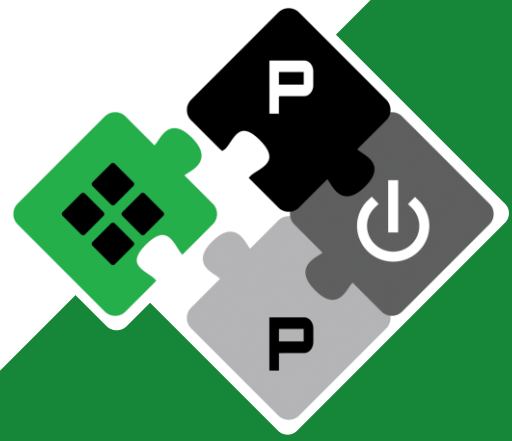
ETH zürich ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Deeploy: Enabling Energy-Efficient Deployment of Small Language Models On Heterogeneous Microcontrollers

Integrated Systems Laboratory (ETH Zürich)

Moritz Scherer scheremo@iis.ee.ethz.ch

Luka Macan, Victor Jung, Philip Wiese, Luca Bompani,
Alessio Burrello, Francesco Conti, Luca Benini



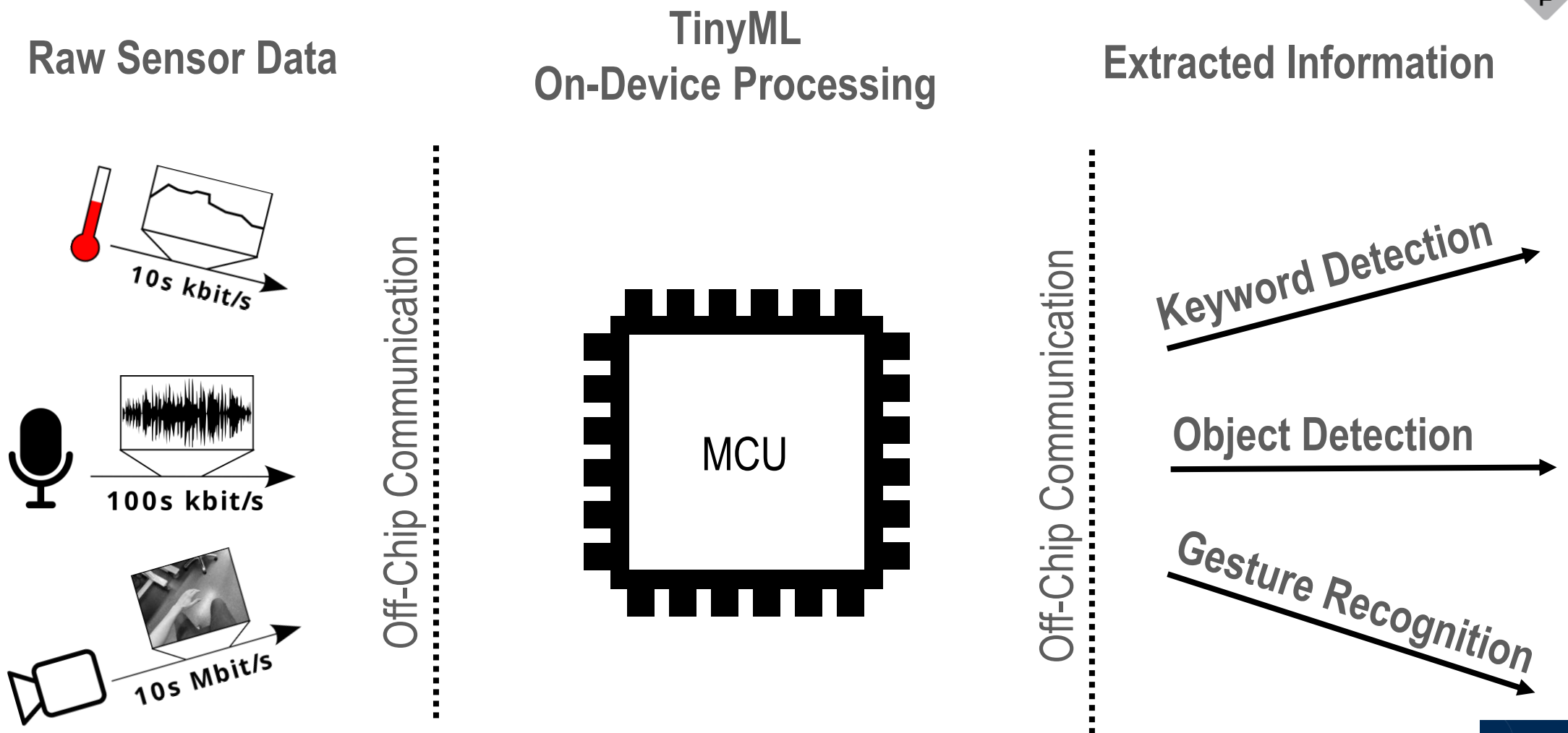
@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

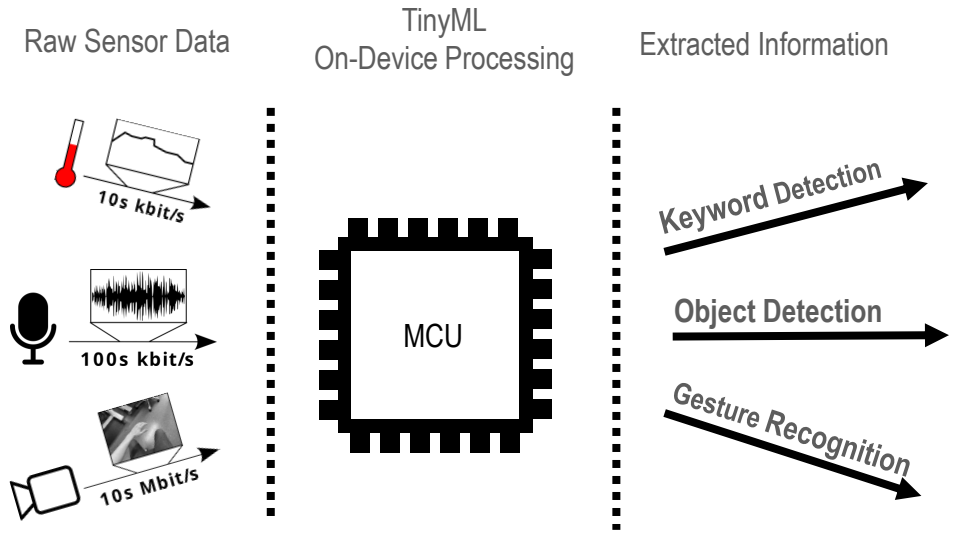
PULP Platform
Open Source Hardware, the way it should be!

Today's TinyML Application Landscape



Towards Multi-modal TinyML

- Today's TinyML is mainly a compression tool
  - Extract and transmit compressed information
  - **Sensor control and actuation logic is hand-crafted**



- Efficient and effective sensor control is hard
  - Dynamic noise conditions
  - **Multi-modal dependencies – your audio input might affect your camera control**

- **Need to close the on-device control loop**
  - Using multi-modal, context-aware models

The TinyML Compiler Gap

- Multi-modal models trend towards foundation models (FMs)
  - Large transformer models (Billions of parameters)
  - **Transformer models are good at multi-modal input processing**
  - **Small Language Models (SLMs) show promise as application-specific FMs**

- TinyML uses heterogeneous MCUs
  - Software-managed caches, few MBs of memory
  - Application-specific accelerators

tvm

ONNX MLIR

- **No turnkey solution for "truly TinyML" heterogeneous MCUs**
  - Options for application-scale devices (PCs, Servers, …)
  - MCU-scale toolchains come with vendor-lock or deployment limitations

STM32 Cube.AI

TensorFlow Lite

EMBEDDED SYSTEMS WEEK

Motivation & Contribution

- **How do we close the TinyML Compiler Gap for small FMs?**

- We introduce Deeploy, a bottom-up DNN compiler for heterogeneous MCUs
  - **Novel tiling & allocation algorithm implements spill-free network execution**
  - Vendor-agnostic backend, with extensible expert-optimized low-level kernels
- Deeploy achieves **SoA results on the MLPerf Tiny benchmark**
- Deployed an SLM on Siracusa, a heterogeneous TinyML MCU
  - **Achieving 340 Tokens/sec @ 490 uJ/Token**
  - Using on-chip memory only

An End-to-end Edge AI Stack

Training & Quantization

TensorFlow

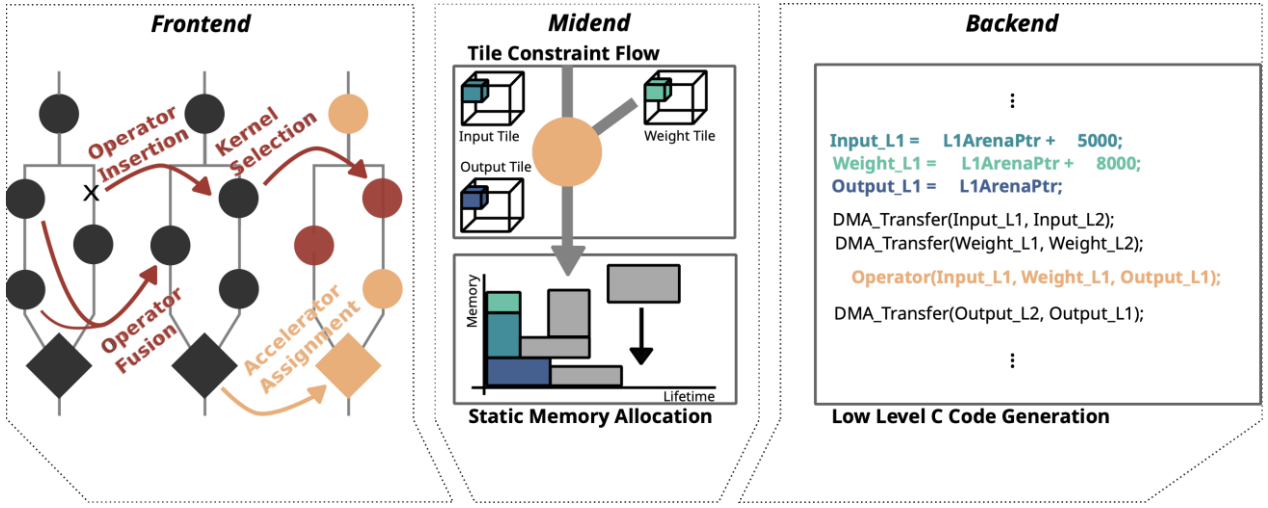
PyTorch

Keras

Model Export

ONNX

Deeploy

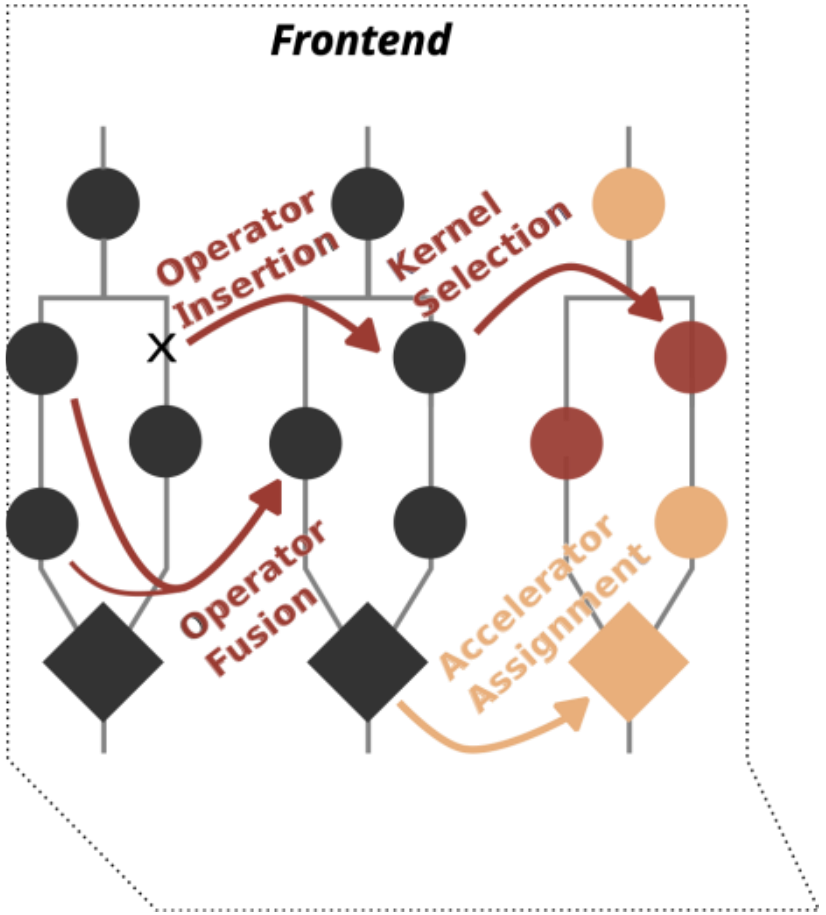


Inference

RISC-V

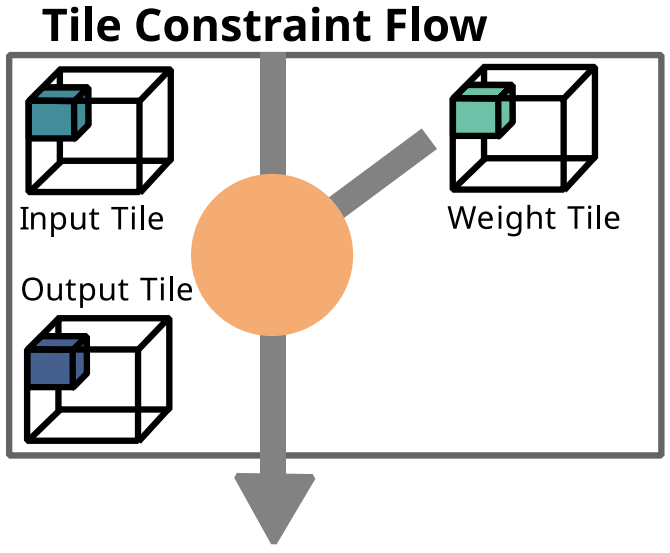
arm

Deeploy's Frontend – Engine-aware lowering



- **Graph lowering is platform-dependent**
  - Taking into account engine mapping options
  - Graph remains fully ONNX compliant
- **Nodes are matched with low-level kernels**
  - Kernel libraries like CMSIS-NN, PULP-NN are supported
  - Custom kernels may be added as well
- Execution schedule is based on heuristics
  - Optionally defined by the user
- How do we manage on-chip memory?

Deeploy's Midend – Tiling & Memory Allocation



Core Idea: Control on-chip memory use

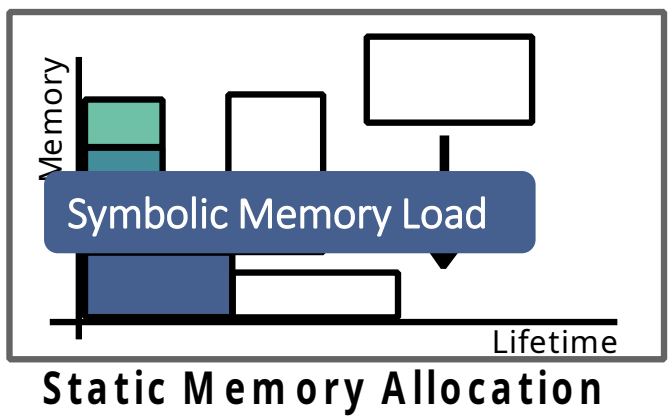
Express tiling constraints in a single Integer Linear Program (ILP)

Operator Constraints:

Constrain partial inputs to produce valid partial outputs

Target Constraints:

Constrain partial inputs to work with HW target



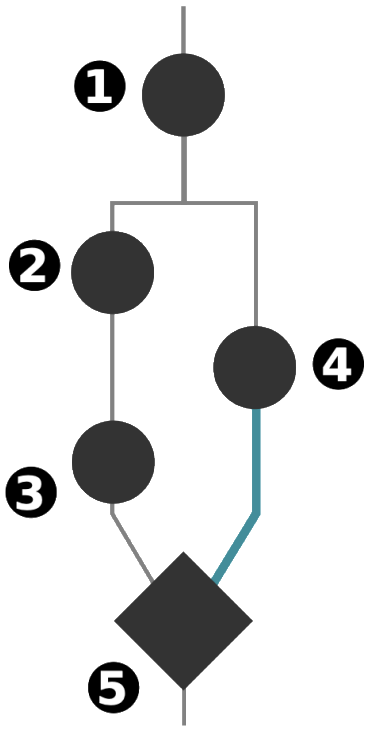
Core Idea: Schedule allocation offline to fit into on-chip memory

Symbolically calculate the maximum memory load

< Available On-Chip Memory

Solve resulting ILP for joint tiling & allocation solution!

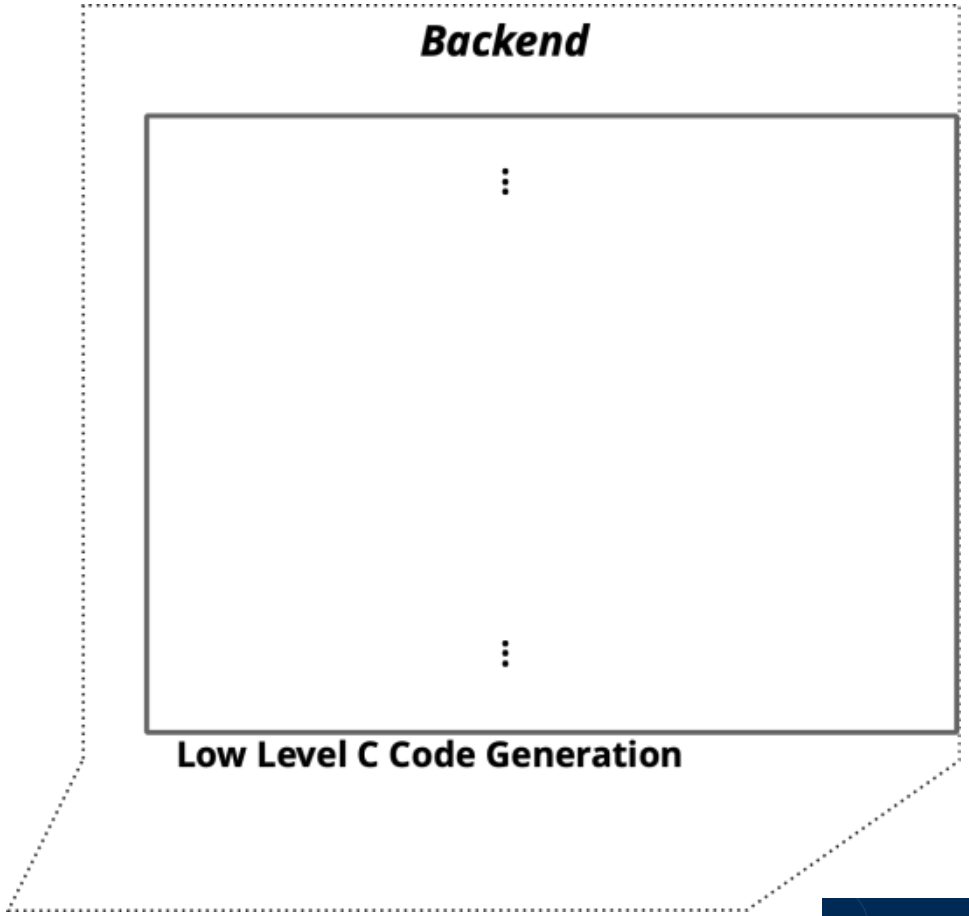
Deeploy's Midend – Optimizing Memory Allocation



Static allocation guarantees spill-free network execution

Deeploy Backend – Vendor agnostic code generation

- **Backend generates low-level C Code**
  - Tiling & memory allocation from midend
  - Using operator mapping from frontend
- Deeploy provides code generation primitives
  - Device offloading
  - Double-buffered DMA transfers
  - Fork-synch based multi-core programming
- **Bottom-up code generation accelerates reuse**
  - Bring your own expert-optimized kernel templates!



Putting Deeploy into Practice - Siracusa

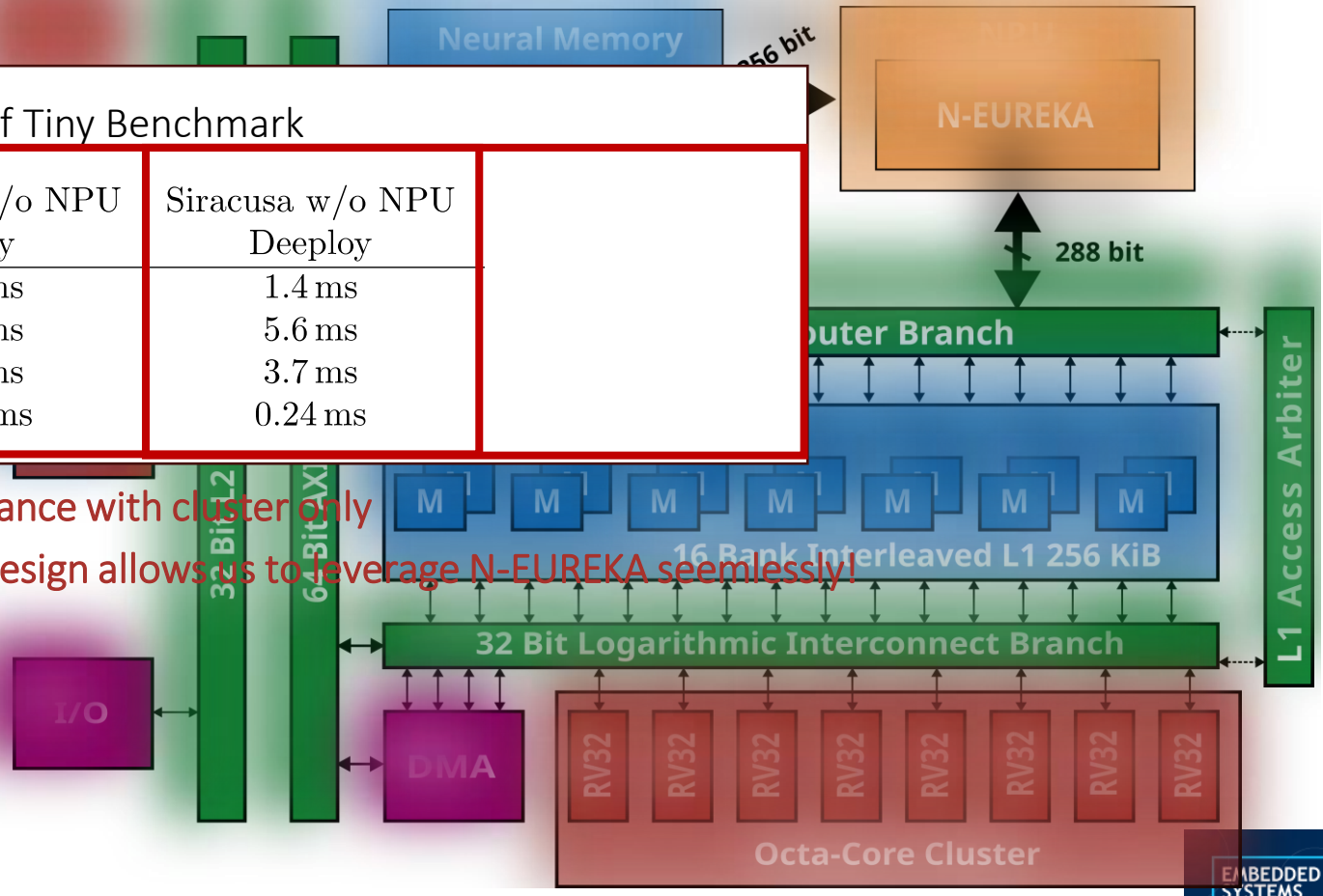
- Target Platform: Siracusa
  - 8-core PULP C
  - 2 MiB L2 Mem
  - 256 KiB L1 Me
- N-EUREKA CNN A
  - 3x3 Convolutio
  - 1x1 Convolution
  - Shared L1 access
- Neural Memory Subsystem
  - Weight memory for N-EUREKA

MLPerf Tiny Benchmark

Benchmark	Siracusa w/o NPU Dory	Siracusa w/o NPU Deeploy
DS-CNN	1.4 ms	1.4 ms
MobileNetv1	5.6 ms	5.6 ms
ResNet	3.7 ms	3.7 ms
ToyAdmos	0.24 ms	0.24 ms

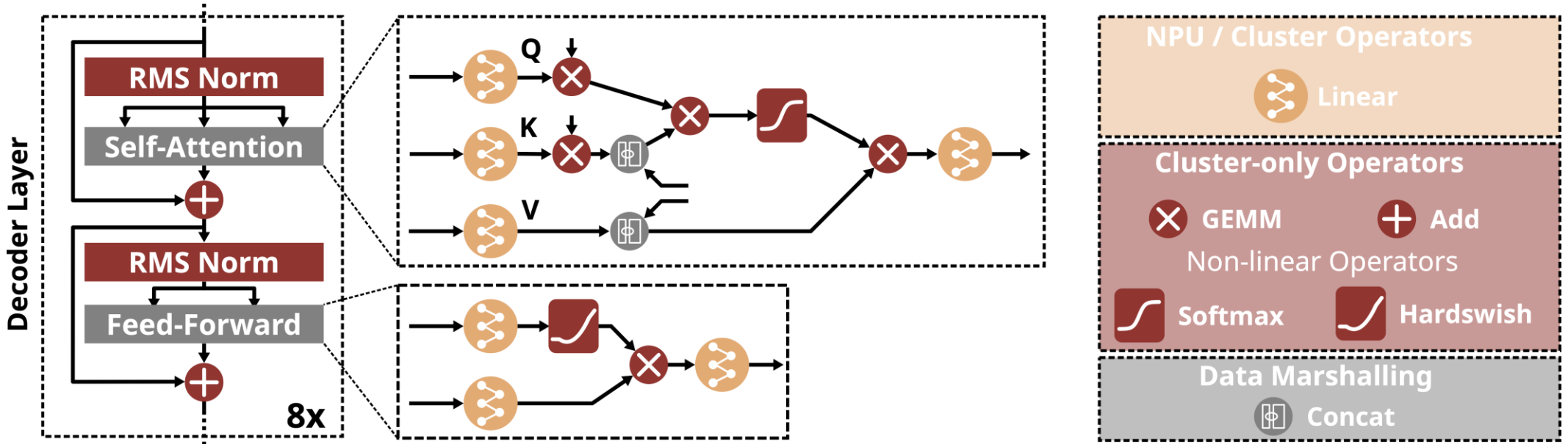
SoA Performance with cluster only

Bottom-up design allows us to leverage N-EUREKA seemlessly!



Towards Small Foundation Models for TinyML

"MicroLlama" – 4.2 MParameter Llama model



- Reuse N-EUREKA's 1x1 convolution to map linear layers
  - Linear weights are kept untiled in weight memory
- Run all other operators on the cluster

Measured in-silicon Performance

340 Tokens per Second

490 µJ per Token

Comparing TinyML compilers

- Deeploy achieves SoA performance
  - Benchmark TinyML workloads
  - Emerging Small Foundation Models
- Deeploy supports your MCU platform
  - ARM, RISC-V, Accelerators, emerging memories, NUMA, …

Embedded Tool Name	Supports Transformers?	Supports Accelerators?	Supports memory-aware Tiling?	Open and extensible?
TensorFlow Lite	✗	✗	✗	✓
STM Cube AI	✗	✗	✗	✗
GreenWaves NNTool	✓	✓	✓	✗
Dory	✗	✗	✓	✓
Deeploy	✓	✓	✓	✓

The Last Word

- Deeploy allows leveraging heterogeneous MCUs for multimodal AI
  - Midend generates fully static memory allocation, including tiling
  - Front- and Backend capable of exploiting dedicated accelerators
- Demonstrated fully on-chip SLM inference on an MCU
  - Running entirely bare-metal, leveraging expert-optimized kernels
  - 340 Tokens/sec @ 490 uJ/Token
- **Everything's open-source!**
  - Quantlib for DNN Training & Quantization
  - Deeploy
  - PULP-NN & PULP-NNX kernel libraries
  - GVSoC Siracusa platform simulator

`github.com/pulp-platform/deeploy`