




Efficient Hardware for Neural Network Processing

Building Inference Engines for
Quantized Neural Networks
Using **Brevitas** and  **FINN**

AMD Research and Advanced Development (RAD)

- **Integrated Comms and AI Lab**
 - ~20 researchers plus university program
 - 5 different locations
 - Established as Xilinx Research Labs and AMD Research
- **Focus: AI and Communications**
 - Building systems, architectural exploration, algorithmic optimizations, benchmarking
 - In collaboration with partners, customers, and universities
 - ETH Zürich, Paderborn University, TU Delft, Imperial College, KIT, NTNU, Politecnico di Milano, NUS, University of Sydney
 - Active Internship Program
 - On average 10 interns at any given time



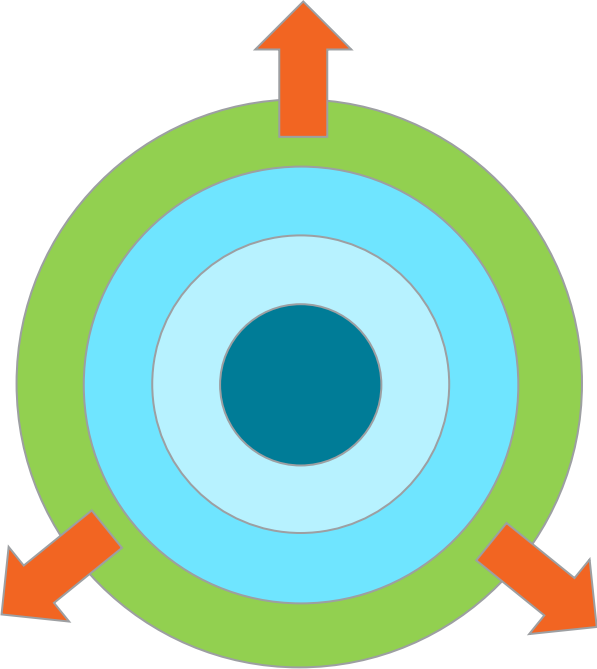
Agenda

-
1. The FINN project
 2. Status and Roadmap

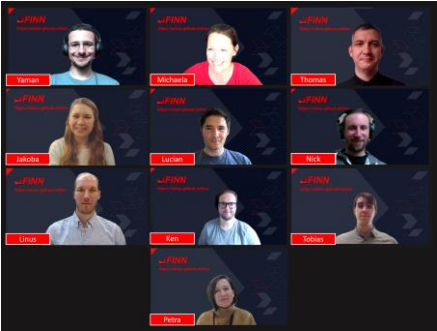
The FINN Project



FINN Project: A Brief Timeline



- 2016 Research project on binarized NNs (BNNs)
- 2017-19 Extended to multi-bit QNNs, multiple application domains
- 2019-21 Rebuilt from scratch for open source, growing community
- 2022-... Professional customer support through CSE Technology and transfer of compiler stack to Engineering



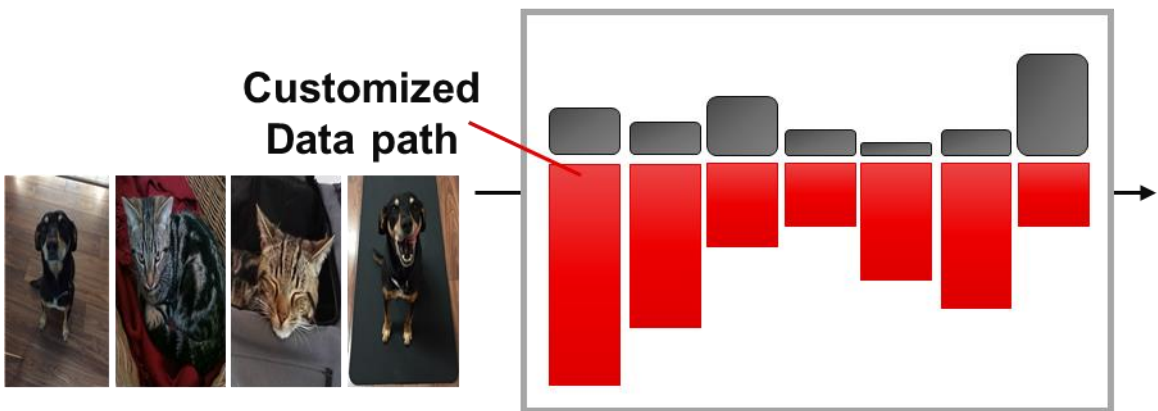
Research and Advanced Development



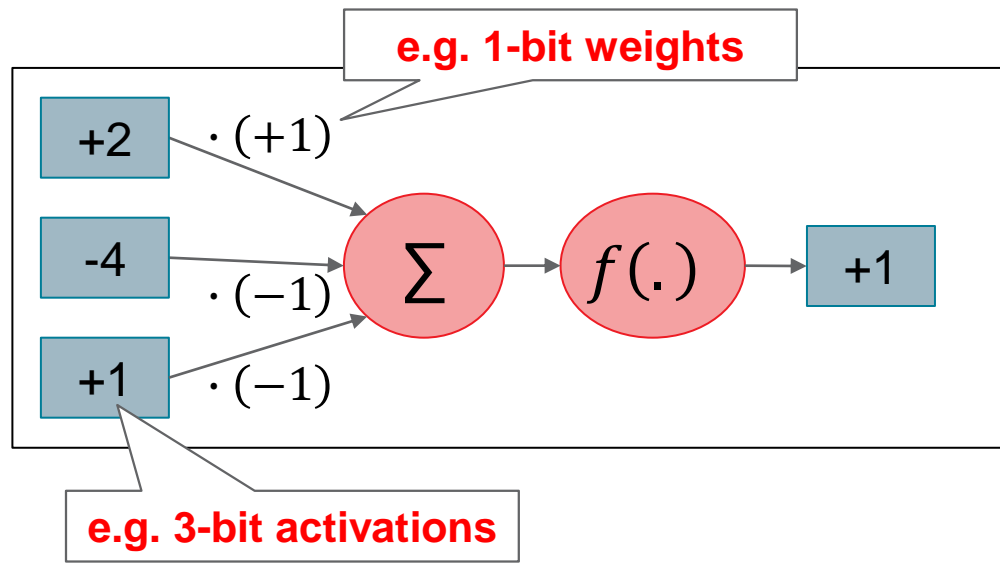
Custom and Strategic Engineering

Two Key Techniques for Customization in FINN

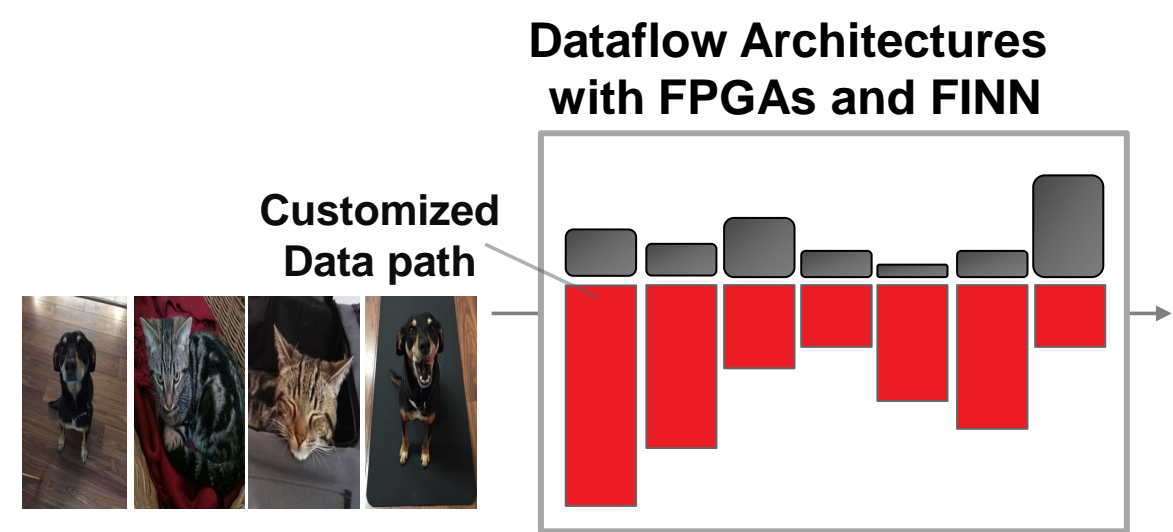
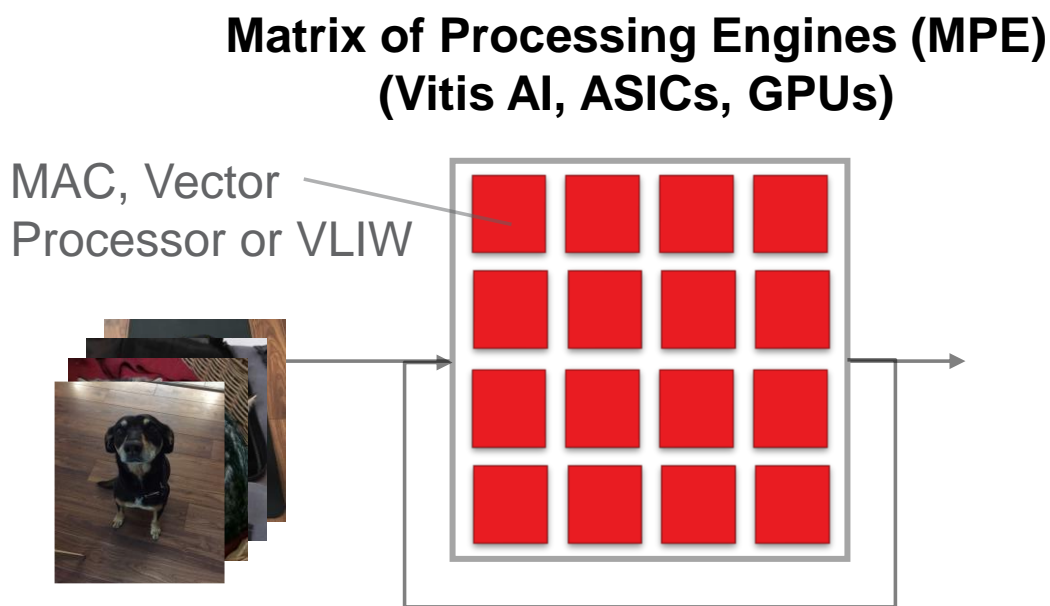
Streaming Dataflow Architectures for FPGAs



Custom Precision: Few-bit Weights and Activations

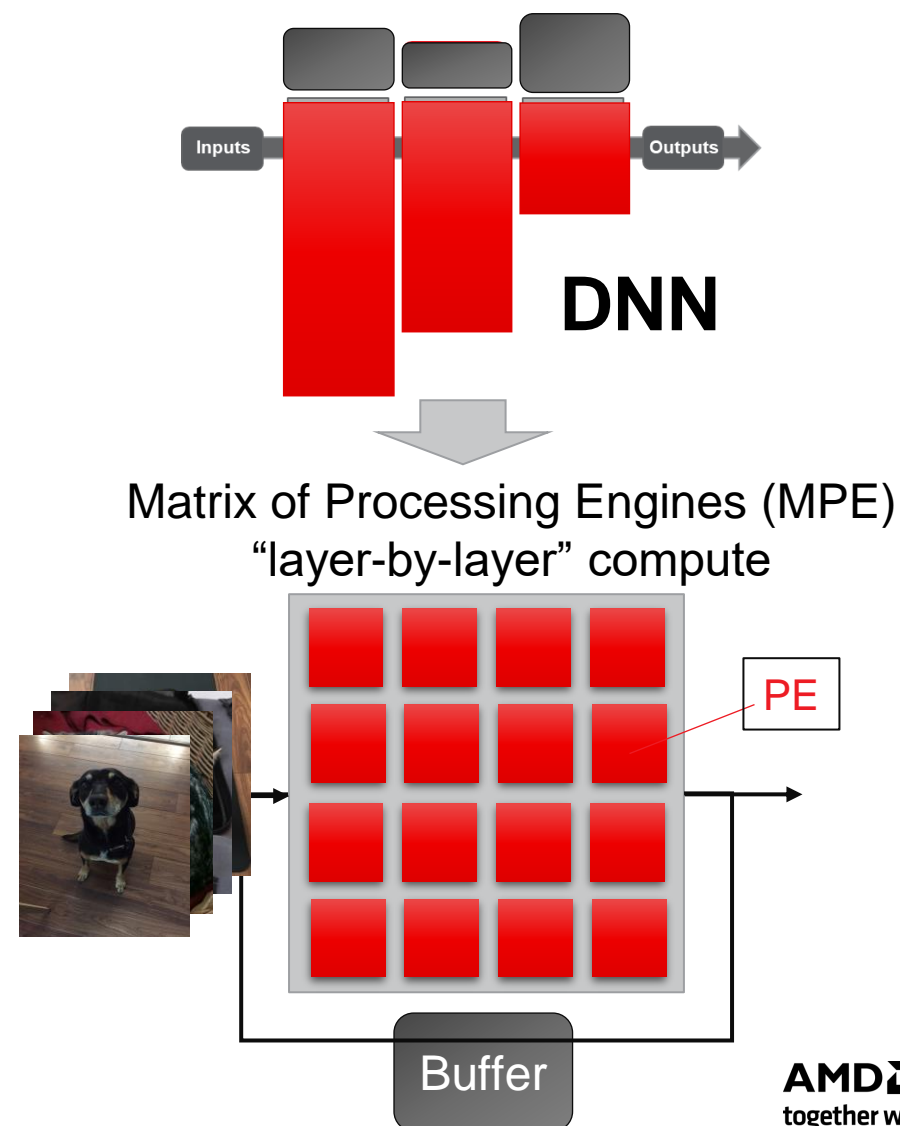


Customized Dataflow Processing versus More Generic Architectures



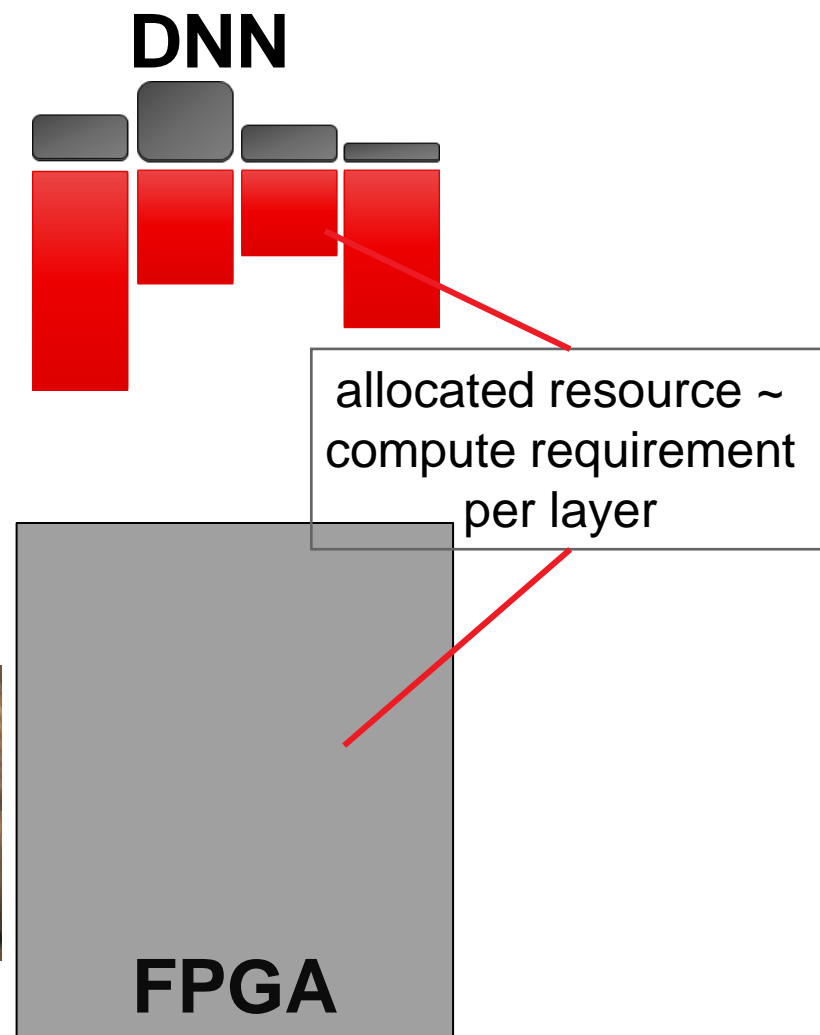
Matrix of Processing Engines (MPEs) Specializing for AI in General

- Popular layer-by-layer compute
- Batching to achieve high compute efficiency
 - At latency cost (latency ~ batch size)
- Customized for ML in general
 - Designed to run any DNN
 - Specialized processing engines
 - Operators
 - ALU types
- Works really well for computer vision and natural language processing
- Popular approach: VitisAI (FPGA or AIE) as well as majority of AI accelerators



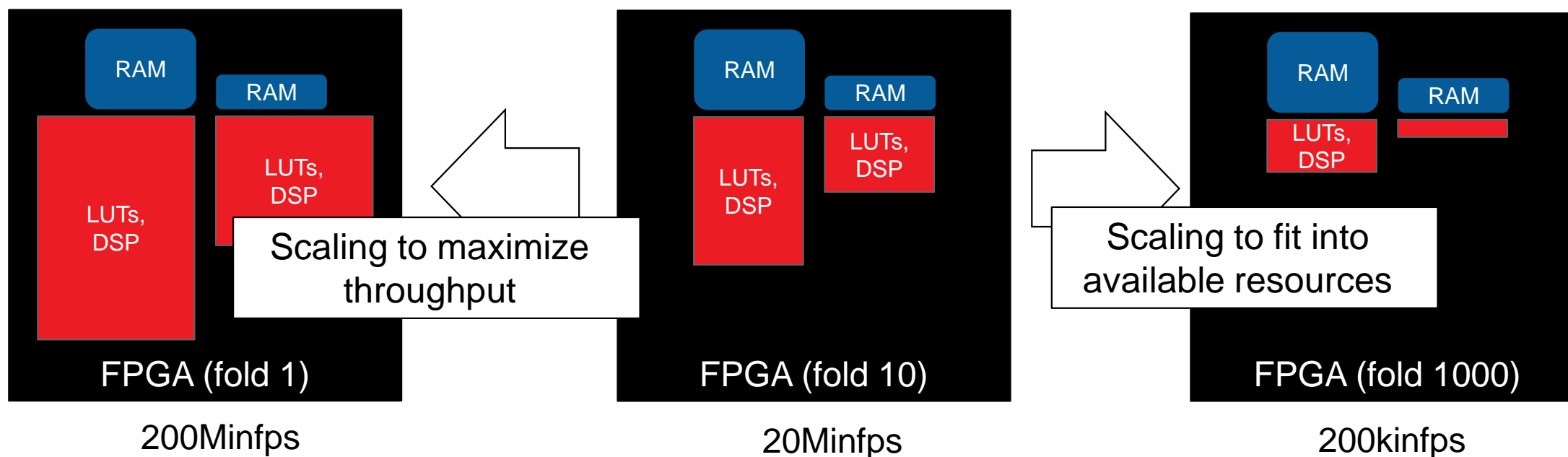
Dataflow - Specializing for Individual Topologies

- Hardware instantiates the topology as a dataflow architecture
- Customize everything to the specifics of the given DNN, any operation, any connectivity
- Benefits:
 - Improved efficiency
 - Low fixed latency
- Scale performance and resources to meet the application requirements



Dataflow Processing:

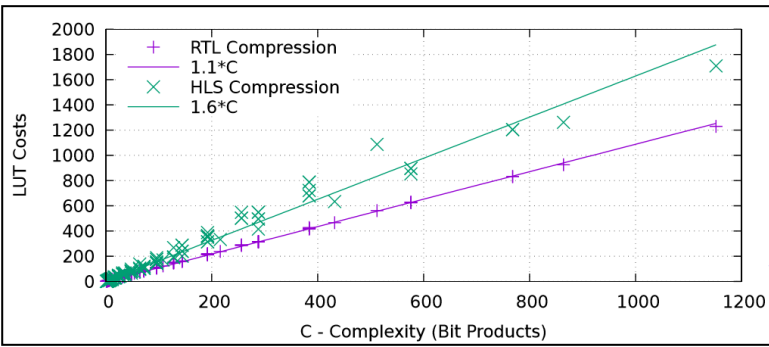
Scaling to Meet Performance and Resource Requirements



1. Scale performance and resources to meet the application requirements
2. If resources allow, we unfold completely, creating a circuit for inference at clock speed

Customizing Arithmetic to Minimum Precision

- Popular approach which reduces bits in the data representation of weights and activations while preserving accuracy
- Reducing precision shrinks hardware cost/scales performance
 - Instantiate n-times more compute within the same fabric, thereby scale performance n-times
- Reduces memory footprint
 - NN model can stay on-chip => no memory bottlenecks
- With dataflow: every layer has dedicated compute resources, we can mix and match precision across layers
 - Exploit custom arithmetic at a greater degree than MPEs

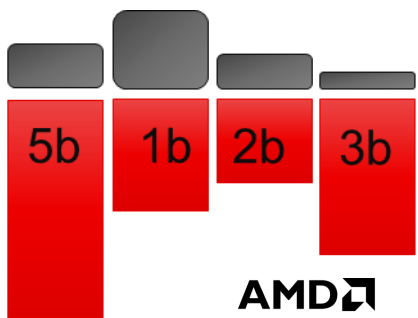


C= f(size of accumulator, size of weight,size of activation)

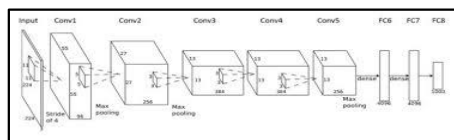
Precision	Modelsize [MB] (ResNet50)
1b	3.2
8b	25.5
32b	102.5

**Reducing precision saves resources/ scales performance,
and reduces memory**

However, it requires quantization support in the training software



FINN Framework: From DNN to FPGA Deployment



Brevitas
Training in PyTorch
Algorithmic optimizations

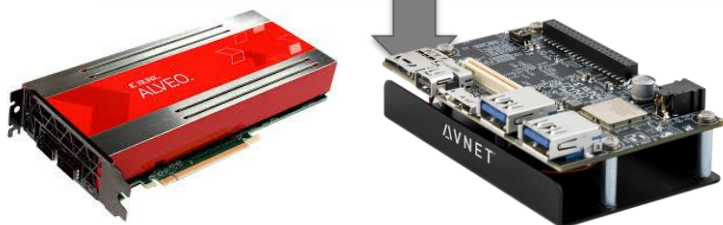
- Train or even learn reduced precision DNNs
- Library of standard layers
- Pretrained examples

FINN Compiler
Hardware Architecture
Build

- Perform optimizations
- Assemble parameterized HLS/RTL modules
- Generate a DNN hardware IP

Deployment

- Embed the DNN IP into an infrastructure design
- Generate a Python run-time
- Enable integration with your application
- System integration available for some embedded and Alveo platforms, including HACC



Status and Roadmap



Status Summary

- **Many strategic customer engagements**
 - Including Sick AG
- **Open-Source Adoption**
 - ~2k+ GitHub stars summarized across repos
 - 250k+ Brevitas downloads
 - ~200k QONNX downloads
 - 17k+ FINN compiler downloads
- **Academic Results**
 - ACM TSETS 2020, FPL'2020, DFT'2019 Best Paper awards
 - 1000+ citations on original paper
- **University Classes on computer architecture for ML with FINN**
 - Stanford, UNC Charlotte, NTNU in Norway, EPFL in Switzerland
 - Regular tutorials, also available on YouTube: <https://www.youtube.com/watch?v=zw2aG4PhzmA>
- **Business units providing customer support**
 - Lead engineering team: Custom and Strategic Engineering, Dublin

*"The FINN toolset is showing **huge potential** using it in upcoming **SICK products**. It is **easy to use** and with an **extraordinary performance** and very promising results. In the future, flexible implementations of ML in our products with FINN can be a great advantage and even replace static architectures as they are currently used. Thanks to the FINN team for the great cooperation"*

<https://github.com/Xilinx/brevitas>
<https://github.com/Xilinx/finn>
<https://github.com/Xilinx/finn-hlslib>
<https://github.com/Xilinx/finn-examples>
<https://github.com/fastmachinelearning/qonnx>

FINN Compiler Updates

**FINN v0.10
Release**

- **Refactoring** of operator instantiation infrastructure
 - FINN compiler used to assume that hardware blocks are synthesized from on HLS
 - New class hierarchy to facilitate integration of RTL components
 - Provide users with an interface to override the compiler's choice for HLS vs. RTL implementation on a per-layer basis
- **RTL component** library optimizing the implementations of critical layers
 - Efficient implementation of 4-bit and 8-bit compute leveraging DSP slices
 - Efficient implementation of multi-level thresholding
 - Eradication of (regularly long) HLS synthesis times for layers with an RTL option
- **Compiler optimization pass** for accumulator and weight bit width minimization
- **Added board support** in system integration flow
 - **RFSoc 4x2** and **U55C** (contributed by University of Paderborn)

FINN Technical Roadmap: Capabilities

- Operator Hardening
 - **Revised RTL Thresholding by binary search**
 - Ingestion of fp32 inputs
 - DSP-enabled Generalized Datatype Support
 - Efficient higher-precision integer compute: **int4**, **int8**, ..., int16
 - Small standard floating-point formats: float16, bfloat16
 - Custom MiniFloats: fp4 – fp8
 - Internal clock pumping of DSP datapaths to increase their operational density. We are aiming at an operational frequency around 500 MHz
- New Operators
 - **SeLU activation function**
 - Optimized **transposed convolution**
 - Fallback float layers to mitigate streamlining limits

FINN Technical Roadmap: Ease of Use

- **FINN Library**

- Refactoring of streamed layer interfaces
 - Packed flat `ap_uint<W>` → explicit `hls::vector<T, N>`
- Combining HLS and RTL components into one FINN Library

- **FINN Examples**

- MobileNet-v1 and VGG10-RadioML **with efficient DSP compute**
- New example: German Traffic Sign Recognition Benchmark

COPYRIGHT AND DISCLAIMER

©2024 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate releases, for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS.' AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

