using FINN in
industry

Using FINN in
education

Various types of engagement with FINN

Contribution to FINN framework

- University of Heidelberg
- Delft University of Technology

Using FINN in research

- Research organizations: ESA, Fraunhofer, CERN, hls4ml
- Various other FPGA NN papers use code from FINN
- LUTNet [FCCM'19], ReBNet [FCCM'18]

Using FINN in industry

- Companies: Daimler and Thales
- Ongoing evaluation by networking companies

Using FINN in education

- University classes in Stanford, UNC Charlotte, NTNU, et al.

Two examples from collaborators

❑ Contribution to FINN framework

LogicNets integration into FINN

Jon Ander Lezeta

❑ Using FINN in research

Enabling support of QuartzNet in FINN

Mirza Mrahorović & Jakoba Petri-König

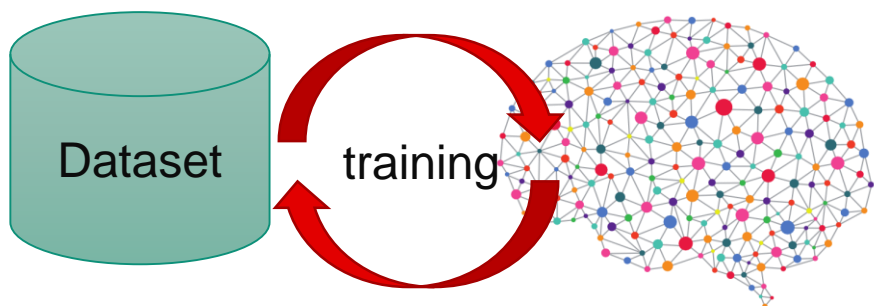


LogicNets integration into FINN

Jon Ander Lezeta

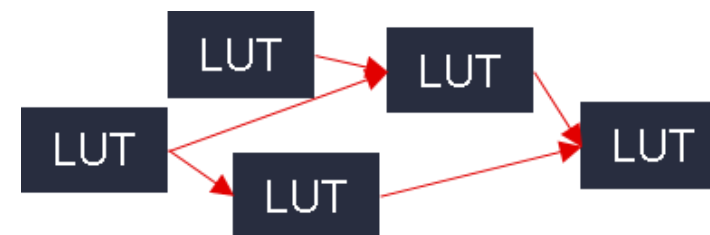
LogicNets

[Umuroglu et al. "LogicNets: Co-Designed Neural Networks and Circuits for Extreme-Throughput Applications", FPL'20]



(with **high sparsity**
+ **activation quantization**)

PyTorch



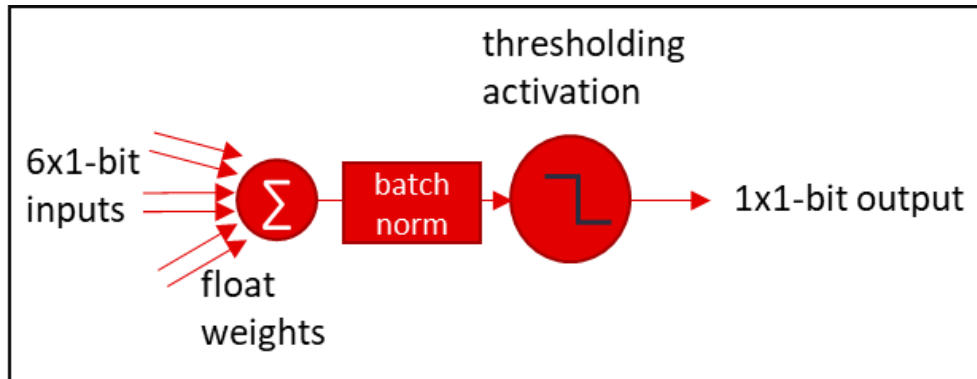
Fully-Spatial
Circuit Implementation

one full sample every clock
low logic depth, high F_{clk}
e.g. intrusion detection at 450M+ samples/second

FPGA

LogicNets

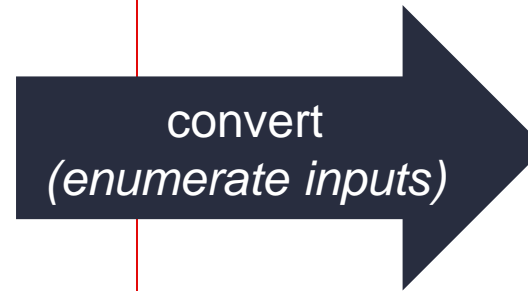
Neuron Equivalent (**NEQ**)



Total input: 6 bits
Total output: 1 bit

Hardware cost: 1 x LUT6

Hardware Building Block (**HBB**)



Total input: 6 bits
Total output: 1 bit

PyTorch FPGA

Limitation of current LogicNets

► LogicNets

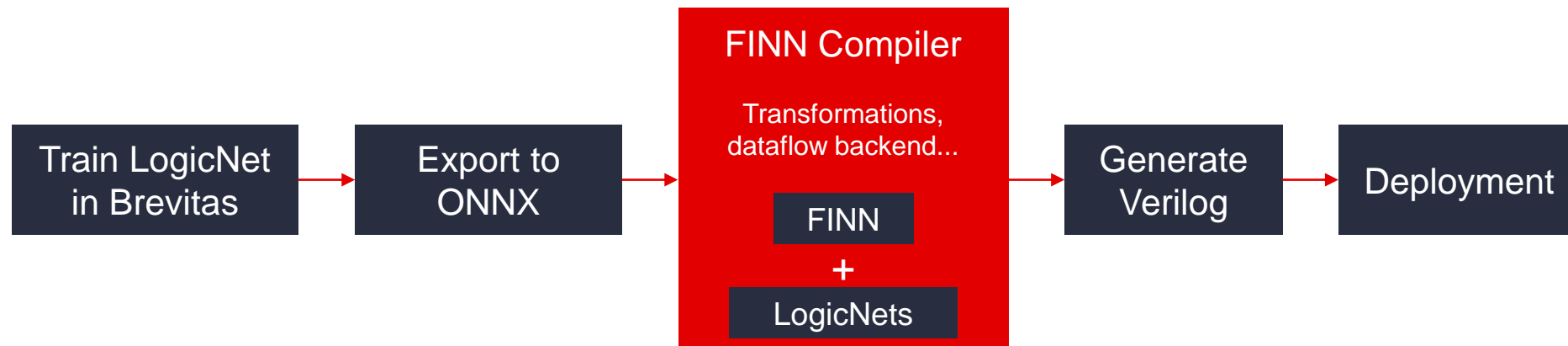
- ↯ Good approach for sparse layers
- ↯ Compute quantized neurons with complex functions in LUTs

► Image Classification

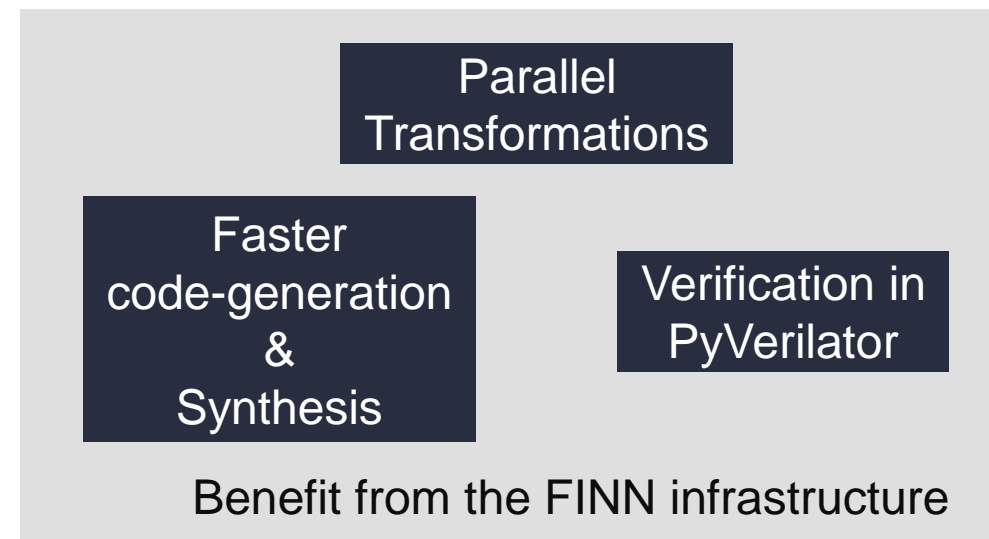
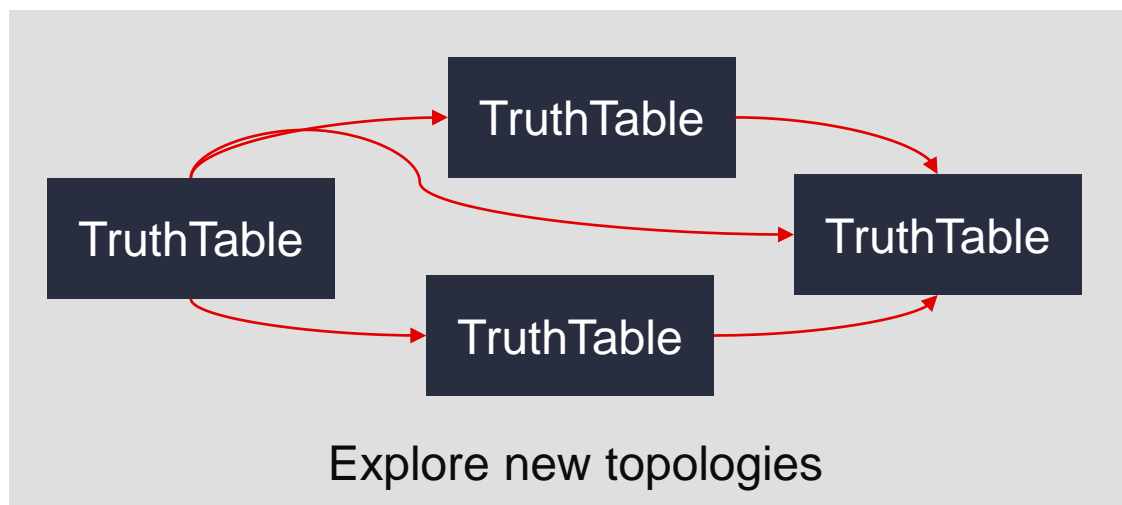
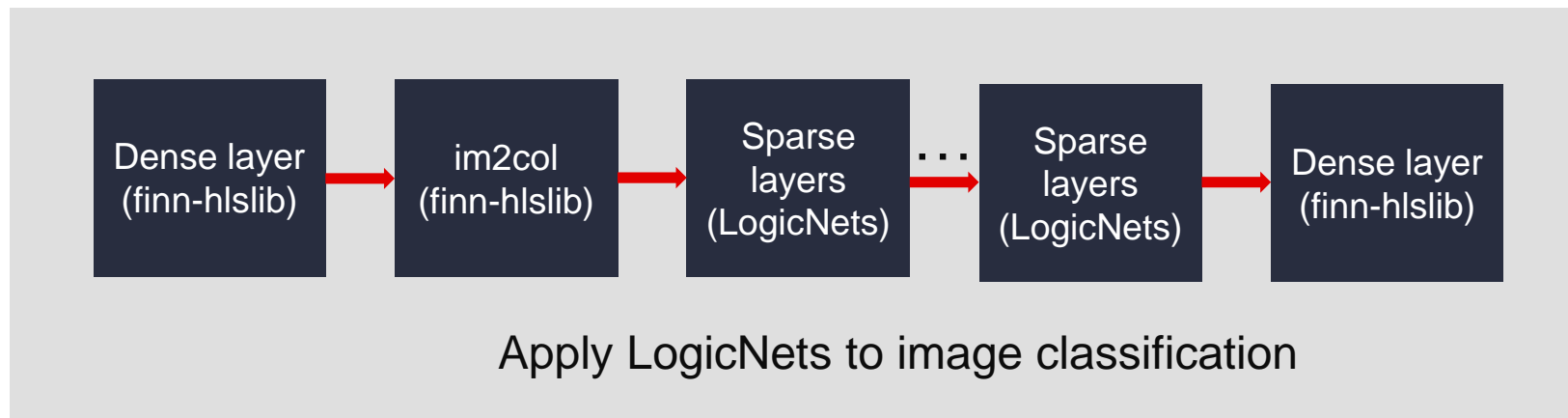
- ↯ First and last layers sensitive to extreme quantization and pruning
- ↯ Can cause a noticeable accuracy drop

► **Solution: FINN + LogicNets**

- ↯ Implement a LogicNets backend in FINN to mix FINN and LogicNets topologies



Benefits of LogicNets in FINN

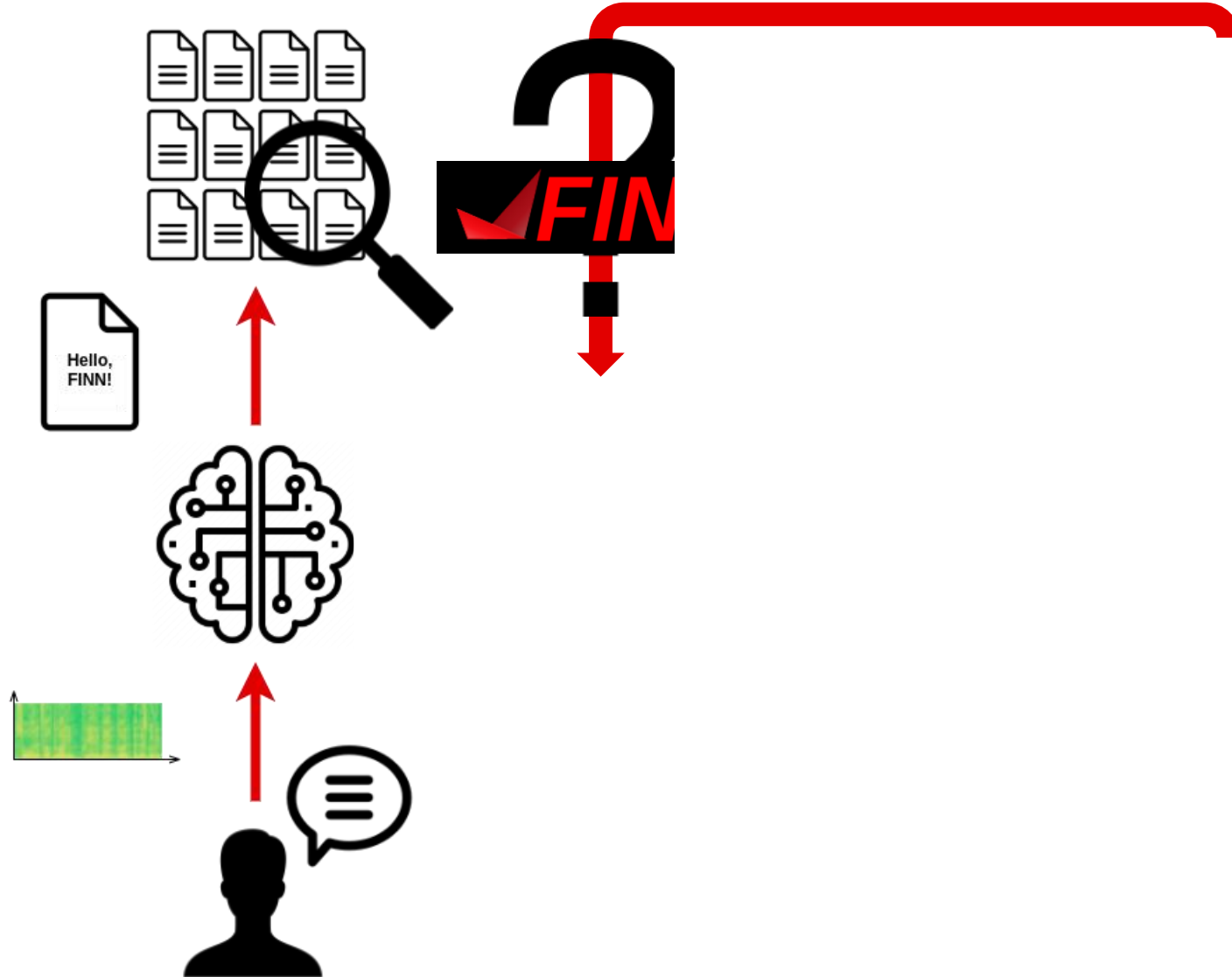




Enabling support of QuartzNet in FINN

Mirza Mrahorović

Enabling support of QuartzNet in FINN



Quantized model

Enabling support of QuartzNet in FINN

Brevitas

Brevitas FINN-ONNX Export

- Added new & reused existing transformations
- Support for non-square convolution lowering

Network Preparation

Streamlining Transformations

Convert to HLS Layers

Adjust folding to maximize performance

Support for non-square inputs

Vivado HLS & IPI

Create HLS IP per layer

Create stitched design

Network of HLS layers, stitched IP
Ready to be integrated in Vivado IPI

In the end:

1. FINN support for models with (non-)square inputs
1. FPGA accelerator for QuartzNet



Call for community building

- ❑ For questions and collaboration ideas please contact!
- ❑ Gitter channel for communication
↴ <https://gitter.im/xilinx-finn/community>
- ❑ Zaid Al-Ars: z.al-ars@tudelft.nl
- ❑ Jakoba Petri-König: J.Petri-Koenig@tudelft.nl
- ❑ Yaman Umuroglu: yamanu@xilinx.com

Q&A session



Thank You

