DESIGN, AUTOMATION & TEST IN EUROPE
09 – 13 March 2020 · ALPEXPO · Grenoble · France
The European Event for Electronic System Design & Test

XpulpNN: Accelerating Quantized Neural Networks On RISC-V Processors Through ISA Extensions

Angelo Garofalo[1], Giuseppe Tagliavini[1], Francesco Conti[1,2], Davide Rossi[1], Luca Benini[1,2]

[1]EEES Lab, University of Bologna
[2]IIS Lab, ETH Zurich

Outline

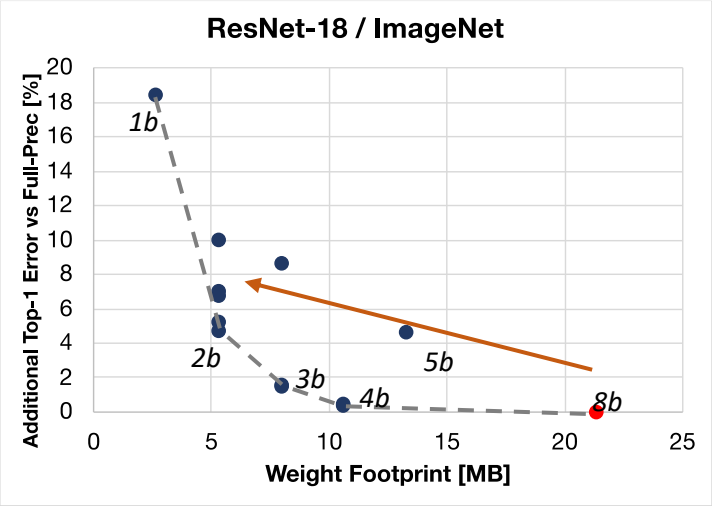
- **Introduction**
- Background
  - RI5CY Core
  - QNN Execution Model
- Contribution of the Work
  - XpulpNN ISA Extensions
  - µArchitecture
- Results
  - Functional Performance
  - Physical Implementation
  - Benchmarking
- Conclusion

Embedding Intelligence at the Extreme Edge

- **ML and DL for data processing on IoT end-nodes (Extreme Edge AI)**
  - Squeeze raw data in more semantically dense format (classes, high-level features/symbols);
  - Less power to trasmit data wirelessly.
- **Challenges:**
  - High computational and memory requirements of ML (DL);
  - Limited memory and computation capabilities of IoT end-nodes (micro-controller systems).
- **HW/SW Codesign to:**
  - Reduce model size (integer arithmetic, quantization, pruning);
  - Increase performance and efficiency (MAC, SIMD, Vectors..).

SoA Edge AI: SW solutions

SoA Quantization Results on ResNet-18



Quantized Neural Networks (QNNs) are a natural target for execution on constrained extreme edge platforms.

Quantizazion of a MobilenetV1_224_1.0 (*)

Quantization Method	Top1 Accuracy	Weight Memory Footprint
Full-precision [11]	70.9%	16.27 MB
PL+FB INT8 [11]	70.1%	4.06 MB
PL+FB INT4 (our)	0.1%	2.05 MB
PL+ICN INT4 (our)	61.75%	2.10 MB
PC+ICN INT4 (our)	66.41%	2.12 MB
PC W4A4 [16]	64.3%	-
PC W4A8 [13]	65%	-
PC+Thresholds INT4 (our)	66.46%	2.35 MB

- < 5 % accuracy loss with respect to Full-precision;
- ~7x memory saving with respect to Full-precision;
- Thresholding – based quantization after MatMul.

(*) Rusci M. et al., Memory-Driven Mixed Low Precision Quantization For Enabling Deep Network Inference On Microcontrollers. . arXiv preprint arXiv:1905.13082.

SoA Edge AI: QNN Embedded Computing Platform

	ASICs	FPGAs	MCUs
Power Budget [mW]	1 – 1 K	1 – 1 K	1 – 1 K
Throughput [Gop/s]	**1 K – 50 K**	10 – 200	0.1 – 2
Energy Efficiency [Gop/s/W]	**10 K – 100 K**	1 - 10	1 – 50
Performance	**High**	Medium	Low
Flexibility	Low	Medium	**High**
Cost	High	Medium	**Low**

Extreme edge IoT devices:

- End-nodes have to be cheap to be economically feasible, and software programmable;

MCU Instruction Set Architecture

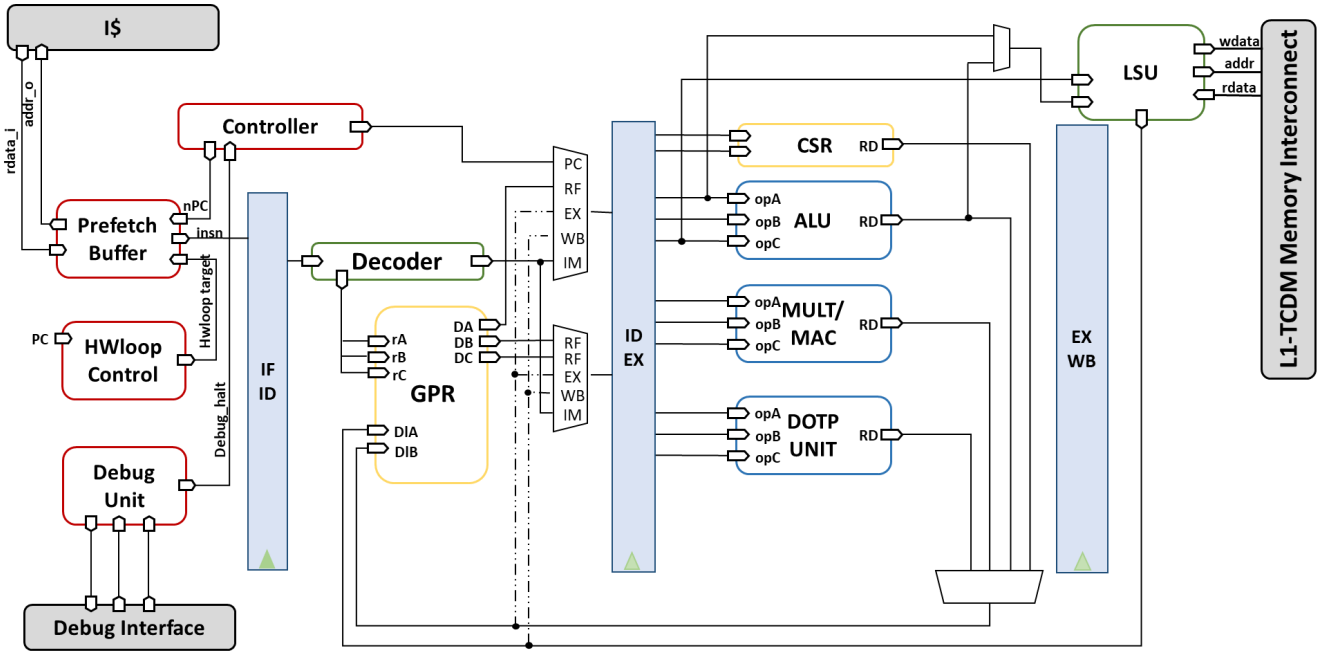
- **Armv8.1-M**: new ISA extensions for ML propsed by Arm [Online, 2020]:
  - Low overhead loops;
  - 8-bit data types support in vector extension;
  - No implementation of this ISA is commercially available (Cortex-M55 just released by ARM).
- **XpulpV2**: RISC-V ISA Extensions for efficient digital signal processing [Gautschi et Al., 2017]:
  - Discussed in the following.

⇨ **Current generation MCUs' ISAs lack support for low-bitwidth SIMD arithmetic instructions.**

Outline

- Introduction
- **Background**
  - RI5CY Core
  - QNN Execution Model
- Contribution of the Work
  - XpulpNN ISA Extensions
  - µArchitecture
- Results
  - Functional Performance
  - Physical Implementation
  - Benchmarking
- Conclusion

Background: RI5CY core



RI5CY(*):

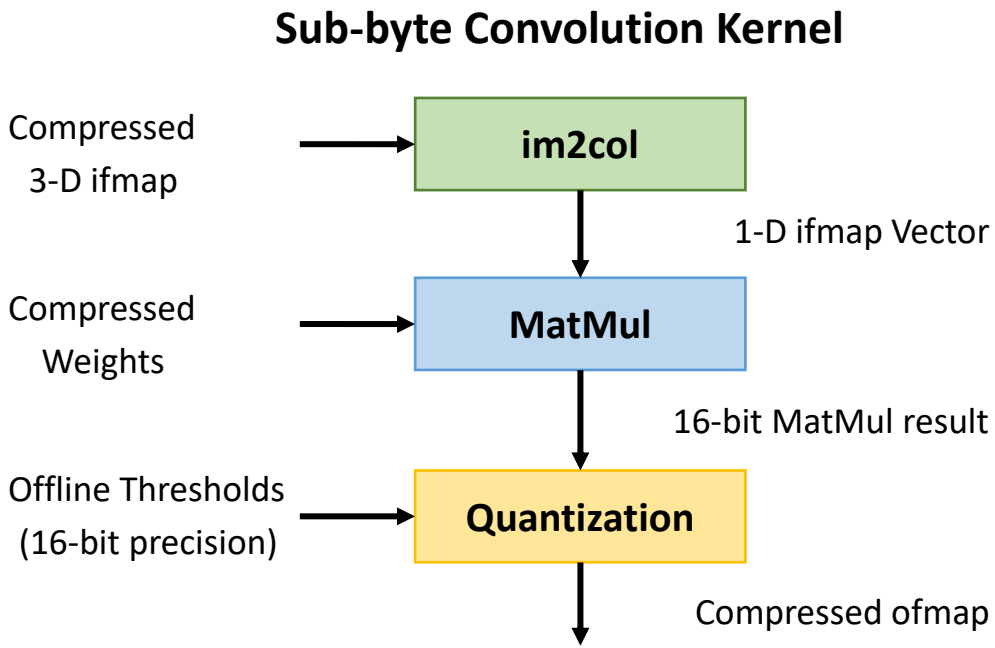
- 4-stage in order single-issue pipeline;
- RISC-V RV32IMCXpulpV2 ISA.

XpulpV2(*):

- Hardware Loop;
- LD/ST with post-increment;
- MAC and 16-/8-bit SIMD instructions;
- Bit Manipulation instructions.

(*) Gautschi et al., Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices, IEEE VLSI, 2017.

Background: QNN Execution Model



Im2col:

- 1-D Vector precision has to match data types natively supported by the underlying HW with SIMD instructions;

MatMul:

- Computes SIMD sdotp of the current im2col to be convolved;

Quantization into Q-bits:

- Compares MatMul result with $2^Q - 1$ thresholds per ofmap channel;
- Binary search tree;

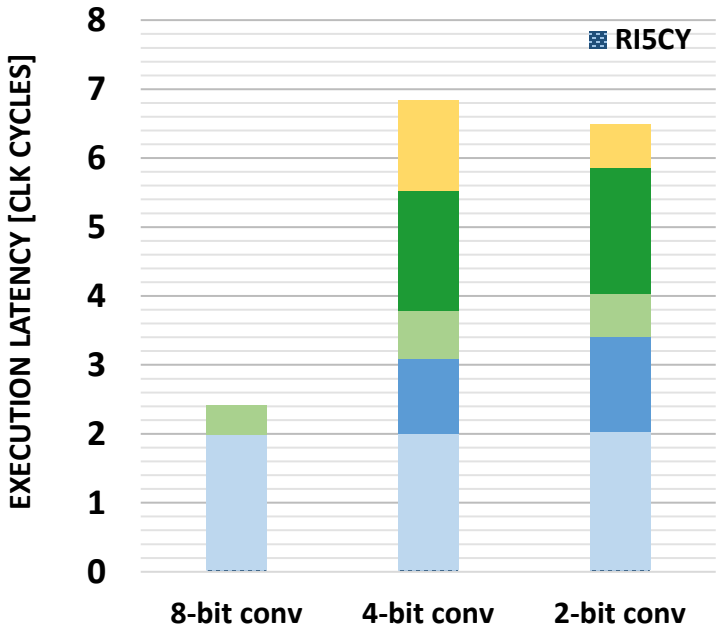
ARM ISA: Rusci et Al. "Quantized NNs as the definitive solution for inference on low-power ARM MCUs?: Work-in-progress." , CODES 2018.
RISCV ISA: Garofalo et al. "PULP-NN: accelerating quantized neural networks on parallel ultra-low-power RISC-V processors." Phil. Trans. A 2019.

Outline

- Introduction
- Background
  - RI5CY Core
  - QNN Execution Model
- **Contribution of the Work**
  - XpulpNN ISA Extensions
  - µArchitecture
- Results
  - Functional Performance
  - Physical Implementation
  - Benchmarking
- Conclusion

XpulpNN: Motivation

Sub-byte Convolution Kernel on XpulpV2 ISA



MatMul kernel

Overhead to extract compressed weights

Im2col function

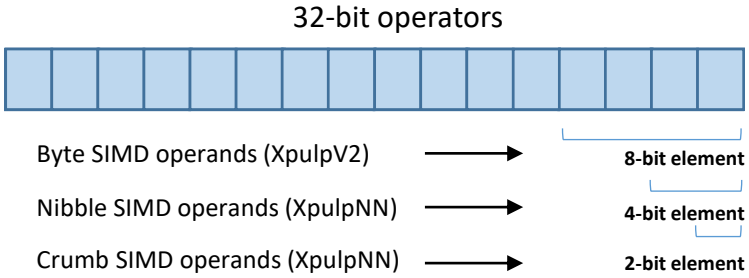
Overhead to extract compressed ifmap

SW Quantization process

Objective: eliminate the overhead to boost the performance and the efficiency of sub-byte kernels

Xpulpnn ISA Extensions

Arithmetic SIMD instructions



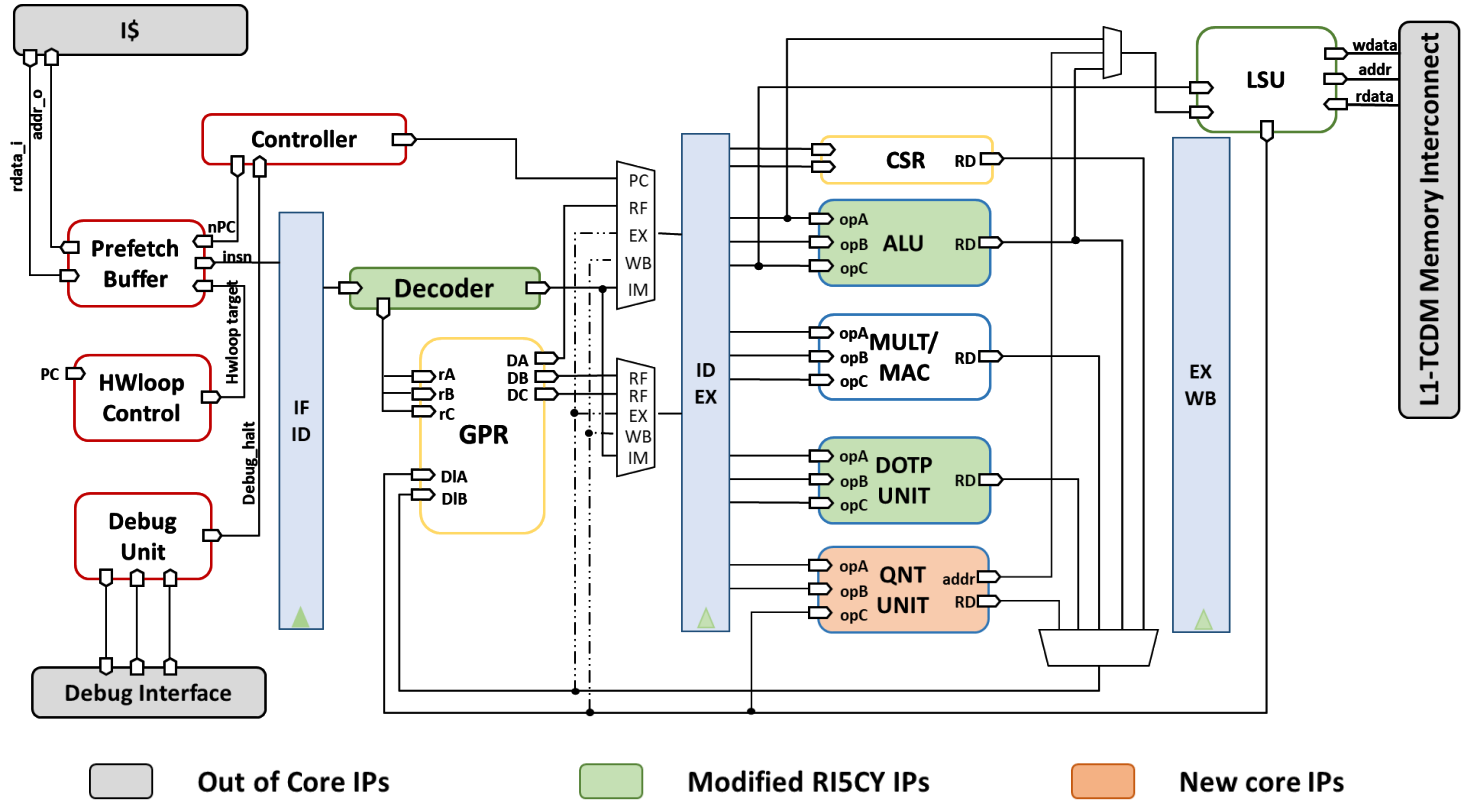
Supported Ops: ALU, Comparison, Shift, abs, Dot Product

No need to unpack sub-byte data

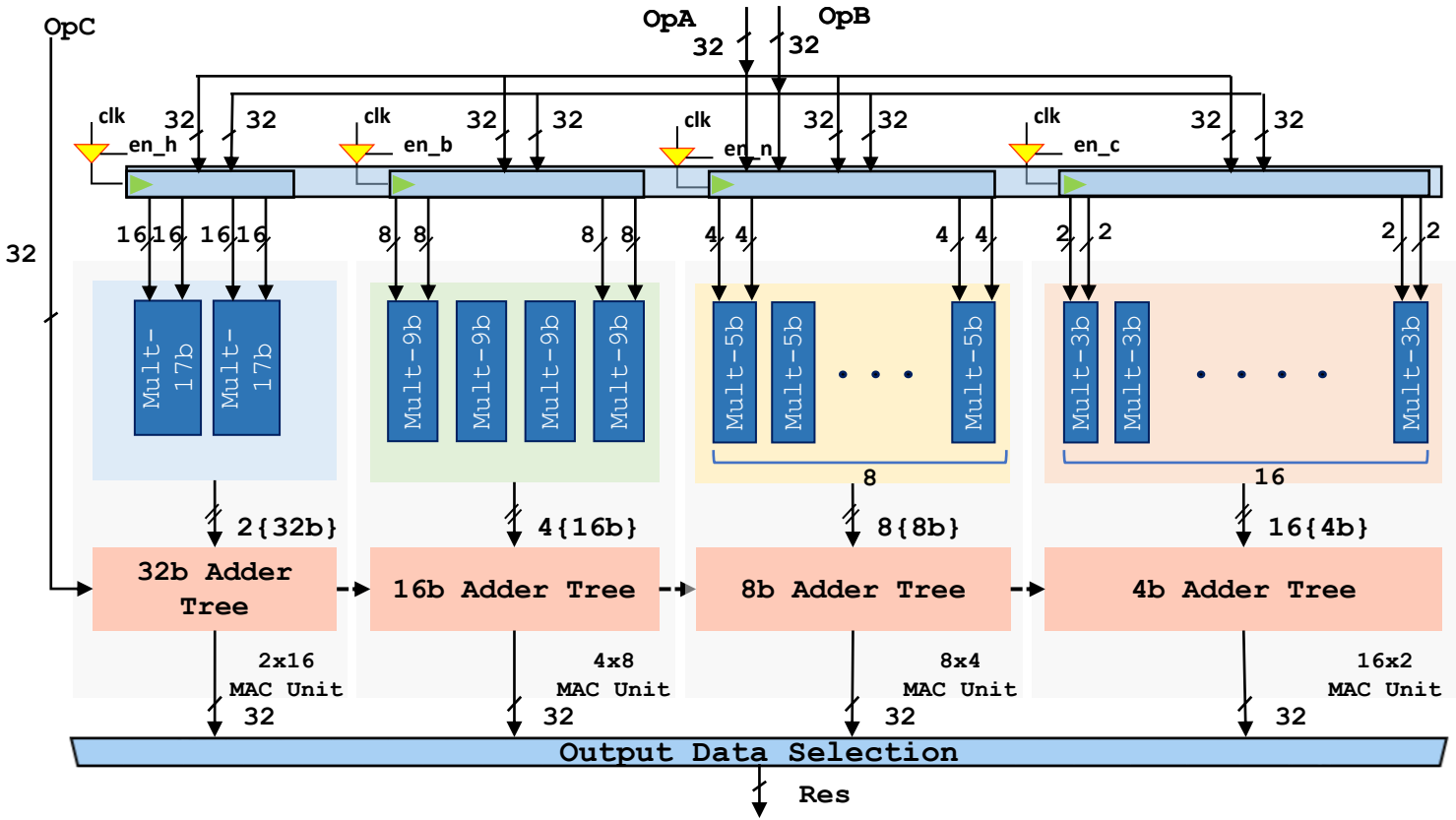
Multi-cycle instruction to efficiently handle the quantization process in HW.

ALU SIMD Op.	**Description for *nibble***
pv.add[.sc].{n, c}	rD[i] = rs1[i] + rs2[i]
pv.sub[.sc].{n, c}	rD[i] = rs1[i] - rs2[i]
pv.avg(u)[.sc].{n, c}	rD[i] = (rs1[i] + rs2[i])>>1
Vector Comparison Op.	
pv.max(u)[.sc].{n, c}	rD[i] = rs1[i] > rs2[i] ? rs1[i] : rs2[i]
pv.min(u)[.sc].{n, c}	rD[i] = rs1[i] < rs2[i] ? rs1[i] : rs2[i]
Vector Shift Op.	
pv.srl[.sc].{n, c}	rD[i] = rs1[i] >> rs2[i] Shift is logical
pv.sra[.sc].{n, c}	rD[i] = rs1[i] >> rs2[i] Shift is arithmetic
pv.sll[.sc].{n, c}	rD[i] = rs1[i] << rs2[i]
Vector abs Op.	
pv.abs.{n, c}	rD[i] = rs1[i] < 0 ? -rs1[i] : rs1[i]
Dot Product Op.	
pv.dotup[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7]
pv.dotusp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7]
pv.dotsp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7]
pv.sdotup[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7] + rD
pv.sdotusp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7] + rD
pv.sdotsp[.sc].{n, c}	rD = rs1[0]*rs2[0] + ... + rs1[7]*rs2[7] + rD
Quantization Op.	
pv.qnt.{n, c}	Dedicated Quantization instruction

HW µArchitecture: Extending the RI5CY core

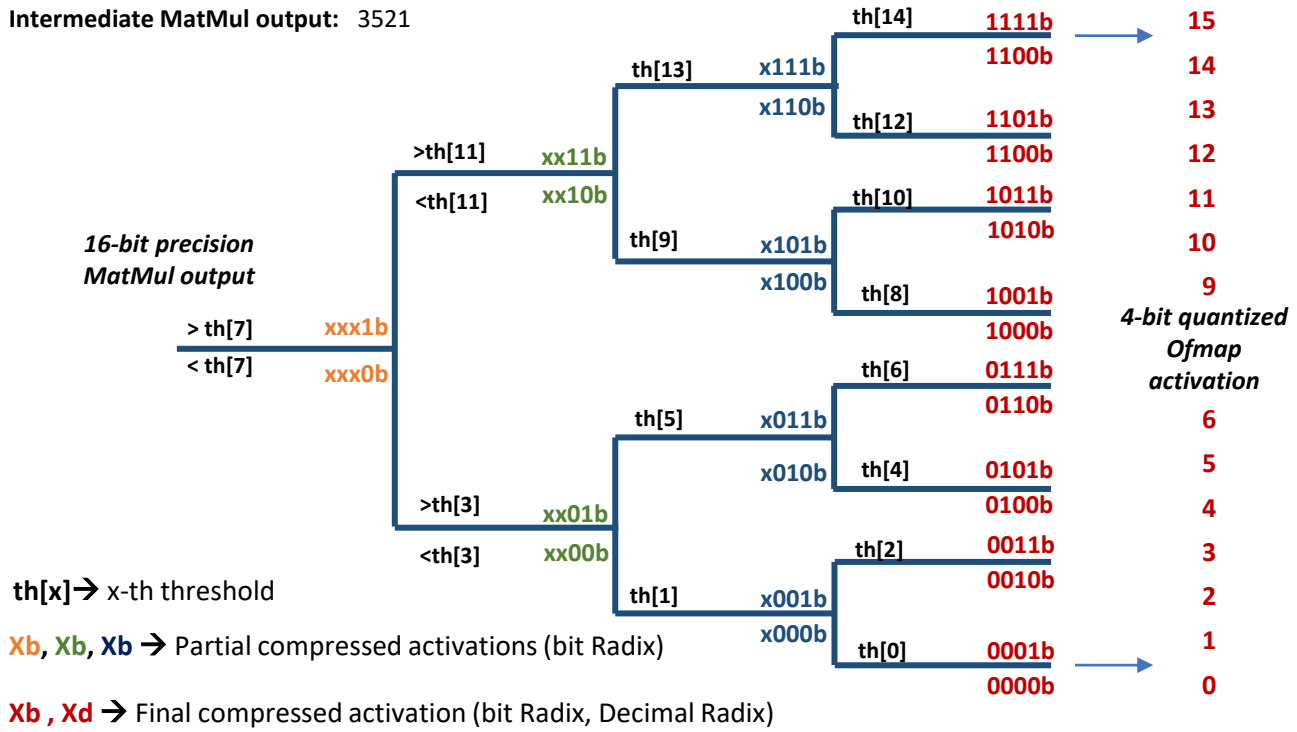


HW µArchitecture: Integer Dotp Unit



HW µArchitecture: Quantization Process

Threshold Vector: {-17432, -15253, -13074, -10895, -8716, -6537, -4358, -2179, 0, 2179, 4358, 6537, 8716, 10895, 13074, 0}



th[x]→ x-th threshold

Xb, Xb, Xb → Partial compressed activations (bit Radix)

Xb , Xd → Final compressed activation (bit Radix, Decimal Radix)

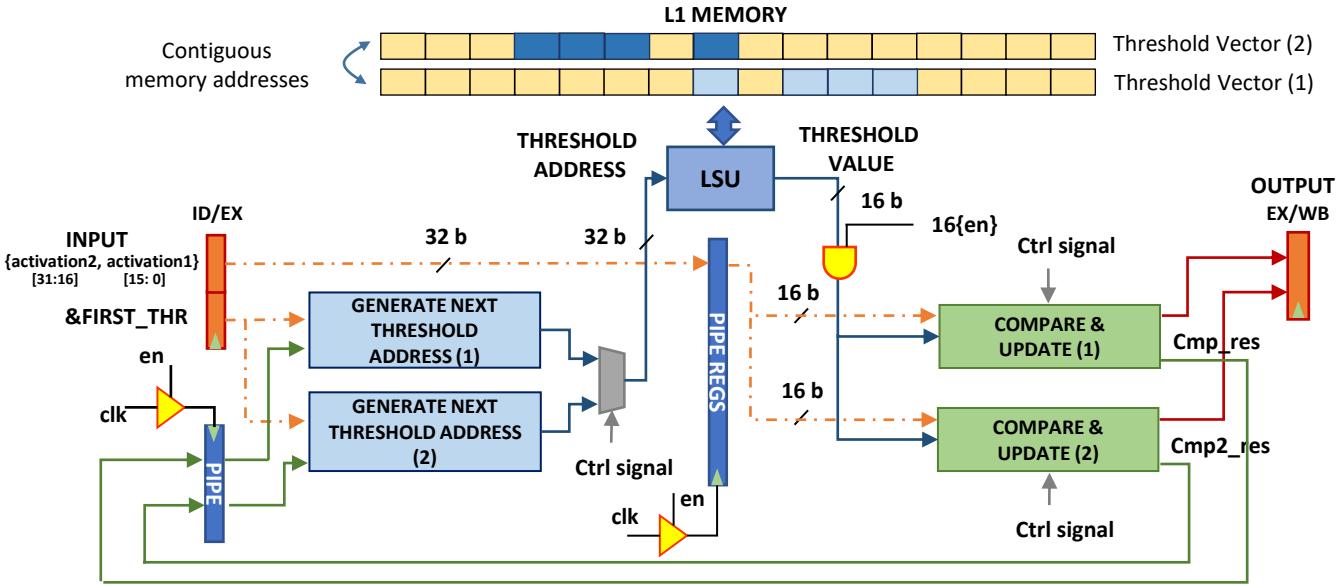
SW quantization:

- Nested If/else statements → huge numer of branches;
- 18 cycles / 9 cycles to quantize one 4- /2-bit ofmap activation.

HW quantization:

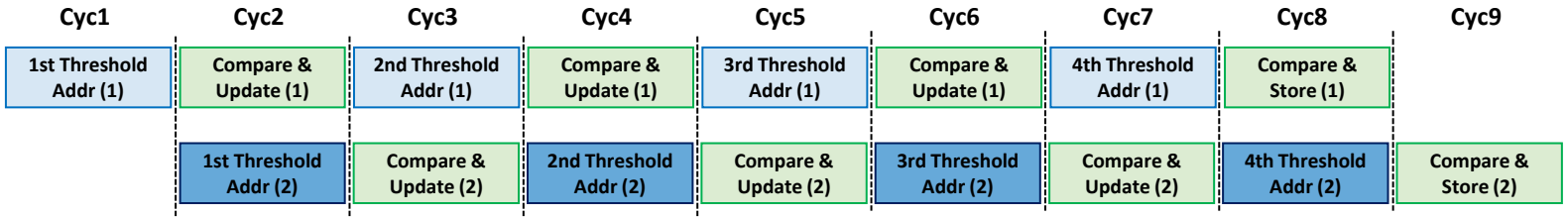
- Avoid branches;
- 9 cycles / 5 cycles quantize **two** 4-/ 2-bit ofmap activations.

HW µArchitecture: Quantization Unit



- Two activations compressed in parallel;
- Entry point of the second binary tree accessible from the first one with an hard-wired offset;
- A FSM orchestrates the interleaved execution;
- Timing and power-aware design;

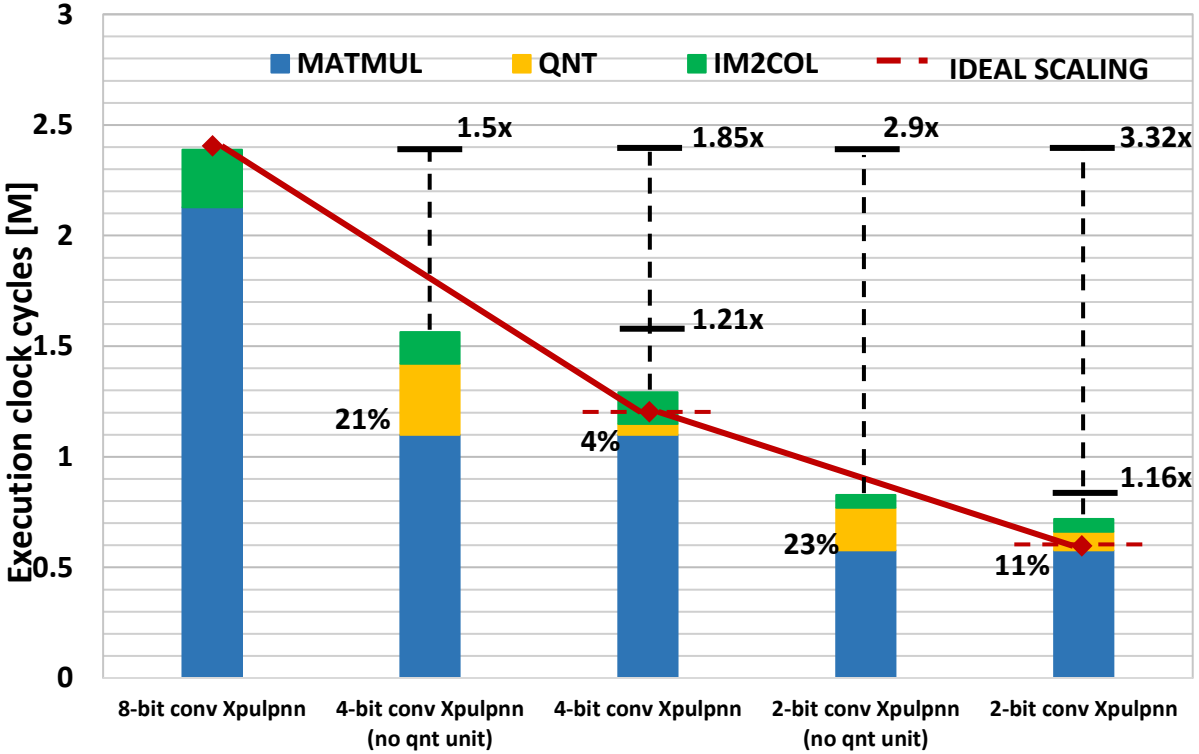
Pipeline Execution Diagram for nibble operands



Outline

- Introduction
- Background
  - RI5CY Core
  - QNN Execution Model
- Contribution of the Work
  - XpulpNN ISA Extensions
  - µArchitecture
- **Results**
  - Functional Performance
  - Physical Implementation
  - Benchmarking
- Conclusion

Results: Functional Performance



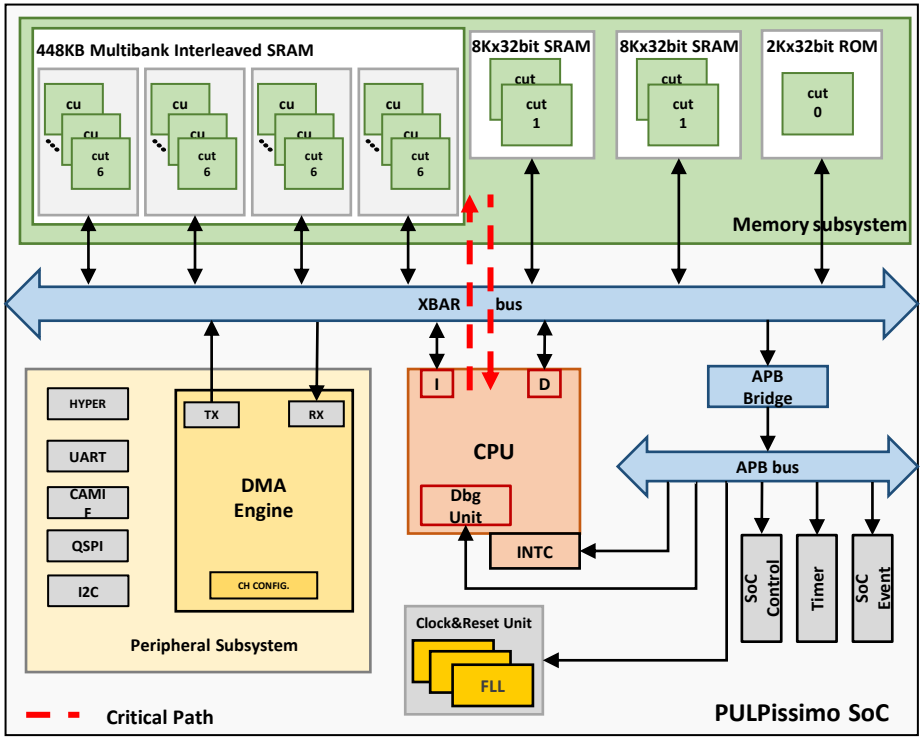
QNN layer benchmarked :

- Ifmap**:** *16x16x32* (HWC);
- 3-D filters: *64 – 3x3x32*(HWC);
- Ofmap: *16x16x64* (HWC);
- Bit-width: *8-, 4-, 2-bit*.

GCC compiler back-end :

- extended with machine description of the XpulpNN instructions.

Results: Physical Implementation Setup



PULPissimo: https://github.com/pulp-platform/pulpissimo.git

- CPU: RI5CY/ Extended RI5CY;
- 512 KB of SRAM;
- uDMA and full set of peripherals.

Synthesis and Place & Route (P&R):

- 22nm FD-SOI;
- 250 MHz;
- SS, 0.59 V, -40°C / 125 °C.

Power Analysis:

- Post P&R power simulation in the typical corner (TT, 0.65 V, 250 MHz).

Results: Physical Implementation

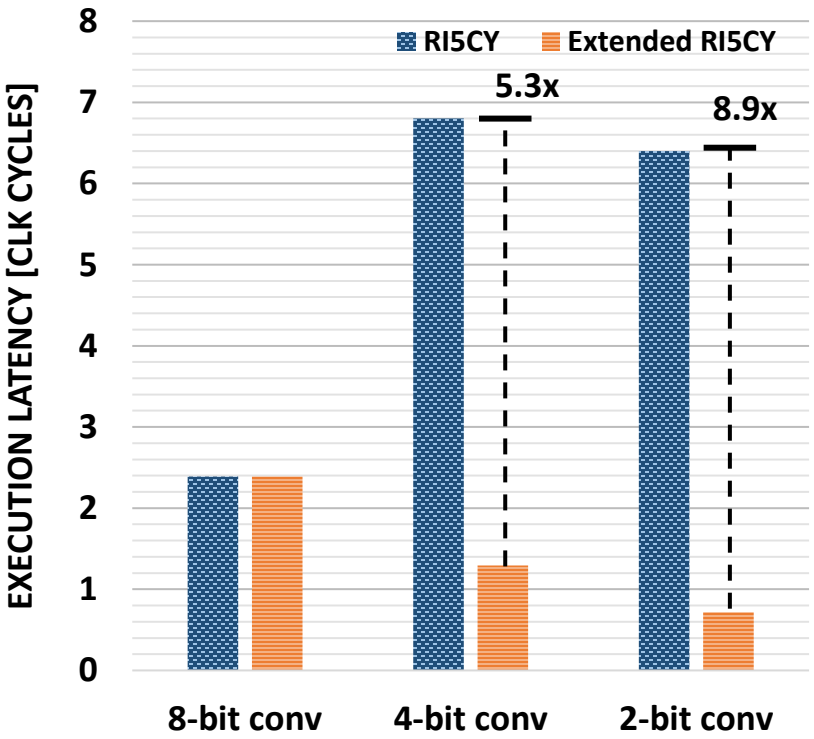
Area [μm^2] (Overhead vs. baseline [%])			
	RI5CY [4]	Ext. RI5CY No Pow. Manag.	Ext. RI5CY Pow. Manag.
Total	19729.9	21424.9 (8.59%)	21912.8 (11.1%)
dotp-Unit	5708.9	6755.8 (18.3%)	6844.4 (19.9%)
ID Stage	6363.1	6530.2 (1%)	6677.8 (5%)
EX Stage	9500.9	11129.1 (17.1%)	11251.6 (18.4%)
LSU	518.0	610.8 (17.9%)	591.2 (14.1%)
Core Power Consumption on 8-bit MatMul at 0.75V, 250MHz [mW]			
Leak. Power	0.023	0.032	0.031
Dyn. Power	1.13	1.38	1.19
Tot. Power	1.15	1.41	1.22
Overhead [%]	–	22.5%	5.9%
PM Savings [%]	–	–	13.5%
PULPissimo SoC Total Power Consumption at 0.75V, 250MHz [mW]			
8-bit MatMul	5.93	6.28 (5.8%)	6.04 (1.8%)
4-bit MatMul	–	8.14	5.71
2-bit MatMul	–	8.99	5.87
GP application	5.65	8.20 (45.2%)	5.85 (3.5%)

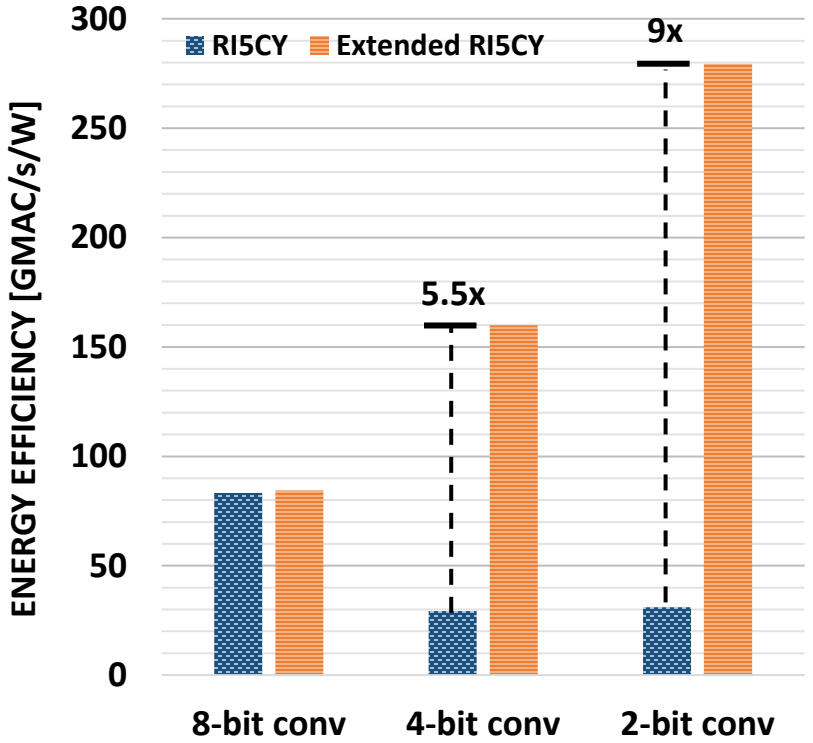
Small overhead due to the extended dotp unit and the addition of the quantization unit in the EX-stage of the core.

Performing operand isolation on critical operands highly reduces the Core total power consumption.

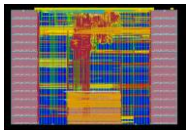
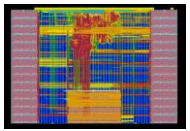
The proposed ISA extensions and µArchitecture do not jeopardize the efficiency of the system on GP apps.

Benchmarking: RI5CY vs. Extended RI5CY



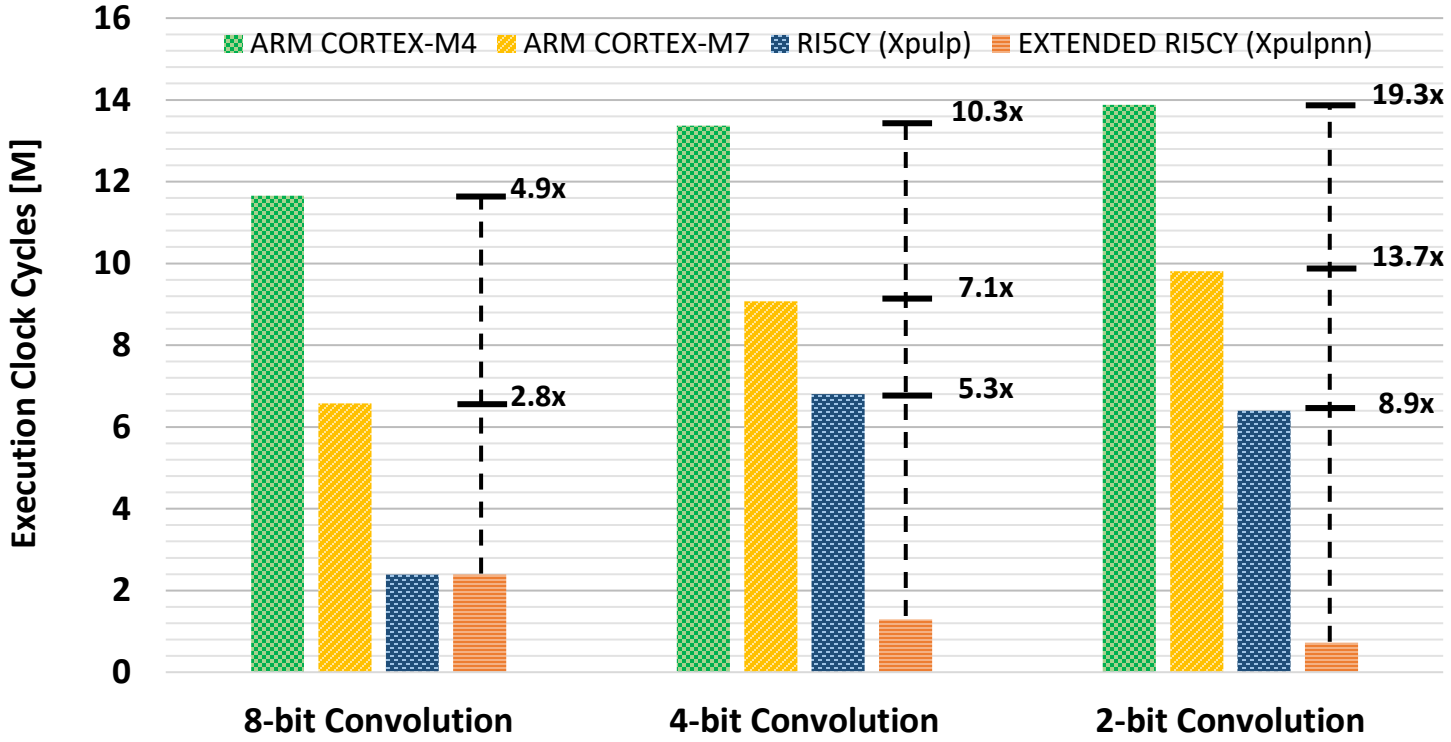


Benchmarking: Setup

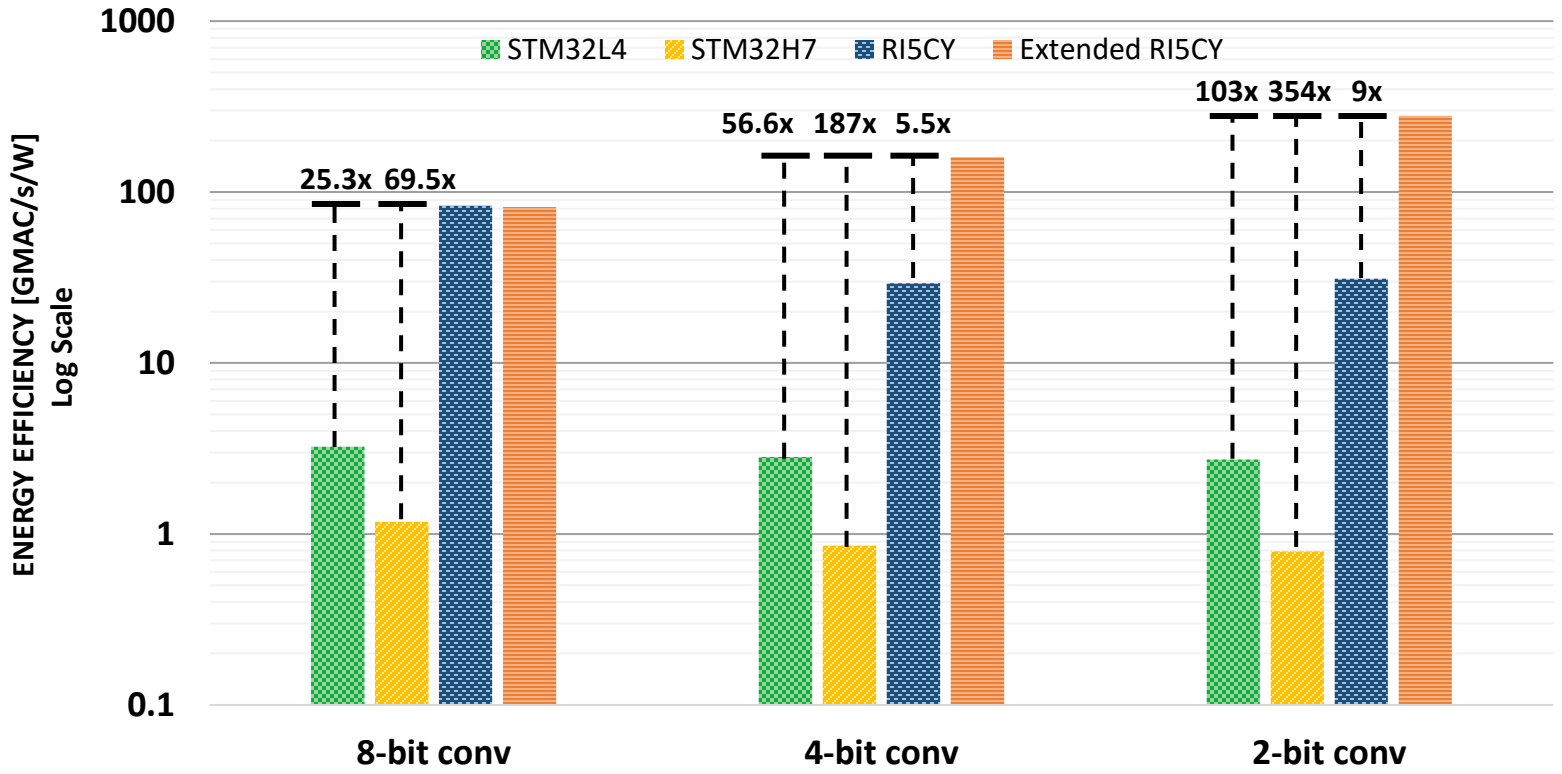
Device	STM32L476	STM32H743	PULPissimo	PULPissimo (This Work)
Core	ARM Cortex-M4	ARM Cortex-M7 (dual issue)	RI5CY Core	Extended RI5CY Core
ISA	ARM v8-m ISA	ARM v8-m ISA	RV32IMCXpulpV2	RV32IMCXpulpNN
				
Operating Point	**10mW@80MHz Phys. measurement Tech: 90nm**	**234mW@480MHz Phys. measurement Tech: 40nm**	**5.65mW@250MHz Post P&R power sim. Tech: 22nm**	**5.85mW@250MHz Post P&R power sim. Tech: 22nm**
SW	CMSIS-NN (*)	CMSIS-NN (*)	PULP-NN (**)	Extended PULP-NN

(*): Lai, Liangzhen et al. , "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus." arXiv preprint arXiv:1801.06601 (2018).
(**): Garofalo et al. "PULP-NN: accelerating quantized neural networks on parallel ultra-low-power RISC-V processors." Phil. Trans. of the Royal Society A 378, no. 2164 (2020): 20190155.

Benchmarking: SoA Comparison - 1



Benchmarking: SoA Comparison - 2



Outline

- Introduction
- Background
  - RI5CY Core
  - QNN Execution Model
- Contribution of the Work
  - XpulpNN ISA Extensions
  - µArchitecture
- Results
  - Functional Performance
  - Physical Implementation
  - Benchmarking
- **Conclusion**

Conclusion

- XpulpNN ISA Extensions to boost the Energy-Efficiency of low-bitwidth QNNs on MCU-class devices;
- Timing and power-aware design of the Extended RI5CY core;
- Full implementation of PULPissimo with the Extended RI5CY core in 22nm FDX technology;
- Low area and power overhead w.r.t. RI5CY; improved efficiency by ~10x on sub-byte QNN kernels;
- Two orders of magnitude better Energy-Efficiency with respect to SoA solutions.

Future Perspective

- Integration of XpulpNN in PULP platform (*) for Energy-Efficient QNN Parallel Computing;
- What about accelerating Mixed-Precision QNN?

(*): https://github.com/pulp-platform/pulp.git