PULP: A Multi-Core Platform for Micropower In-Sensor Analytics

OPRECOMP SUMMER SCHOOL 03.09.2019

Davide Rossi
davide.rossi@unibo.it

1Department of Electrical, Electronic and Information Engineering
2Integrated Systems Laboratory
A Very Short Review on CMOS power (and power minimization)
Summary of Power Dissipation Sources

\[ P \sim \alpha \cdot (C_L) \cdot V_{\text{swing}} \cdot V_{DD} \cdot f + (I_{DC} + I_{\text{leak}}) \cdot V_{DD} \]

- \( \alpha \) – switching activity
- \( C_L \) – load capacitance
- \( V_{\text{swing}} \) – voltage swing
- \( f \) – frequency

\[ P = \frac{\text{energy}}{\text{operation}} \times \text{rate} + \text{static power} \]

- \( I_{DC} \) – static current
- \( I_{\text{leak}} \) – leakage current
The Traditional Design Philosophy

- Maximum performance is primary goal
  - Minimum delay at circuit level
- Architecture implements the required function with target throughput, latency
- Performance achieved through optimum sizing, logic mapping, architectural transformations.
- Supplies, thresholds set to achieve maximum performance, subject to reliability constraints
The New Design Philosophy

- Maximum performance (in terms of propagation delay) is too power-hungry, and/or not even practically achievable
- Many (if not most) applications either can tolerate larger latency, or can live with lower than maximum clock-speeds
- Excess performance (as offered by technology) to be used for energy/power reduction

Trading off speed for power
In energy-constrained world, design is trade-off process

♦ Minimize energy for a given performance requirement
♦ Maximize performance for given energy budget
Reducing power @ all design levels

- Algorithmic level
- Compiler level
- Architecture level
- Micro-Architecture
- Circuit level
- Silicon level

- Important concepts:
  - Lower Vdd and freq. (even if errors occur) / dynamically adapt Vdd and freq.
  - Reduce circuit
  - Exploit locality
  - Reduce switching activity, glitches, etc.
Algorithmic level

- The best indicator for energy is ..... 
  .... the number of cycles

- Try alternative algorithms with lower complexity
  - E.g. quick-sort, O(n log n) ⇔ bubble-sort, O(n²)
  - ... but be aware of the 'constant': O(n log n) ⇒ c*(n log n)

- Heuristic approach
  - Go for a good solution, not the best !!

Biggest gains at this level !!
Compiler level

- Source-to-Source transformations
  - loop trafo's to improve locality
- Strength reduction
  - E.g. replace Const * A with Add's and Shift's
  - Replace Floating point with Fixed point
- Reduce register pressure / number of accesses to register file
  - Use software bypassing
- Scenarios: current workloads are highly dynamic
  - Determine and predict execution modes
  - Group execution modes into scenarios
  - Perform special optimizations per scenario
    - DFVS: Dynamic Voltage and Frequency Scaling
    - More advanced loop optimizations
- Reorder instructions to reduce bit-transistions
Going parallel
Going heterogeneous
- tune your architecture, exploit SFUs (special function units)
- trade-off between flexibility / programmability / genericity and efficiency
Add local memories
- prefer scratchpad i.s.o. cache
Cluster FUs and register files (see next slide)
Reduce bit-width
- sub-word parallelism (SIMD)
Organization (micro-arch.) level

- Enabling Vdd reduction
  - Pipelining
    - cheap way of parallelism
  - Enabling lower freq. \( \Rightarrow \) lower \( V_{dd} \)
  - Note 1: don't pipeline if you don't need the performance
  - Note 2: don't exaggerate (like the 31-stage Pentium 4)

- Reduce register traffic
  - avoid unnecessary reads and write
  - make bypass registers visible
Circuit level

- Clock gating
- Power gating
- Multiple Vdd modes
- Reduce glitches: balancing digital path's
- Exploit Zeros
- Special SRAM cells
  - normal SRAM can not scale below Vdd = 0.7 - 0.8 Volt
- Razor method; replay
- Allow errors and add redundancy to architectural invisible structures
  - branch predictor
  - caches
- .. and many more ..
Silicon level

- Higher $V_t$ ($V_{\text{threshold}}$)
- Back Biasing control
  - see thesis Maurice Meijer (2011)
- SOI (Silicon on Insulator)
  - silicon junction is above an electr. insulator (silicon dioxide)
  - lowers parasitic device capacitance
- Better transistors: Finfet
  - multi-gate
  - reduce leakage (off-state current)
- .. and many more
Near-Sensor Processing
Computing for the Internet of Things

Sense
- MEMS IMU
- MEMS Microphone
- ULP Imager
- EMG/ECG/EIT
- Battery + Harvesting powered → a few mW power envelope

Analyze and Classify
- ULP parallel processor
- µController
- L2 Memory
- IOs
- 1 ÷ 25 MOPS
- 1 ÷ 10 mW

Transmit
- Short range, medium BW
- Low rate (periodic) data
- SW update, commands
- Long range, low BW
- Idle: ~1µW
- Active: ~50mW

Davide Rossi | 06.09.2019 | 15
Near-Sensor Processing

- **Image**
  - Tracking: 80 Kbps 1.34 GOPS 0.16 Kbps 500x
  - [*Lagroce2014]*

- **Voice/Sound**
  - Speech: 256 Kbps 100 MOPS 0.02 Kbps 12800x
  - [*VoiceControl]*

- **Inertial**
  - Kalman: 2.4 Kbps 7.7 MOPS 0.02 Kbps 120x
  - [*Nilsson2014]*

- **Biometrics**
  - SVM: 16 Kbps 150 MOPS 0.08 Kbps 200x
  - [*Benatti2014]*

**Exremely compact output (single index, alarm, signature)**

**Computational power of ULP µControllers is not enough**

**Parallel workloads**
PULP: pJ/op Parallel ULP computing

- Programming Model
  - OpenVX
  - OpenMP
- Virtualization Layer
- Compiler Infrastructure
  - MicroPython
  - RTOS
- Processor & Hardware IPs
- Low-Power Silicon Technology

Parallel + Programmable + Heterogeneous ULP computing
1mW-10mW active power
Parallel Ultra Low Power
Near-Threshold Computing (NTC):

- Don’t waste energy pushing devices in strong inversion
- Recover performance with parallel execution
- Aggressively manage idle power (switching, leakage)
- Manage Process and temperature variations in NT

![Diagram showing energy per cycle vs. voltage with 32nm CMOS, 25°C and 4.7X improvement at 0.55 V.](VivekDeDATE2013)
### Parallel NTC

**Core Power [mW]** vs **Workload [MOPS]**

- **Single-core** vs **Multi-core**

#### Low Workloads
- SUB-Vth

#### High Workloads
- NEAR-Vth

#### Target Workload
- [DoganICSDPTMO20 11]

<table>
<thead>
<tr>
<th>Target Workload [MOPS]</th>
<th>1-Core Energy Efficiency (ideal) [MOPS/mW]</th>
<th>4-Cores Energy Efficiency (ideal) [MOPS/mW]</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>43</td>
<td>55</td>
<td>1.3x</td>
</tr>
<tr>
<td>200</td>
<td>33</td>
<td>50</td>
<td>1.5x</td>
</tr>
<tr>
<td>400</td>
<td>18</td>
<td>43</td>
<td>2.4x</td>
</tr>
</tbody>
</table>

*Measured on our first prototype*
The main constraint here is the power envelope active period.

SINGLE-CORE @ MAX FREQUENCY (e.g. 200MHz)

- Low Workload (duty cycled)

MULTI-CORE @ MAX FREQUENCY (e.g. 200 MHz)

- Ideally same energy of single-core solution
- Deep sleep
- Saved energy
- Low Workload (duty cycled)

- Going faster allows to integrate system power over a smaller period
- The main constraint here is the power envelope
Building PULP
Building PULP

**SIMD + MIMD + sequential**

- Private per-core instruction cache
- 4-stage, in-order OpenRISC core
- DEMUX
- 1 Cycle Shared Multi-Banked L1 Data Memory + Low Latency Interconnect

**“GPU like” shared memory → low overhead data sharing**
- Near Threshold but parallel → Maximum Energy efficiency when Active
- + strong power management for (partial) idleness

Near threshold FDSOI technology

Body bias: Highly effective knob for power & variability management!
Near Threshold + Body Biasing Combined

**FBB vs. FREQUENCY**

- VDD=0.5V
- VDD=0.6V
- VDD=0.7V
- VDD=0.8V
- VDD=0.9V
- VDD=1.0V
- VDD=1.1V
- VDD=1.2V

+ 2.5x @0.5V

**RBB vs. LEAKAGE**

- VDD=0.5V
- VDD=0.6V
- VDD=0.7V
- VDD=0.8V
- VDD=0.9V
- VDD=1.0V
- VDD=1.1V
- VDD=1.2V

- 10x @0.5V

**RVT transistors**

- State retentive (no state retentive registers and memories)
- Ultra-fast transitions (tens of ns depending on n-well area to bias)
- Low area overhead for isolation (3µm spacing for deep n-well isolation)
- Thin grids for voltage distribution (small transient current for wells polarization)
- Simple circuits for on-chip VBB generation (e.g. charge pump)

**But even with aggressive RBB leakage is not zero!**
Selective, Fine Grained Body Biasing

- The cluster is partitioned in separate clock gating and body bias regions
- Body bias multiplexers (BBMUXes) control the well voltages of each region
- A Power Management Unit (PMU) automatically manages transitions between the operating modes
- Power modes of each region:
  - **Boost mode**: active + FBB
  - **Normal mode**: active + NO BB
  - **Idle mode**: clock gated + NO BB (in LVT) RBB (in RVT)
PULPv1 issues:

1) World record energy efficiency (60 MOPS/mW), but @ too small performance (20 MOPS)
2) SRAM limits voltage scalability (very well known problem…)

---

**PULPv1**

**CHIP FEATURES**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>28nm FDSOI (RVT)</td>
</tr>
<tr>
<td>Chip Area</td>
<td>3mm²</td>
</tr>
<tr>
<td># Cores</td>
<td>4xOpenRISC</td>
</tr>
<tr>
<td>I$</td>
<td>4x1kbyte (private)</td>
</tr>
<tr>
<td>TCDM</td>
<td>16 kbyte</td>
</tr>
<tr>
<td>L2</td>
<td>16 kbyte</td>
</tr>
<tr>
<td>BB regions</td>
<td>6</td>
</tr>
<tr>
<td>VDD range</td>
<td>0.45V - 1.2V (SRAM: 0.55V - 1.2V)</td>
</tr>
<tr>
<td>VBB range</td>
<td>-1.8V - +0.9V</td>
</tr>
<tr>
<td>Perf. Range</td>
<td>1 MOPS - 1.9 GOPS</td>
</tr>
<tr>
<td>Power Range</td>
<td>100 µW - 127 mW</td>
</tr>
<tr>
<td>Peak Efficiency</td>
<td>60 GOPS/W@20 MOPS, 0.55V</td>
</tr>
</tbody>
</table>

---

D. Rossi et al., «A 60 GOPS/W, −1.8V to 0.9V body bias ULP cluster in 28nm UTBB FD-SOI technology», in Solid-State Electronics, 2016.
The Memory bottleneck

**PULPv1 POWER BREAKDOWN @ BEST ENERGY POINT:**

**PULPv1 issues:**

1) World record energy efficiency (60 MOPS/mW), but @ too small performance (20 MOPS)

2) SRAM limits voltage scalability (very well known problem…)

3) SRAM forms a huge bottleneck for energy efficiency (>60% of total power)
ULP (NT) Bottleneck: Memory

- **“Standard” 6T SRAMs:**
  - High VDDMIN
  - Bottleneck for energy efficiency
    - >50% of energy can go here!!!
- **Near-Threshold SRAMs (8T):**
  - Lower VDDMIN
  - Area/timing overhead (25%-50%)
  - High active energy
  - Low technology portability
- **Standard Cell Memories:**
  - Wide supply voltage range
  - Lower read/write energy (2x - 4x)
  - High technology portability
  - Major area overhead 4x → 2.7x with controlled placement

---


---

**256x32 6T SRAMs vs. SCM**

- **Voltage [V]**
  - 0.3 to 1
- **Read Energy [32 bit] [pJ]**
  - 0 to 5

- **SCM**
- **High VDDMIN**
- **Near-Threshold SRAMs (8T)**
- **Low technology portability**
- **Low read/write energy (2x - 4x)**
- **Wide supply voltage range**
- **Major area overhead 4x → 2.7x with controlled placement**

---

**Figure captions:**

- **256x32 6T SRAMs vs. SCM**
  - Voltage [V] from 0.3 to 1
  - Read Energy [32 bit] [pJ] from 0 to 5
- **SCM**, **High VDDMIN**, **Near-Threshold SRAMs (8T)**, **Low technology portability**, **Low read/write energy (2x - 4x)**, **Wide supply voltage range**, **Major area overhead 4x → 2.7x with controlled placement**

---

PULPv2

CHIP FEATURES

- **Technology**: 28nm FDSOI (LVT)
- **Chip Area**: 3mm²
- **# Cores**: 4xOpenRISC
- **I$ (SCM)**: 4x1kbyte (private)
- **TCDM**: 32 + 8 Kbyte
- **L2**: 64 kbyte
- **BB regions**: 10
- **VDD range**: 0.3-1.2V (0.5-1.2V)
- **VBB range**: 0V-2V
- **Perf. Range**: 1 MOPS - 4 GOPS
- **Power Range**: 10µW - 300 mW
- **Peak Efficiency**: 192 GOPS/W@0.5V

I$: a Look Into ‘Real Life’ Applications

SCM-BASED I$ IMPROVES EFFICIENCY BY ~2X ON SMALL BENCHMARKS, BUT...

Survey of State of The Art

<table>
<thead>
<tr>
<th>Existing ULP processors</th>
<th>Latch based I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>REISC (ESSCIRC2011)</td>
<td>64b</td>
</tr>
<tr>
<td>Sleepwalker (ISSCC 2012)</td>
<td>128b</td>
</tr>
<tr>
<td>Bellevue (ISCAS 2014)</td>
<td>128b</td>
</tr>
</tbody>
</table>

Issues:

1) Area Overhead of SCMs (4Kb/core not affordable....)
2) Capacity miss (with small caches)
3) Jumps due to runtime (e.g. OpenMP, OpenCL) and other function calls
OpenMP for PULP

- **OpenMP on PULP:**
  - Lightweight implementation on top of a bare-metal runtime (custom GCC libgomp)
  - A subset of OpenMP 3.0 supported (e.g. no tasking)
  - Power management embedded in the runtime (transparent to the end-user)

- **Architectural Implications**
  - #pragmas are translated into runtime function calls
  - Function calls cause L1$ cache pollution
  - Call to OpenMP functions feature intrinsic overhead

```c
#pragma omp parallel
{
  #pragma omp sections
  {
    #pragma omp section
    {
      Task_A();
    }
  #pragma omp section
  {
    Task_B();
  }
  #pragma omp section
  {
    Task_C();
  }
  #pragma omp section
  {
    Task_D();
  }
}
```

The Solution: Shared I$

- Share instruction cache
  - OK for data parallel execution model
  - Not OK for task parallel execution model, or very divergent parallel threads
- Architectures
  - SP: single-port banks connected through a read-only interconnect
    - Pros: Low area overhead
    - Cons: Timing pressure, contention
  - MP: Multi-ported banks
    - Pros: High efficiency
    - Cons: Area overhead (several ports)
- Hierarchical Cache:
  - Pros: P&R friendly
  - Cons: Slightly slower than the others.

The Solution: Shared $I$\$

**Multi-Port Shared $I$\$**

- MP: Multi-ported banks
  - Pros: High efficiency
  - Cons: Area overhead (several ports)

**Hierarchical Shared $I$\$**

- Hierarchical Cache:
  - Pros: P&R friendly
  - Cons: Slightly slower than the others.

---

Synthetic Benchmarks

Real-Life Applications

Throughput

Energy Efficiency

HW Synchronizer: Impact on OpenMP primitives

- Cost of OpenMP runtime reduced by more than one order of magnitude
- Better scalability with number of cores

Extending RISC-V for NSP

<32-bit precision → SIMD2/4 opportunity

1. HW loops and Post modified LD/ST
2. Bit manipulations
3. Packed-SIMD ALU operations with dot product
4. Rounding and Normalization
5. Shuffle operations for vectors

V1  Baseline RISC-V RV32IMC
    HW loops

V2  Post modified Load/Store
    Mac

V3  SIMD 2/4 + DotProduct + Shuffling
    Bit manipulation unit
    Lightweight fixed point

Small Power and Area overhead

ISA Extensions improve performance

for (i = 0; i < 100; i++)
    d[i] = a[i] + b[i];

Baseline

mv x5, 0
mv x4, 100
Lstart:
    lb x2, 0(x12)
    lb x3, 0(x12)
    addi x10, x10, 1
    addi x11, x11, 1
    add x2, x3
    sb x2, 0(x12)
    addi x4, x4
    add x2, x3
    addi x12, x12
    bne x4, x5

Auto-incr load/store

mv x5, 0
mv x4, 100
Lstart:
    lb x2, 0
    lb x3, 0
    addi x10, x10, 1
    addi x11, x11, 1
    add x2, x3
    sb x2, 0
    addi x4, x4
    add x2, x3
    addi x12, x12
    bne x4, x5, Lstart

HW Loop

11 cycles/output 8 cycles/output 5 cycles/output 1.25 cycles/output

Packed-SIMD

lp.setups 100, Lend
lp.setups 25, Lend
lw x2, 0(x10!)
lw x3, 0(x11!)

Pulp

Davide Rossi | 06.09.2019 | 39
What About FP?

- The adoption of floating point formats requiring a lower number of bits (smallFloats) can reduce execution time and energy consumption
  - Simpler Logic (smaller pj/op)
  - Vectorization (smaller execution time)
- SW support:
  - flexFloat library for emulation of smallFloat format
  - Automatic exploration tool to tune precision of individual FP operations for the required application accuracy
- Hardware support:
  - smallFloat scalar and vector operations
  - Casting operations to move data through different FP types
Process & Temperature Variations in NTC

Process variation over a distribution of 60 chips @ 0.6V

25 MHz ± 7 MHz (3σ)

Normalized Frequency

120°C

100x @0.5V

-40°C

0.5V

0.8V

Voltage

Normalized Frequency
Body Bias-Based Performance Monitoring and Control

Goal:

- Compensate the effects of external factors like ambient temperature
- Reduce PVT Margins at design time
- Improve Energy Efficiency

→ Mixed Hardware/Software control loop exploiting on-chip frequency measurement (PMB)

---

Dynamic PVT Compensation

- PULPv3 board featuring voltage regulator (down to 0.5V)
- Peltier element to heat-up / cool the samples
- TEC controller to drive the peltier element
- Embecosm MAGEEC Energy Monitoring Shield for power measurements
- JTAG Programmer for loading code on PULPv3
- Host laptop to show plots

Extended operating range for zero-margin design (i.e. signoff in typical corner)

- 30% energy reduction
- >2x leakage reduction @ 70°C

A Full SoC Perspective
The SoC is an advanced microcontroller based on an ultra-low-power 32-bit RISC-V processor

- 512kB of L2 Memory

Rich set of peripherals:

- QSPI, SPI, I2C, I2S
- HyperRam + HyperFlash
- Camera Interface
- JTAG (Debug), GPIOs, PWM, ROM
- RTC, Interrupt controller

Autonomous IO DMA Subsystem

Advanced power management

- 2 Switchable Power Domains
- 2 low-power FLLs (IO, SoC, Cluster)
- On-chip DCDC and LDO
- State-Retentive L2 Memory

Parallel Programmable Accelerator:

- 8x DSP enhanced RISC-V processors
- 64kB of Shared L1 Memory
- 2x Shared Floating Point Units
- Shared latch-based Instruction Cache
- High performance DMA(L2<->L1)
- Event Unit supporting fast synchronization among cores
Mr. Wolf Chip Results

<table>
<thead>
<tr>
<th>Technology</th>
<th>CMOS 40nm LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip area</td>
<td>10 mm²</td>
</tr>
<tr>
<td>VDD range</td>
<td>0.8V - 1.1V</td>
</tr>
<tr>
<td>Memory Transistors</td>
<td>576 Kbytes</td>
</tr>
<tr>
<td>Logic Transistors</td>
<td>1.8 Mgates</td>
</tr>
<tr>
<td>Frequency Range</td>
<td>32 kHz – 450 MHz</td>
</tr>
<tr>
<td>Power Range</td>
<td>72 µW – 153 mW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power Management (DC/DC + LDO)</th>
<th>VDD [V]</th>
<th>Freq.</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Sleep</td>
<td>0.8</td>
<td>n.a.</td>
<td>72 µW</td>
</tr>
<tr>
<td>Ret. Deep Sleep</td>
<td>0.8</td>
<td>n.a.</td>
<td>76.5 - 108 µW</td>
</tr>
<tr>
<td>SoC Active</td>
<td>0.8 - 1.1</td>
<td>32 kHz - 450 MHz</td>
<td>0.97 - 38 mW</td>
</tr>
<tr>
<td>Cluster Active</td>
<td>0.8 - 1.1</td>
<td>32 kHz - 350 MHz</td>
<td>1.6 - 153 mW</td>
</tr>
</tbody>
</table>

Efficient Synchronization and Power Management

GOALS:
- Reduce cost of parallelization of data-parallel programming models (OpenMP, OpenCL)
- Shut-down (clock gating) processors during idle periods

RESULTS
- ~15x latency and energy reduction for a barrier
- 30%-50% latency and energy reduction for a mutex

ARCHITECTURE

Mr. Wolf Parallel Speed-up

- Amdahl’s Limit
- Speed-Up

- Diagram showing speed-up as a function of the number of cores for various tasks.
  - PCA
  - HD
  - CNN Layer (16-bit)
  - BNN
  - CNN Layer (8-bit)
  - FIR
  - SVM
  - MatMul (8-bit)
  - 5x5 Conv (16-bit)
  - 5x5 Conv (32-bit)
  - 5x5 Conv (8-bit)
  - MatMul (32-bit)
  - MatMul (32-bit)
  - FFT
  - DWT
On 2D Matrix Multiplication
Mr. Wolf Computing Efficiency (Fixed+Float)

On 2D Matrix Multiplication

Efficiency [MMAC/S/mW]

VDD [V]

SoC Integer Efficiency (32-bit)
Cluster Int Efficiency (32-bit)
Cluster FP Efficiency (32-bit)

12x
Recovering Efficiency Through Flexible Customization
Closing The Accelerator Efficiency Gap with Agile Customization

Recovering More Silicon Efficiency

GOPS/W

1

3

6

> 100

General-purpose Computing
CPU

Throughput Computing
GPGPU

ULP parallel Computing

Accelerator Gap

HW IP

SW
Mixed
HW

1GOPS/mW

Mixing SW and HW Modes
Recovering Even More Efficiency

Fixed function accelerators have limited reuse… how to limit proliferation?

Shared Tightly-Coupled Data Memory (TCDM)

Logarithmic Interconnect

Peripheral Interconnect

DMA Channel #0

DMA Channel #1

EXT2MEM

EXT2PER

ENC2EXT

Timers

HWS

Memory Bank #0

Memory Bank #1

Memory Bank #2

Memory Bank #3

Memory Bank #30

Memory Bank #31

16-KB P$ PE #0

16-KB P$ PE #15

PULP
Learn to Accelerate

- Brain-inspired (deep convolutional networks) systems are high performers in many tasks over many domains

→ Human:
  85% (untrained), 94.9% (trained)

→ CNN:
  93.4% accuracy

Image recognition
[Russakovsky IMAGENET2014]

Speech recognition
[Hannun ARXIV 2014]

Flexible acceleration: learned CNN weights are “the program”
PULP CNN Performance

Average performance and energy efficiency on a 32x16 CNN frame

**PERFORMANCE**

- **8 GOPS**
- **61x**

**ENERGY EFFICIENCY**

- **6500 GOPS/W**
- **47x**

**PULPv3 ARCHITECTURE, CORNER: tt28, 25°C, VDD= 0.5V, FBB = 0.5V**
Sub pJ/OP? Approximate Computing

0% bit flips

437 GOPS/W @1.2V
803 GOPS/W @0.8V

CNNs are intrinsically resilient to (a small amount) of soft errors

1% bit flips

1.84x energy improvement

1.2pJ/OP
From Approximate to Transprecision: YodaNN\(^1\)

- Approximation at the algorithmic side → Binary weights
- BinaryConnect [Courbariaux, NIPS15]
  - Reduce weights from 12-bit to a binary value -1/+1
  - Stochastic Gradient Descent with Binarization in the Forward Path

\[
\begin{align*}
w_{b,\text{stoch}} &= \begin{cases} 
-1 & p_{-1} = \sigma(w) \\
1 & p_{1} = 1 - p_{-1}
\end{cases} \\
w_{b,\text{det}} &= \begin{cases} 
-1 & w < 0 \\
1 & w > 0
\end{cases}
\end{align*}
\]

- Learning large networks is still an issue with binary connect…

- Ultra-optimized HW is possible!
  - Power reduction because of arithmetic simplification (multipliers → Two’s complement + muxes)
  - Major arithmetic density improvements
    - Area can be used for more energy-efficient weight storage
  - SCM memories for lower voltage → E goes with 1/V\(^2\)

\(^1\)After the Yedi Master from Star Wars - “Small in size but wise and powerful” cit. www.starwars.com
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Neuflow</td>
<td>320</td>
<td>490</td>
<td>1.0</td>
<td>17</td>
</tr>
<tr>
<td>Leuven</td>
<td>102</td>
<td>2600</td>
<td>0.5 - 1.1</td>
<td>64</td>
</tr>
<tr>
<td>Eyeriss</td>
<td>84</td>
<td>160</td>
<td>0.8 - 1.2</td>
<td>46</td>
</tr>
<tr>
<td>NINEX</td>
<td>569</td>
<td>1800</td>
<td>1.2</td>
<td>51</td>
</tr>
<tr>
<td>k-Brain</td>
<td>411</td>
<td>1930</td>
<td>1.2</td>
<td>109</td>
</tr>
<tr>
<td>Origami</td>
<td>196/74</td>
<td>437/803</td>
<td>1.2/0.8</td>
<td>90/34</td>
</tr>
<tr>
<td>This Work*</td>
<td>1510/55</td>
<td>9800/61200</td>
<td>1.2/0.6</td>
<td>1135/41</td>
</tr>
</tbody>
</table>

*UMC 65nm Technology, post place & route, 25°C, tt, 1.2V/0.6V

Breakthrough for ultra-low-power CNN ASIC implementation

fJ/op in sight: manufacturing in Globalfoundries 22FDX

Davide Rossi | 06.09.2019 | 59
From Frame-based to Event-based
Back to System-Level

Smart Visual Sensor $\rightarrow$ idle most of the time (nothing interesting to see)

- **Event-Driven Computation**, which occurs only when relevant events are detected by the sensor
- **Event-based sensor interface** to minimize IO energy (vs. Frame-based interface)
- **Mixed-signal event triggering** with an ULP imager with internal processing AMS capability

A Neuromorphic Approach for doing *nothing* VERY well
**GrainCam Imager (FBK)**

**Pixel-level spatial-contrast extraction**

\[ V_C = V_{PE}(t_1) - V_{PO}(t_1) = (V_R - V_{TH}) \left( \frac{I_{PO} - I_{PE}}{I_{PE}} \right) \]

**Anallog internal image processing**

- Contrast Extraction
- Motion Extraction, differencing two successive frames
- Background Subtraction, differencing the reference image, stored in the memory, with the current frame
GrainCam Readout

Readout modes:
- IDLE: readout the counter of asserted pixels
- ACTIVE: sending out the addresses of asserted pixels (address-coded representation), according raster scan order

Event-based sensing: output frame data bandwidth depends on the external context-activity

Ultra Low Power Consumption e.g. 10-20uW @10fps
Power Management

Graincam

PULP

#events > threshold
Switch to ACTIVE

Wake-up event

Data event

Graincam

IDLE

10μW @10fps

ACTIVE

10-20μW @10fps

READOUT

Data

PULP

Deep Sleep

Deep Sleep

7μW

2.88 mW Active Power @ 0.55V, 81MHz

### Results

- **3 Classes:**
  - Pedestrians
  - Bikes
  - Cars
- **84.6% vs. 81.6% Accuracy**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>BNN with RGB input</th>
<th>Event-based BNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Sensor Power Consumption</td>
<td>1.1mW @ 30fps 632446 bits 66.7 µJ</td>
<td>100µW @ 50fps 8192 bits 2 µJ</td>
</tr>
<tr>
<td>Image Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image Sensor Energy for frame capture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer Time (4bit SPI @ 50MHz)</td>
<td>3.1 msec</td>
<td>0.04 msec</td>
</tr>
<tr>
<td>Transfer Energy (8.9mW @ 0.7V)</td>
<td>28 µJ</td>
<td>2 µJ</td>
</tr>
<tr>
<td>BNN Execution Time (168MHz)</td>
<td>81.3 msec</td>
<td>75.3 msec</td>
</tr>
<tr>
<td>BNN Energy consumption (8.9mW @ 0.7V)</td>
<td>725 µJ</td>
<td>671 µJ</td>
</tr>
<tr>
<td>Total System Energy for Classification</td>
<td>820 µJ</td>
<td>674 µJ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistics per frame</th>
<th>Frame-Based</th>
<th>Event-based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Idle (no motion)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Power</td>
<td>1.1mW</td>
<td>20µW</td>
</tr>
<tr>
<td>Avg Sensor Data</td>
<td>19764 Bytes</td>
<td></td>
</tr>
<tr>
<td>Transfer Time</td>
<td>790µsec</td>
<td></td>
</tr>
<tr>
<td>Processing Time</td>
<td>3.02 msec</td>
<td></td>
</tr>
<tr>
<td>Avg Processor Power</td>
<td>1.45mW</td>
<td>0.3mW (sleep)</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Power</td>
<td>1.1mW</td>
<td>60µW</td>
</tr>
<tr>
<td>Avg Sensor Data</td>
<td>19764 Bytes</td>
<td>~536 Bytes</td>
</tr>
<tr>
<td>Transfer Time</td>
<td>790µsec</td>
<td>21.4µsec</td>
</tr>
<tr>
<td>Processing Time</td>
<td>3.47 msec</td>
<td>187.6µsec</td>
</tr>
<tr>
<td>Avg Processor Power</td>
<td>1.57mW</td>
<td>0.511mW</td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Power</td>
<td>2mW</td>
<td>60µW</td>
</tr>
<tr>
<td>Avg Sensor Data</td>
<td>79056 Bytes</td>
<td>1024 Bytes</td>
</tr>
<tr>
<td>Transfer Time</td>
<td>3.16 msec</td>
<td>41µsec</td>
</tr>
<tr>
<td>Processing Time</td>
<td>81.3 msec</td>
<td>75.3 msec</td>
</tr>
<tr>
<td>Processor Energy</td>
<td>760 µJ</td>
<td>677 µJ</td>
</tr>
</tbody>
</table>

---

Davide Rossi | 06.09.2019 | 66
Outlook and Conclusion
Conclusion

- Near-sensor processing → Energy efficiency requirements: pJ/OP and below
  - Technology scaling alone is not doing the job for us
  - Ultra-low power architecture and circuits are needed
- CNNs-based visual functions can be squeezed into mW envelope
  - Non-von-Neumann acceleration
  - Very robust to trans-precision computations (deterministic and statistical)
  - fJ/OP is in sight!
- More than CNN is needed (e.g. linear algebra, online optimization)
- Open Source HW & SW approach → innovation ecosystem
Thanks for your attention!!!