Many Shades of TinyML Acceleration
a RISC-V open platform approach

Luca Benini (lbenini@ethz.ch, luca.Benini@unibo.it)

PULP Platform
Open Source Hardware, the way it should be!
AI, ML Disruption: Technology cannot Keep the Pace

The drive to bigger AI models

The scale of artificial-intelligence neural networks is growing exponentially, as measured by the models' parameters (roughly, the number of connections between their neurons)*.

Energy Efficiency \(\frac{1}{\text{Power-Time}}\)

10x every 2 years

10x every 12 years...

*"Sparse" models, which have more than one trillion parameters, but use only a fraction of them in each computation, are not shown.
Necessity is the Mother of Invention
The Renaissance of Design

Gains from

- Number Representation
  - FP32, FP16, Int8
  - (TF32, BF16)
  - ~16x

- Complex Instructions
  - DP4, HMMA, IMMA
  - ~12.5x

- Process
  - 28nm, 16nm, 7nm, 5nm
  - ~2.5x

- Sparsity
  - ~2x

- Model efficiency has also improved – overall gain > 1000x
Energy-Efficient Computing: Core to Platform

- **Core**
  - Improving core efficiency with ISA and uAch extensions

- **Cluster**
  - Efficient shared-mem cluster
  - From a few to thousand processing elements

- **Full platform**
  - Heterogeneity: host, processor, accelerator(s)
  - IOs, main memory
  - Chips → chiplets → stacks (2D to 3D)
<table>
<thead>
<tr>
<th>Year</th>
<th>Process</th>
<th>Company</th>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>STM 28FD</td>
<td>PULPv1</td>
<td>v1</td>
<td>STM 28FDSOI Multi-core processor</td>
</tr>
<tr>
<td>2014</td>
<td>UMC 65</td>
<td>Diana</td>
<td>UMC</td>
<td>4-core system with approximate FPUs</td>
</tr>
<tr>
<td>2015</td>
<td>UMC 65</td>
<td>Fulmine</td>
<td>UMC</td>
<td>4-core system with ML and Crypto accelerators</td>
</tr>
<tr>
<td>2016</td>
<td>SMIC 130</td>
<td>VivoSoC</td>
<td>2.001</td>
<td>Mixed signal system for biosignal acquisition</td>
</tr>
<tr>
<td>2017</td>
<td>TSMC 40</td>
<td>Mr. Wolf</td>
<td></td>
<td>8+1 core IoT processor</td>
</tr>
<tr>
<td>2018</td>
<td>GF 22FDX</td>
<td>Poseidon</td>
<td></td>
<td>Dual 64bit RISC-V core, 32bit Microcontroller system, ML accelerator</td>
</tr>
<tr>
<td>2019</td>
<td>TSMC 65</td>
<td>Baikonur</td>
<td></td>
<td>IoT processor with 16 cores and QNN enhancements</td>
</tr>
<tr>
<td>2020</td>
<td>GF 22FDX</td>
<td>Dustin</td>
<td></td>
<td>IoT processor with Spiking Neural and Ternary Inference Engines</td>
</tr>
<tr>
<td>2021</td>
<td>GF 12LPP</td>
<td>Kraken</td>
<td></td>
<td>ML accelerator with 216 + 1 cores and HBM interface</td>
</tr>
<tr>
<td>2022</td>
<td>GF 12LPP</td>
<td>Occamy</td>
<td></td>
<td>ML accelerator with 216 + 1 cores and HBM interface</td>
</tr>
</tbody>
</table>
Edge ML Market

**TinyML challenge**

**AI capabilities in the power envelope of an MCU: 10-mW peak (1mW avg)**
The Challenge: Energy efficiency@GOPS

ARM Cortex-M MCUs: M0+, M4, M7 (40LP, typ, 1.1V)*

High performance MCUs

*data from ARM’s web
High-Performance vs. Energy-Efficient

“In classical” core performance scaling trajectory

- Faster CLK → deeper pipeline → IPC drops
- Recover IPC → superscalar → ILP bottleneck (dependencies)
- Mitigate ILP bottlenecks → OOO → huge power, area cost!

[Azizi et al. ISCA10]
A way Out: Processor Specialization

A way Out: Processor Specialization

3-cycle ALU-OP, 4-cyle MEM-OP ➔ only IPC loss: LD-use, Branch

ISA is extensible by construction (great!)

V1 Baseline RV (not good for ML)
Extensions for Data Processing

V2 Data motion (e.g. auto-increment)
Data processing (e.g. MAC)

V3 Domain specific data processing
Narrow bitwidth
HW support for special arithmetic

ISA extension cost 25 kGE ➔ 40 kGE (1.6x), energy efficient if 0.6Texec

[Gautschi et al. TVLSI 2017]
Achieving 100% dotp Unit Utilization

8-bit Convolution

<table>
<thead>
<tr>
<th>RV32IMC</th>
<th>N/4</th>
<th>RV32IMCXpulp</th>
</tr>
</thead>
<tbody>
<tr>
<td>addi  a0,a0,1</td>
<td></td>
<td>pv.nnsdot.h</td>
</tr>
<tr>
<td>addi  t1,t1,1</td>
<td></td>
<td>pv.nnsdot.b</td>
</tr>
<tr>
<td>addi  t3,t3,1</td>
<td></td>
<td>pv.nnsdot.b</td>
</tr>
<tr>
<td>addi  t4,t4,1</td>
<td></td>
<td>pv.nnsdot.b</td>
</tr>
<tr>
<td>lbu   a7,-1(a0)</td>
<td></td>
<td>pv.nnsdot.b</td>
</tr>
<tr>
<td>lbu   a6,-1(t4)</td>
<td></td>
<td>end</td>
</tr>
<tr>
<td>lbu   a5,-1(t3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lbu   t5,-1(t1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mul   s1,a7,a6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mul   a7,a7,a5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>add   s0,s0,s1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mul   a6,a6,t5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>add   t0,t0,a7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mul   a5,a5,t5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>add   t2,t2,a6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>add   t6,t6,a5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bne   s5,a0,1c000bc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8-bit SIMD sdotp

<table>
<thead>
<tr>
<th>N</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>addi  a0,a0,1</td>
<td>a1</td>
</tr>
<tr>
<td>addi  t1,t1,1</td>
<td>t2</td>
</tr>
<tr>
<td>addi  t3,t3,1</td>
<td>t4</td>
</tr>
<tr>
<td>addi  t4,t4,1</td>
<td>t3</td>
</tr>
<tr>
<td>lbu   a7,-1(a0)</td>
<td>a5</td>
</tr>
<tr>
<td>lbu   a6,-1(t4)</td>
<td>a6</td>
</tr>
<tr>
<td>lbu   a5,-1(t3)</td>
<td>t1</td>
</tr>
<tr>
<td>lbu   t5,-1(t1)</td>
<td>a7</td>
</tr>
<tr>
<td>mul   s1,a7,a6</td>
<td>a3</td>
</tr>
<tr>
<td>mul   a7,a7,a5</td>
<td>a2</td>
</tr>
<tr>
<td>add   s0,s0,s1</td>
<td>a1</td>
</tr>
<tr>
<td>mul   a6,a6,t5</td>
<td>a5</td>
</tr>
<tr>
<td>add   t0,t0,a7</td>
<td>a7</td>
</tr>
<tr>
<td>mul   a5,a5,t5</td>
<td>a6</td>
</tr>
<tr>
<td>add   t2,t2,a6</td>
<td>a7</td>
</tr>
<tr>
<td>add   t6,t6,a5</td>
<td>a5</td>
</tr>
<tr>
<td>bne   s5,a0,1c000bc</td>
<td>a0</td>
</tr>
</tbody>
</table>

HW Loop

8-bit sdotp + LD

<table>
<thead>
<tr>
<th>N</th>
<th></th>
</tr>
</thead>
</table>

N/4

11

8-bit Convolution

8-bit SIMD sdotp

8-bit sdotp + LD

9x less instructions than RV32IMC

Yes! dotp+ld

Init NN-RF (outside of the loop)

lp.setup

pv.nnsdot.h s0,ax1,9

pv.nnsdot.b s1,aw2,0

pv.nnsdot.b s2,aw4,2

pv.nnsdot.b s3,aw3,4

pv.nnsdot.b s4,ax1,14

end

14.5x less instructions at an extra 3% area cost (~600GEs)

ETH Zürich

11
Hardware for \texttt{dotp+ld}

NN RF: 6 32-bit regs (weights and input activations)

Special-purpose registers

[A. Garofalo et al., TEC21]
Mixed Precision SIMD Processor

- Can support all variants:
  - 16x16, 16x8, 16x4, 16x2
  - 8x8, 8x4, 8x2
  - 4x4, 4x2
  - 2x2

- Avoids Pack/unpack Overheads
- Maximizes performance (SIMD)
- Maximizes RF use (Data Locality)

How to encode all these instructions?
Virtual SIMD Instructions

- Encode operation as a virtual SIMD in the ISA (e.g. sdotsp.v)
- Format specified at runtime by a Control Register (e.g. 4x4)
- **180 → 18** Instructions needed for SIMD DOTP
- Potential to avoid code replication for different formats
- Tiny Overhead on QNN for Switching format
  - Format switch not frequent in DNN, e.g. every layer.
**Scaling performance: Parallel, Ultra-Low Power (PULP)**

- As VDD decreases, operating speed decreases.
- However, efficiency increases → more work done per Joule.
- Until leakage effects start to dominate.
- Put more units in parallel to get performance up and keep them busy with a parallel workload.

ML is massively parallel and scales well (P/S ↑ with NN size).

---

![Graph showing efficiency vs VDD](attachment:graph.png)

- **Optimum point**: Better to have $N \times$ PEs running at optimum Energy than 1 PE running fast at low Energy efficiency.

---

[Rossi et al. IEEE Micro 2017]
Multiple RI5CY Cores (1-16)
Low-Latency Shared TCDM

- Parallel memory access with low contention
  - Multi-banked, address-interleaved L1

- Fast interconnect with physical design awareness
  - Logarithmic depth of combinational switchboxes

![Diagram](image-url)
Fast synchronization, non-blocking DMA L1-L2 copies

Tightly Coupled Data Memory BF=2

Mem0 Mem4 Mem5 Mem6 Mem7
Mem1 Mem2 Mem3 Mem4 Mem5
Mem0 Mem1 Mem2 Mem3 Mem4

CLUSTER

~15x latency and energy reduction for a barrier

[Glaser TPDS20]
- Two-level I$\$:
  - Private (P) + Shared (S)
- Most IFs from I$\$-P:
  - Low IF energy
- I$\$-S for capacity:
  - Reduces miss latency
Host for sequential, I/O + Data-Parallel Cluster

Open sourced since 2017: github.com/pulp-platform/pulp
Combining ISA extension + Efficient parallel execution

- 8-bit convolution
  - Open source DNN library
- 10x through xPULP
  - Extensions bring real speedup
- Near-linear speedup
  - Scales well for regular workloads
- 75x overall gain
- 7-8 GMACs
  - 250MHz
  - 4 MAC/Cycle (8bit)
  - 8 Cores

Overall Speedup of 75x

Near-Linear Speedup

10x Speedup w.r.t. RV32IMC
(ISA does matter😊)

[Garofalo et al. Philos. Trans. R. Soc 20]
8-Cores Cluster + ISA (FDX22nm)

- STM32L4 (M4)
- STM32H7 (M7)
- PULP (RI5CY) 0.65V
- PULP (RI5CY) 0.8V
- PULP (XpulpNN + m&l) 0.65V
- PULP (XpulpNN + m&l) 0.8V

**ENERGY EFFICIENCY [TOPS/W] Log scale**

- 8-bit convolution: 146x, 1.6x, 401x
- 4-bit convolution: 294x, 1600x
- 2-bit convolution: 356x, 7.4x, 1230x

[Garofalo et al. OJSSCS22]

More GOPS, Less Power?
What’s next? Tightly-coupled HW Compute Engine

Acceleration with flexibility: zero-copy HW-SW cooperation
Hardware Processing Engines (HWPEs)

HWPE efficiency \( \frac{MAC}{A(mm^2)E(J)W(bps)} \) vs. optimized RISC-V core

1. Dedicated control (no I-fetch) with shadow registers (overlapped config-exec)
2. Specialized high-BW interco into L1 (on data-plane)
3. Specialized datapath: supporting configurable & aggressive quantization
Reconfigurable Binary Engine

\[ y(k_{out}) = \text{quant} \left( \sum_{i=0}^{N} \sum_{j=0}^{M} 2^{i} 2^{j} (W_{\text{bin}}(k_{out}, k_{in}) \otimes x_{\text{bin}}(k_{in})) \right) \]

Energy efficiency 10-20x (0.1pJ/OP) wrt to SW on cluster @same accuracy
All together in VEGA: Extreme Edge IoT Processor

- RISC-V cluster (8 cores +1)
  614 GOPS/W @ 7.6 GOPS (8 bit DNNs),
  79 GFLOPS/W @ 1 GFLOP (32 bit FP appl)
- Multi-precision HWCE(4b/8b/16b)
  3×3×3 MACs with normalization / activation: 32.2 GOPS and 1.3 TOPS/W (8 bit)
- 1.7 μW cognitive unit for autonomous wake-up from retentive sleep mode
All together in VEGA: Extreme Edge IoT Processor

- RISC-V cluster (8 cores +1)
  614GOPS/W @ 7.6GOPS (8bit DNNs),
  79GFLOPS/W @ 1GFLOP (32bit FP appl)
- Multi-precision HWCE(4b/8b/16b)
  3×3×3 MACs with normalization / activation: 32.2GOPS and 1.3TOPS/W (8bit)
- 1.7 µW cognitive unit for autonomous wake-up from retentive sleep mode
- **Fully-on chip DNN inference with 4MB MRAM (high-density NVM with good scaling)**

In cooperation with [D. Rossi, ISSCC21]

<table>
<thead>
<tr>
<th>Technology</th>
<th>22nm FDSOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip Area</td>
<td>12mm²</td>
</tr>
<tr>
<td>SRAM</td>
<td>1.7 MB</td>
</tr>
<tr>
<td>MRAM</td>
<td>4 MB</td>
</tr>
<tr>
<td>VDD range</td>
<td>0.5V - 0.8V</td>
</tr>
<tr>
<td>VBB range</td>
<td>0V - 1.1V</td>
</tr>
<tr>
<td>Fr. Range</td>
<td>32 kHz - 450 MHz</td>
</tr>
<tr>
<td>Pow. Range</td>
<td>1.7 µW - 49.4 mW</td>
</tr>
</tbody>
</table>
Full DNN Energy (MobileNetV2) on Vega

Bandwidth [MB/s]

Energy per byte [pJ/B]

end-to-end on-chip computation

weights on MRAM

weights on HyperRAM

3.5x less energy

1.19 mJ

4.16 mJ

Energy per Inference [mJ]
Respectively 85% and 65% of GAP8 and GAP9 are based on open-source IPs
RISC-V (PULP) Currently dominates the TinyML benchmarks
Extreme Edge Use Case: Nano-Drones

Advanced autonomous drone

- 3D Mapping & Motion Planning
- Object recognition & Avoidance
- 0.06m² & 800g of weight
- Battery Capacity 5410mAh

Nano-drone
https://www.bitcraze.io/products/crazyflie-2-1

- Smaller form factor of 0.008m²
- Weight 27g (30X lighter)
- Battery capacity 250mAh (20X smaller)

Can we fit sufficient intelligence in a 30X smaller payload, 20X lower energy budget?
Achieving True Autonomy on Nano-UAVs

Multiple, complex, heterogeneous tasks at high speed and robustness fully on board

Obstacle avoidance & Navigation

Environment exploration

Object detection

Multi-GOPS workload at extreme efficiency $\rightarrow P_{\text{max}} 100\text{mW}$
Monte Carlo Localization (Known Map)

Particle filter-based

Convergence + Low ATE for $\text{N}_{\text{part}} > 1024$, 2ToF, FP16 acceptable

- 12MHz, 1Kpart. 13mW, 60msec
- 400MHz, 1Kpart 61mW, 1msec
- 400MHz 16Kpart 61mW, 30msec
What’s next? Multiple Heterogeneous Accelerators

*Brain-inspired*: Multiple areas, different structure different function!
What’s next? Multiple Heterogeneous Accelerators

The **Kraken**: an "Extreme Edge" Brain

- RISC-V Cluster (8 Cores + 1)
- **CUTIE** – dense ternary neural network accelerator
- SNE – energy-proportional spiking neural network accelerator
- PULPO – Floating point online optimizer (advanced control, state estimation...)

\[
\begin{align*}
\mathbf{z} &= \mathbf{A}\mathbf{x} + \mathbf{y} \quad (\mathcal{M}) \\
\mathbf{z} &= \mathbf{y} - \tau \mathbf{A}^H \mathbf{x} \quad (\mathcal{H}) \\
\mathbf{z} &= \text{prox}(\rho \mathbf{x}) \quad (\mathcal{P})
\end{align*}
\]

[Di Mauro HotChips22]
KxK window on all input channels unrolled, cycle-by-cycle sliding

Completely unrolled inner products one output activation per cycle!

Zeros in weights and activations, spatial smoothness of activations reduce switching activity

96 OCUs, 96 Input channels, 3x3 kernels: 96 * 96 * 3 * 3 = 82'944 TMAC/cycle

CUTIE: Minimize Switching Activity & Data Movement

[Scherer et al. TCAD22]

Aggressive quantization and full specialization
Different Sensor Type, different Acceleration Engine

- **SNE Engine**: 16 Adaptive-LIF neuron data paths (NG). A NG executes one Synaptic Operation (SOP) per cycle
  - $1 \text{ SOP} = 1 \text{ 4b-ADD} + 2 \text{ 8b-MUL} + 1 \text{ 8b-ADD} + 1 \text{ 8b-CMP}$
- For fully connected layers one NG is time-shared for 64 virtual neurons
- Optimized buffering and neuron state update for 64x16 neurons in just 12 cycles for a 3x3 event receptive field
  - Equivalent number of 85 SOP/cycle per engine (682 SOP/cycle on 8 engines)

**Event Sensors**: DVS
- Ultra-low latency
- Energy-proportional interface

[Dimauro et al. DATE22]

**Weight memory (~1.1kB)**
- 256 slots of 9 4bits weights

**State memory (4kB)**
- 64x16 32bits states

**Event router**
- Spike event in
- 16 NGs

**Spike event out**

SNE works seamlessly with DVS (event-based) sensors
Specialization in perspective

Using 22FDX tech, NT@0.6V, High utilization, minimal IO & overhead

Energy-Efficient RV Core $\rightarrow$ 20pJ (8bit) $\rightarrow$ XPULPV2 & V3

ISA-based 10-20x $\rightarrow$ 1-5pJ (8bit) $\rightarrow$ HWCE, RBE

Configurable DP 10-20x $\rightarrow$ 20-100fJ (4bit) $\rightarrow$ XNE, CUTIE*

Highly specialized DP 10-20x $\rightarrow$ 1-5fJ (ternary) $\rightarrow$ *

* sub 1fJ in 7nm
Advancing the SOA on all tasks

RISC-V Cluster
- Comparable 32bits-8bits SOA Energy efficiency to other PULPs [7]
- The highest energy efficiency on sub-byte SIMD operations (4b-2b)

SNE
- 1.7X higher than SOA [5] energy/efficiency

CUTIE
- 2X higher energy efficiency improvement over SOA [6]

CUTIE, SNE can work concurrently for SNN + TNN “fused” inference (never done so far)

Not only Efficiency: Achieving sub-mW Average Power?

1mW average power with 10mW active power (10GOPS @ 1pJ/OP) \(\rightarrow\) sub mW sleep

Duty cycling not acceptable when input events are asynchronous \(\rightarrow\) watchful Sleep

Log(P)

Detect&Compress \(\rightarrow\) 1-10mW

Watchful sleep \(\rightarrow\) <1mW
Need μW-range always-on Intelligence

PULPissimo

Smart Wakeup Module

HWCE

RISC-V core

I/O

L2 Mem

Mem Cont

Ext. Mem

Tightly Coupled Data Memory BF=2

Mem

Mem

Mem

Mem

Logarithmic Interconnect

DMA

HW SYNC

CLUSTER

41
HD-Based smart Wake-Up Module
HD-Based smart Wake-Up Module
HD-Based smart Wake-Up Module

- Tightly Coupled Data Memory BF=2
- Logarithmic Interconnect
- RISC-V core
- Mem
- DMA
- HW SYNC
- I/O
- Ext. Mem
- L2 Mem
- RISC-V core
- Preprocessor
- Autonomou
- HD-Computing Unit
- Always-on Domain
- Wake Up
- ADC
- Ext. ADC
- I/O (SPI)
Not Only CNNs: Hyper-Dimensional Computing

Mapping

\[
\begin{bmatrix}
0 & 1 & 0 & 1 & \ldots & 1 \\
1 & 1 & 1 & 0 & \ldots & 1 \\
1 & 1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 1 & 1 & \ldots & 1 \\
1 & 1 & 1 & 1 & \ldots & 1 \\
0 & 1 & 0 & 1 & \ldots & 1 \\
0 & 1 & 0 & 1 & \ldots & 1 \\
\end{bmatrix}
\]

Low Dimensional Input Data (e.g. 7-bit LBP)

HD-Encoding

- Component-wise Majority
- XOR
- Permutation

Search Vector

[1 1 0 1 \ldots 1]

Similarity Search (e.g. Hamming Distance)

Prototype Vectors

[0 1 0 1 \ldots 1]

[1 1 1 0 \ldots 1]

[1 1 0 0 \ldots 0]

[0 1 1 1 \ldots 1]

[1 1 1 1 \ldots 1]

[0 1 0 1 \ldots 1]

[0 1 0 1 \ldots 1]

Merge storage & computation i.e. In-memory computing

Highly parallel, fault-tolerant binary operators, assoc-min-distance search

Accuracy

\begin{align*}
\text{2048} & \quad \text{4096} & \quad \text{8192} \\
\text{ETH} & \quad \text{ETH} & \quad \text{ETH} \\
\text{Ours} & \quad \text{Data et. al.}
\end{align*}

90%
In-memory Hyperdimensional Computing

Associative Memory
(latch based SCM)

\[
\begin{array}{c}
[0100010] \\
[1000101] \\
[0100101] \\
\vdots \\
[0100101] \\
[0100101] \\
\end{array}
\]

\[ \cdots \]

Adder Tree

A > B

\( N_{\text{CLASS}} \) cycles
# HD-Based smart Wake-Up Module - Hypnos

[github.com/pulp-platform/hypnos](https://github.com/pulp-platform/hypnos)

## Design (post P&R)

<table>
<thead>
<tr>
<th></th>
<th>Technology</th>
<th>GF22 UHT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Area</strong></td>
<td></td>
<td>670kGE</td>
</tr>
<tr>
<td><strong>Max. Frequency</strong></td>
<td></td>
<td>3 MHz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>f_{clk}</th>
<th>32kHz</th>
<th>200kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>max. sampling rate</strong></td>
<td>150 SPS/Channel</td>
<td>1kSPS/Channel</td>
<td></td>
</tr>
<tr>
<td><strong>P_{SWU, dynamic}</strong></td>
<td>0.99uW</td>
<td>6.21uW</td>
<td></td>
</tr>
<tr>
<td><strong>P_{SWU, leakage}</strong></td>
<td>0.7uW</td>
<td>0.7uW</td>
<td></td>
</tr>
<tr>
<td><strong>P_{SPI, dynamic}</strong></td>
<td>1.28uW</td>
<td>8.00uW</td>
<td></td>
</tr>
<tr>
<td><strong>P_{SWU, total}</strong></td>
<td>Measured</td>
<td>2.97uW</td>
<td>14.9uW</td>
</tr>
</tbody>
</table>

Implemented with lowest leakage cell library (UHVT)

[Eggiman et al. TCAS22]
Processors with their narrow memory ports and LIC are arbitrated vs. HWPEs at the memory interface via a shallow interconnect.
HWPEs expose a unified high-bandwidth port (e.g., 256-bit or 512-bit) towards a simple Port Reordering block ("wide" access without self-contention)
High-Bandwidth access: Heterogeneous Cluster Interconnect

A Shallow Interconnect dispatches accesses to single 32-bit memory banks and manages conflicts by means of rotating configurable priority (e.g., max 3 cycles of stall)

How much can we scale it? … a lot with a bit of latency!
Scaling up: The MemPool Family

Hierarchical low-latency interconnect + Latency-Tolerant Core (snitch)

MinPool: first tape-out
- 16 cores, 64 KiB, 3 cycles
- TSMC 65

MemPool: main driver
- 256 cores, 1 MiB, 5 cycles
- GF 22FDX
  - 500 MHz (wc)
- MemPool-3D

TeraPool: going even bigger
- 1024 cores, 4 MiB, 7 cycles
- GF 12 (500MHz+)
- Break the TOPS barrier
- TeraPool-3D?
Why Scaleup? The Era of Foundation Models

- Versatility
  - Natural language processing, computer vision, robotics, biology, ...
- Homogenization of models
  - Transformers as *foundation models*!
- Transfer learning
  - Train on a large-scale dataset and fine-tune on specific tasks with smaller datasets.

Attention is all you need!

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention
Attention but how?

Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm

Linear
Softmax
Linear
Linear
Linear

Query
Key
Value

I love ISLPED!
Challenges in Attention

- Attention matrix is a square matrix of order input length.
  - Computational complexity
  - Memory requirements
Challenges in **Attention**

- Attention matrix is a square matrix of order input length.
  - Computational complexity
  - Memory requirements

- Every attention layer applies **Softmax** to attention matrix!
Challenges in **Attention**

- Attention matrix is a square matrix of order input length.
  - Computational complexity
  - Memory requirements

- Every attention layer applies **Softmax** to attention matrix!
  - 3 passes over a row.
  - Quantization is problematic.

$$\text{Softmax}(x)_i = \frac{e^{x_i - \text{max}(x)}}{\sum_{j}^{n} e^{x_j - \text{max}(x)}}$$
ITA: Integer Transformer Accelerator

- **Attention** accelerator for transformers!
- INT8 quantized networks
- Output stationary - Local weight stationary
  - Spatial input reuse
  - Spatial output partial sum reuse
- Fused Q.K^T and A.V computation
- Special **Softmax** unit!

[Islamoglu et al. ISLPED23]
ITA – Architecture

$N = 16$ dot product units that compute the dot product between two vectors of $M = 64$ elements.
Hardware-friendly **Softmax**

\[
\text{Softmax}(x)_i = \frac{e^{x_i - \text{max}(x)}}{\sum_{j}^{n} e^{x_j - \text{max}(x)}}
\]
Hardware-friendly **Softmax**

$$ \text{Softmax}(x)_i = \frac{e^{x_i - \max(x)}}{\sum_j^n e^{x_j - \max(x)}} $$

$$ \text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5} $$

- Directly operates on quantized values.
- No exponentiation modules and multipliers.
- Computes softmax on streaming data.
Hardware-friendly *Softmax*

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qj} - \max(x_q))\gg 5}} 2^{(x_{qi} - \max(x_q))\gg 5}
\]
Hardware-friendly **Softmax**

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]
Hardware-friendly **Softmax**

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]
Hardware-friendly **Softmax**

$$\text{Softmax}(x)_i = \frac{1}{\sum^n_j 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}$$
Hardware-friendly *Softmax*

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]

\[
\text{Element Normalization} \quad \text{Denominator Inversion} \quad \text{Denominator Accumulation}
\]
Output stationary - Local weight stationary

Input  ×  Weight  =  Output
Output stationary - Local weight stationary
Output stationary - Local weight stationary

Input × Weight = Output
Output stationary - Local weight stationary

Input \times Weight = Output
Output stationary - Local weight stationary
Output stationary - Local weight stationary
Output stationary - Local weight stationary

Input \xrightarrow{\times} \text{Weight} \equiv \text{Output}
Output stationary - Local weight stationary

- Input
- Weight
- Output

Dot Product Units

<table>
<thead>
<tr>
<th>Q</th>
<th>K</th>
<th>V</th>
<th>Output</th>
</tr>
</thead>
</table>

- Output
- Local weight stationary
Fused $Q.K^T$ and A.V computation

Dot Product Units

- Q
- K
- V
- $Q.K^T$
- A.V
- Output

Softmax

- DA
- EN
- DI

Input 1

Input 2

Output
Physical Implementation

- Implemented in GF22FDX
  - Target frequency of 500 MHz (SS/0.72V/125°C)
- Area 0.17 mm²
  - Softmax module has only 3.3% area contribution, corresponding to 28.7 KGE.
- Power 60 mW (TT/0.80V/25°C)
  - Softmax module consumes 1.4% of the power.
Comparison to a software baseline on MemPool

- Many-core system with low-latency L1 memory
- Designed for highly parallel workloads
- 256 32-bit RISC-V cores
- 1 MiB L1 scratchpad memory
Comparison to a software baseline on MemPool

- Performance [TOPS] increase of 6x
- Energy Efficiency increase of 45x
- Area Efficiency increase of 220x

MemPool vs. ITA System with 64 KiB SRAM
Integrating ITA into MemPool

• Not very straightforward!
• Hierarchical architecture
  • Cluster
  • Group
  • Tile
Integrating ITA into MemPool

• Where to put ITA?
• How to connect ITA to L1 memory?
• How to refill L1 from L2 memory for ITA?
One ITA core per Group

- ITA fits **four tiles** of MemPool
One ITA core per Group

- ITA fits **four tiles** of MemPool
- Bottom right
- Remove the cores
- Rearrange the banks
One ITA core per Group

- ITA fits **four tiles** of MemPool
- Bottom right
- Remove the cores
- Rearrange the banks
- Each ITA core works on one head of attention
Integrating ITA into MemPool

✓ Where to put ITA?
  • How to connect ITA to L1 memory?
  • How to refill L1 from L2 memory for ITA?
Modified Interconnect of Four Tiles

- 4 tiles = 64 banks
- ITA needs to access 28 banks per cycle
- 3 types of requests/responses
  - Core
  - ITA
  - DMA

ITA > DMA > Core
Integrating ITA into MemPool

- Where to put ITA?
- How to connect ITA to L1 memory?
  - How to refill L1 from L2 memory for ITA?
Adding a special DMA for ITA

- Moves transformer data from L2 to L1 memory
- Inputs are broadcasted to all groups
- Two 16 bytes/cycle ports per group
Integrating ITA into MemPool

- Where to put ITA?
- How to connect ITA to L1 memory?
- How to refill L1 from L2 memory for ITA?

End-to-end heterogeneous collaborative Transformer deployment
Comparison to MemPool and ITA System

<table>
<thead>
<tr>
<th></th>
<th>Throughput [TOPS]</th>
<th>Energy efficiency [TOPS/W]</th>
<th>Area efficiency [TOPS/mm²]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemPool</td>
<td>0.135</td>
<td>0.159</td>
<td>0.0114</td>
</tr>
<tr>
<td>ITA</td>
<td>3.43</td>
<td>7.09</td>
<td>2.10</td>
</tr>
<tr>
<td>ITA &amp; Banks</td>
<td>1.02</td>
<td>12.3</td>
<td>5.02</td>
</tr>
</tbody>
</table>

ITA DMA

ITA

ITA

ITA

ITA

ITA

ITA

ITA

ITA

ITA

ITA

ITA

ITA
### Comparison to MemPool and ITA System

<table>
<thead>
<tr>
<th></th>
<th>MemPool</th>
<th>ITA &amp; Banks</th>
<th>ITA only</th>
<th>ITA System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput [TOPS]</td>
<td>0.135</td>
<td>3.43</td>
<td>3.43</td>
<td>1.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>25x</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Energy efficiency</td>
<td>0.159</td>
<td>7.09</td>
<td>12.3</td>
<td>8.46</td>
</tr>
<tr>
<td>[TOPS/W]</td>
<td></td>
<td><strong>45x</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Area efficiency</td>
<td>0.0114</td>
<td>2.10</td>
<td>5.02</td>
<td>2.52</td>
</tr>
<tr>
<td>[TOPS/mm²]</td>
<td></td>
<td></td>
<td><strong>2x</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table compares the performance metrics of MemPool and ITA System. The values indicate improvements or reductions in various aspects such as throughput, energy efficiency, and area efficiency. The columns show the performance metrics for MemPool, ITA & Banks, ITA only, and the ITA System itself. The highlighted values (e.g., **25x**, **45x**, **2x**) indicate significant differences or improvements. The ITA (ITA System) is highlighted to emphasize its unique features and performance compared to MemPool and ITA & Banks.
Future of ITA: Scaling up further

- Target workloads like GPT
- Floating-point capability

Accelerate LLMs and reach **100 TFLOPS or higher!**
Off-chip (Memory & Sensors): Feel the (IO) Pain!

No good solutions to curtail IO power for extreme edge ML

- SPIs
  - I/O VDD=1.8V
  - fspi-max=50MHz,
  - Assuming duty-cycled operation @ various bandwidths
- ULP serial link (duty-cycled)
  - 10.2x less energy and 15.7x higher maximum BW compared to single SPI
  - 2.56x higher efficiency than the DDR Octal SPI @787Mbps
  - 5 \( \rightarrow \) 3pJ/bit
  - However it’s still 2mW@ 500Mbps
- 3D integration: 0.15pJ/bit and below

2.5D and 3D coming fast even for extreme edge!

[Okuhara et al. ISCAS20]
Closing thoughts

- Edge Computing requires **flexibility, efficiency and energy proportionality**
  - Multiple efficiency boosters (accelerators): ISA extensions, HWCEs: 1-4 OoM!
  - On-chip non-volatile memory, event-based processing for proportionality

- Moving forward – **proliferation of acceleration engines**
  - For tuning accelerator to sensor (like the brain)
  - For high-level sensor fusion, control, planning

- Managing inter-accelerator dataflows – **low latency & high-bandwidth**
  - Low-latency interconnects are key
  - Processors, Memory and interconnect fabrics need to be co-designed
  - Scaling to full-die and (heterogeneous) chiplets – next generation NoCs

- Scale-up for foundation models
Perceptive → Generative → Embodied AI

Precise

Interactive, creative

Efficient, RT-safe, secure
Thanks!

[9] pulp-platform/CUTIE (github.com)

[8] pulp-platform/sne (github.com)

https://www.research-collection.ethz.ch/handle/20.500.11850/565105