Working with RISC-V

Part 3 of 4: PULP concepts

Luca Benini <lbenini@iis.ee.ethz.ch>
Frank K. Gürkaynak <kgf@ee.ethz.ch>
Summary

- Part 1 – Introduction to RISC-V ISA
- Part 2 – Advanced RISC-V Architectures
- Part 3 – PULP concepts
  - Cores
  - Clusters
  - Heterogeneous systems
  - Accelerators
- Part 4 – PULP based chips
AI Workloads from Cloud to Edge (Extreme?)

High OP/B ratio
Massive Parallelism
MAC-dominated
Low precision OK
Model redundancy

GOP+
MB+
Energy efficiency @ GOPS is THE Challenge

High performance MCUs

Low-Power MCUs

Cool.. But how??

1pJ/OP=1TOPS/W

InceptionV4 @1fps in 10mW
RI5CY – Recap from Part 1

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

**V1** Baseline RISC-V RV32IMC (not good for ML)

**V2** HW loops, Post modified Load/Store, Mac

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

**XPULP 25 kGE → 40 kGE (1.6x) but 9+ times DSP!**

**Nice – But what about the GOPS?**

Faster+Superscalar is not efficient!

M7: 5.01 CoreMark/MHz - 58.5 µW/MHz

M4: 3.42 CoreMark/MHz - 12.26 µW/MHz
ML & Parallel, Near-threshold: a Marriage Made in Heaven

- 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)

Better to have N× PEs running at lower voltage than one PE at nominal voltage!
Multiple RI5CY Cores (1-16)
Low-Latency Shared TCDM

Tightly Coupled Data Memory

Mem
Mem
Mem
Mem
Mem
Mem
Mem
Mem

Logarithmic Interconnect

RISC-V core
RISC-V core
RISC-V core
RISC-V core

CLUSTER
High speed single clock logarithmic interconnect

Processors

Routing Tree

Arbitration Tree

Memory Banks

Ultra-low latency ➔ short wires + 1 clock cycle latency

World-level bank interleaving «emulates» multiported mem
**TCDM Interconnect**

- **Synchronous low latency crossbar based on binary trees**
  - Single or dual channel (Each with $2^n$ master ports, $n=[0,1,2 ...]$)
  - **Distributed Arbitration** (Round robin)
  - Combinational handshake (single phase)
  - Deterministic access latency (1 cycle) + response in second cycle
  - **Test&Set Supported**
  - Word-level Interleaving

- **Emulates Multiported RAMs**
  - Slaves are pure memories.
  - Bank conflicts can be alleviated → higher Banking Factor (BF=1,2,4)
Peripheral Interconnect

- Synchronous low latency crossbar based on binary trees
  - Single or dual channel Each with $2^n$ master ports, $n=\{0,1,2 \ldots\}$
  - Distributed Arbitration (Round robin)
  - Combinatorial handshake (single phase)
  - Custom port mapping (Address ranges)
  - Decoupled request and response path
    - Slaves have unpredictable latencies

- Used to build Peripheral systems
  - Slaves are pure generic peripherals like bridges, timers etc (Req/grant)
  - Mostly used to move data in and out from the cluster processors
FPU Interconnect (1/2)

- Synchronous low-power/low latency crossbar used to share several FPUs in a multicore system:
  - $2^n$ master ports (n=[0,1,2 ...])
  - $2^m$ slave ports (m=[0,1,2 ...]) \(\rightarrow\) General purpose FPUs(ADD/SUB, MUL etc)
  - Combinatorial handshake (single phase)
  - Allocator
    - Random: given a request, a random fpu is choosen
    - Optimal: Maxime the utilization of FPUs
FPU Interconnect

- **Features:**
  - Independent response paths for each sub-FPU block
    - fully pipelined FPU sub-blocks with different latencies
  - Two Operators (A,B), one command (OP) and the ID are carried to the FPU.
  - No need of Flow control on the FPU side.
  - Flexible and parametrizable

![Example of FPU attached to the FPU interconnect]

![Results normalized to private FPU cfg]

- Performance/Power
  - 8x8: 1.5, 8x4: 2, 8x2: 3
  - 16x16: 3.5, 16x8: 3.2, 16x4: 2.5, 16x2: 2

- Performance/Area
  - 8x8: 0.5, 8x4: 1, 8x2: 1.5
  - 16x16: 2, 16x8: 1.5, 16x4: 1, 16x2: 0.5

Results normalized to private FPU cfg
DMA for data transfers from/to L2

Tightly Coupled Data Memory

Mem
Mem
Mem
Mem
Mem
Mem

Logarithmic Interconnect

RISC-V core
RISC-V core
RISC-V core
RISC-V core

CLUSTER
PULP MCHAN DMA Engine

- A DMA engine optimized for integration in tightly coupled processor clusters
  - Dedicated, per-core non blocking programming channels
  - Ultra low latency programming (~10 cycles)
  - Small footprint (30Kgates) : avoid usage of large local FIFOs by forwarding data directly to TCDM (no store and forward)
  - Support for multiple outstanding transactions
  - Parallel RX/TX channels allow achieving full bandwidth for concurrent load and store operations

Configurable parameters:
- # of core channels
- Size of command queues
- Size of RX/TX buffer
- # of outstanding transactions

Integration in a PULP cluster
Fast synchronization and Atomics

Tightly Coupled Data Memory

L2 Mem

Interconnect

DMA

Logarithmic Interconnect

RISC-V core

RISC-V core

RISC-V core

RISC-V core

CLUSTER

Mem

Mem

Mem

Mem

Mem

Mem

Mem
Synchronization & Events

Avoid busy waiting!
Minimize sw synchro. overhead
Efficient fine-grain parallelization

Private, per core port
→ single cycle latency
→ no contention
PULP Cluster Event Unit

Cluster synchronization unit

- periph. event FIFO
- HW barrier
- HW mutex
- HW dispatch
- software event matrix
  - event logic
  - event logic
  - event logic

- DMA
- CNN acc.

- core 0
- core 1
- core N

event source single-cycle core accessible
Energy-efficient Event Handling

- Single instruction rules it all:
  - address determines action
  - read datum provides corresponding information

\[
evt\_data = evt\_read(addr);
\]

- event buffer content,
- program entry point
- mutex message
- triggering core id
- ...

- trigger sw event,
- trigger barrier,
- try mutex lock,
- read entry point,
- auto buffer clear?
- ...

- instruction turns off core clock
  \(\Rightarrow\) NO pipeline flush!
Results: Barrier

- Fully parallel access to SCU: Barrier cost constant
- Primitive energy cost: Down by up to 30x
- Minimum parallel section for 10% overhead in terms of …
  - … cycles: ~100 instead of > 1000 cycles
  - … energy: ~70 instead of > 2000 cycles
Results: Mutex

- Sequential execution: Cycle overhead always large
- TAS-variable inherently well-suited for mutex; lower cycle savings compared to barrier
- SCU still avoids L1 accesses: Energy of TAS mutex up to 1.6x higher
- Smallest parallel section for 10% energy overhead: ~1000 instead of 1600 cycles
How do we work: Initiate a DMA transfer
Data copied from L2 into TCDM

PULPissimo

Tightly Coupled Data Memory

Ext. Mem

L2 Mem

Mem Cont

RISC-V core

I/O

DMA

Event Unit

interconnect

I$
Once data is transferred, event unit notifies cores

![Diagram showing the interconnect and data flow between cores and memory.]
Cores can work on the data transferred
Once our work is done, DMA copies data back.
During normal operation all of these occur concurrently
Shared instruction cache with private “loop buffer”
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**

- **Read Energy [32 bit [pJ]]**
  - SCM
  - DOUG SAMPLED SCM
  - LOW VOLT. SRAM
  - LOW LEAK. SRAM
  - HIGH PERF. SRAM

- **Voltage [V]**
  - 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1

- **Area [–][mm^2]**
  - SCM
  - SPLL1CACHE
  - SPHD
  - SPREG

- **Scaling:**
  - 2x
  - 3.1x
  - 2.5x

- **Comparison:**
  - 2x-4x
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

Applications on PULP

Performance Degradation

ISSize [B]

BFS, CT, SLIC, FAST, MD, HOG
**Shared I$**

- **Shared 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)

- **Results**
  - Up to 40% better performance than private I$
  - Up to 30% better energy efficiency
  - Up to 20% better energy*area efficiency
Results: RV32IMCXpulp vs RV32IMC

- 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

Overall Speedup of 75x
Near-Linear Speedup

10x Speedup w.r.t. RV32IMC (ISA does matter 😊)
An additional I/O controller is used for IO

Ext. Mem
Mem Cont
L2 Mem
RISC-V core
I/O
PULPissimo

CLUSTER

Tightly Coupled Data Memory
Mem
Mem
Mem
Mem
Mem
Mem

Logarithmic Interconnect
RISC-V core
RISC-V core
RISC-V core
RISC-V core

I$
uDMA Subsystem

PERIPHERAL #ID

- CONFIG Registers
- PERIPH TX PROTOCOL
- PERIPH RX PROTOCOL

FILTER

- CONFIG Registers
- DSP

APB
UDMA_TX
UDMA_RX

cfg_data

tx_data

rx_data

2tx_data

stream

rx_data
Offload pipeline

- Efficient use of system resources
- HW support for double buffering allows continuous data transfers
- Multiple data streams can be time multiplexed
PULP interrupts controller (INTC)

- It generates interrupt requests from 0 to 31
- Mapped to the APB bus
- Receives events in a FIFO from the SoC Event Generator (i.e. from peripherals)
  - Unique interrupt ID (26) but different event ID
- Mask, pending interrupts, acknowledged interrupts, event id registers
- Set, Clear, Read and Write operations by means of load and store instructions (memory mapped operations)
- Interrupts come from:
  - Timers
  - GPIO (rise, fall events)
  - HWCE
  - Events i.e. uDMA
Tightly-coupled HW Compute Engine

Open Parallel Ultra-Low Power Platforms for Extreme Edge AI

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

- **Memory**
  - On the data plane, memory "sees" HWPEs as a set of SW cores

- **RISCY**
  - On the control plane, cores control HWPEs as a memory mapped peripheral (e.g., a DMA)

- **HWPE**

- **"Virtual" Core #N+1**

- **"Virtual" Core #N+2**

- **"Virtual" Core #N+3**

- **TCDM Interconnect**

- **Shared Memory**

- **Cores**

- **Peripheral Interconnect**

- **Address generation**

- **Data plane**

- **Control plane**

- **HWPE wrapper**

- **Control**

- **Register File + Control Logic**

"Virtual" Core #N+1 essentially refers to HWPEs on the data plane, which are seen as SW cores to the memory. Similarly, "Virtual" Core #N+2 and #N+3 on the control plane are treated as memory-mapped peripherals (e.g., a DMA) by the cores.

HWPEs provide a "Virtual" Core for each additional Core, with HWPE #N+1 being the first and #N+2 and #N+3 providing further layers of virtualization.

These HWPEs are part of the Open Parallel Ultra-Low Power Platforms for Extreme Edge AI, offering efficiency and flexibility in processing tasks at the edge of the network.
A toy HWPE

- Externally, uses memory accesses (master ports)
- Internally, uses streams

Shadow control register context

Peripheral access (target port)
HW Convolution Engine

4. Weights for each convolution filter are stored privately.

3. Allow “jobs” to be offloaded to the HWCE by the RISC-V cores.

2. Decouple the streaming domain from the shared memory domain; convert streams in 3D-strided memory accesses.

1. Perform convolve-accumulate in streaming fashion.

5. Fine-grain clock gating to minimize dynamic power consumption.

HWCE Sum-of-Products

Each input channel pixel is read $N_{outFeatures}$ times.

Each weight is read once.

For each out_feature $y$:

For each in_feature $x$:

$y += \text{conv}(W,x)$

Linebuffer: min-bandwidth sliding window

$y_{in}$: 1x 16-bit pixel

$W$: 5x5 16-bit filters


$y_{out}$: 1x 16-bit pixel

$NORM$ $SAT$

Each output channel's partial sum pixel is read $(N_{inFeatures} - 1)$ times.

SoP
Need μW-range always-on Intelligence

Tightly Coupled Data Memory

Logarithmic Interconnect

HWCE

RISC-V core

RISC-V core

RISC-V core

RISC-V core

I$

I$

I$

I$

Smart Wakeup Module
HD-Based smart Wake-Up Module

Ext. Mem
Mem Cont
L2 Mem
RISC-V core
I/O

PULPissimo

I/O
Preprocessor
Autonomous HD-Computing Unit

Always-on Domain

Ext. ADC

Tightly Coupled Data Memory
Mem
Mem
Mem
Mem
Mem

Logarithmic Interconnect
RISC-V core
RISC-V core
RISC-V core
RISC-V core

I$
HD-Based smart Wake-Up Module

PULPissimo

Ext. Mem

Mem Cont

L2 Mem

RISC-V core

I/O

Tightly Coupled Data Memory

Mem

Mem

Mem

Mem

Mem

DMA

Mem

Mem

Mem

Mem

Mem

Logarithmic Interconnect

RISC-V core

RISC-V core

RISC-V core

RISC-V core

I$

I$

I$

I$

CLUSTER

Wake Up

Ext. ADC

I/O (SPI)

Preprocessor

Autonomous HD-Computing Unit

Always-on Domain
# HD-Based smart Wake-Up Module

### Results (post-P&R) - TO done

<table>
<thead>
<tr>
<th>Technology</th>
<th>GF22 UHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>670kGE</td>
</tr>
<tr>
<td>Max. Frequency</td>
<td>3 MHz</td>
</tr>
<tr>
<td>SCM-Memory</td>
<td>32 kBit</td>
</tr>
<tr>
<td>Module Power</td>
<td>~ 15uW</td>
</tr>
<tr>
<td>Consumption</td>
<td></td>
</tr>
<tr>
<td>(@ 1 kSPS/channel, 3 channels)</td>
<td></td>
</tr>
</tbody>
</table>
PULP includes Cores+Interco+IO+HWCE → Open Platform

**RISC-V Cores**
- RI5CY 32b
- Ibex 32b
- Snitch 32b
- Ariane + Ara 64b

**Peripherals**
- JTAG
- UART
- SPI
- I2S
- DMA
- GPIO

**Interconnect**
- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

**Platforms**
- **Single Core**
  - PULPino
  - PULPissimo
- **Multi-core**
  - Fulmine
  - Mr. Wolf
- **Multi-cluster**
  - Hero
  - Open Piton

**Accelerators**
- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st ord. opt)
Nice, but what exactly is “open” in Open Source HW?

A modern, open, free ISA, extensible by construction
Endorsed and Supported by 300+ Companies

But… an open ISA is not Open HW (it’s a prerequisite!)
Nice, but what exactly is “open” in Open Source HW?

- Only the first stage of the silicon production pipeline can be open HW → **RTL source code** (in an HDL such as SystemVerilog)
- Later stages contain closed IP of various actors + tool licensing issues

Permissive, Copyright (e.g APACHE) License is Key for industrial adoption
PULP is silicon proven

37 SoCs & counting
Luca Benini, Davide Rossi, Andrea Borghesi, Michele Magno, Simone Benatti, Francesco Conti, Francesco Beneventi, Daniele Palossi, Giuseppe Tagliavini, Antonio Pullini, Germain Haugou, Manuele Rusci, Florian Glaser, Fabio Montagna, Bjoern Forsberg, Pasquale Davide Schiavone, Alfio Di Mauro, Victor Javier Kartsch Morinigo, Tommaso Polonelli, Fabian Schuiki, Stefan Mach, Andreas Kurth, Florian Zaruba, Manuel Eggimann, Philipp Mayer, Marco Guermandi, Xiaying Wang, Michael Hersche, Robert Balas, Antonio Mastrandrea, Matheus Cavalcante, Angelo Garofalo, Alessio Burrello, Gianna Paulin, Georg Rutishauser, Andrea Cossettini, Luca Bertaccini, Maxim Mattheeuws, Samuel Riedel, Sergei Vostrikov, Vlad Niculescu, Hanna Mueller, Matteo Perotti, Nils Wistoff, Luca Bertaccini, Thorir Ingulfsson, Thomas Benz, Paul Scheffler, Alessio Burello, Moritz Scherer, Matteo Spallanzani, Andrea Bartolini, Frank K. Gurkaynak, and many more that we forgot to mention

http://pulp-platform.org  @pulp_platform