Leveraging the PULP Platform to Build Reliable Systems

Luca Bertaccini (ETH Zurich) <lbertaccini@iis.ee.ethz.ch>
Michael Rogenmoser (ETH Zurich) <michaero@iis.ee.ethz.ch>
Agenda


- Michael Rogenmoser (ETHZ): “Adding Reliability Features to PULP Systems”
Agenda

- Michael Rogenmoser (ETHZ): “Adding Reliability Features to PULP Systems”
A reliable cognitive computing node:

- AI capabilities in the power envelope of an MCU
- Reliability features on the edge device
Edge Processing

Sensors

Signals

Processing

Transmission

100μW - 1mW

1mW - 10mW

1mW (idle) - 50mW (active)
How can we build an energy-efficient IoT end node with AI capabilities?
Energy efficiency @ GOPS is the Challenge

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

How??

*data from ARMs web
RISC-V: an Open Extensible ISA

A modern, open, free ISA, extensible by construction
Endorsed and Supported by 600+ Members
Changed the picture on Computing Systems Research!
RI5CY Processor: in-order 4-stage Core

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
RI5CY ISA extensions improve ML performance

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

8-bit workload

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Auto-incr load/store</th>
<th>HW Loop</th>
<th>Packed-SIMD</th>
</tr>
</thead>
</table>
| mv x5, 0
mv x4, 100
Lstart:
  lb x2, 0(x10)
  lb x3, 0(x11)
  addi x10, x10, 1
  addi x11, x11, 1
  add x2, x3, x2
  sb x2, 0(x12)
  addi x4, x4, -1
  add x2, x3, x2
  sb x2, 0(x12!)
  bne x4, x5, Lstart | mv x5, 0
mv x4, 100
Lstart:
  lb x2, 0(x10!)
  lb x3, 0(x11!)
  addi x4, x4, -1
  add x2, x3, x2
  sb x2, 0(x12!)
  bne x4, x5, Lstart | `lp.setup 100, Lend`
  lb x2, 0(x10!)
  lb x3, 0(x11!)
  add x2, x3, x2
  Lend: sb x2, 0(x12!)
  bne x4, x5, Lstart | `lp.setup 25, Lend`
  lw x2, 0(x10!)
  lw x3, 0(x11!)
  `pv.add.b` x2, x3, x2
  Lend: `sw` x2, 0(x12!)
  bne x4, x5, Lstart |

11 cycles/output 8 cycles/output 5 cycles/output 1,25 cycles/output

RI5CY ISA extensions improve ML performance
8-bit workload
RI5CY Processor: in-order 4-stage Core

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!
Better to have N× PEs running at optimum Energy than 1 PE running fast at low Energy efficiency

- 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)
PULP cluster contains multiple RISC-V cores (1-16)
All cores can access all memory banks in the cluster.
Data is copied from a higher level through DMA
A (shared) instruction cache fetches from L2
Hardware Accelerators can be added to the cluster.
Event unit to manage resources (fast sleep/wakeup)
A microcontroller system (PULPissimo) for I/O
How do we work: Initiate a DMA transfer
Data copied from L2 into TCDM
Once data is transferred, event unit notifies cores/accel
Cores can work on the data transferred
Accelerators can work on the same data
Once our work is done, DMA copies data back
During normal operation all of these occur concurrently

Open-source: github.com/pulp-platform/pulp
Results: RV32IMCXpulp vs RV32IMC (DNN)

- 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
  - 2 orders of magnitude with DOTP+LW (122x)
  - Sub-byte (nibble, crumb) supported (537x, 939x)

Overall Speedup of 75x
Near-Linear Speedup
10x Speedup w.r.t. RV32IMC (ISA does matter 😊)
PULP: Cores + Interco + IO + HWCE → Open Platform

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

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

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

**Platforms**
- Single Core
  - PULPino
  - PULPissimo
- Multi-core
  - Open-PULP
  - PULP-PM
- Multi-cluster
  - Hero
  - MANTICORE

**Accelerators**
- ML-HWCE (inference)
- Neurostream (ML training)
- HWCrypt (crypto)
- PULPO (1st ord. opt)

https://github.com/pulp-platform/
How can we leverage this open-source platform to build reliable systems?
Agenda


- Michael Rogenmoser (ETHZ): “Adding Reliability Features to PULP Systems”
A Low-Power Fractal End Node

A reliable cognitive computing node:

- AI capabilities in the power envelope of an MCU
- Reliability features on the edge device

Supported in part by the European Union’s H2020 Fractal #877056

https://fractal-project.eu
Critical & Hostile Environments

Space Environment

Automotive

Particle Accelerator

And many more…
Leverage the PULP cluster
Leverage the PULP cluster
Protecting the Cores

System

Memory

Core Inputs
(instructions, data, ...)

Core Outputs
(request, status, ...)

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core
Protecting the Cores

Core Inputs
(instructions, data, ...)

Core Outputs
(request, status, ...)

System

Memory

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core
Protecting the Cores

Core Inputs
(instructions, data, ...)

System

Memory

Voter

RISC-V core

Core Outputs
(request, status, ...)

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core
Triple-Core Lock Step

Core Inputs
(instructions, data, ...)

System

Memory

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

Voter

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

Voter

Core Outputs
(request, status, ...)

38
Re-synchronization

Core Inputs
(instructions, data, ...)

Core Outputs
(request, status, ...)

Radiation causes soft-error

Error detected by Voter

Core state stored to Memory

Core state loaded back into cores
Redundancy Requirement

- **Mission-Critical Application**
  - Must be correct

- **Data Processing Algorithm**
  - Individual errors can be tolerated
On-Demand Redundancy Grouping

Core Inputs
(instructions, data, ...)

System

Memory

Voter

RISC-V core

RISC-V core

RISC-V core

Voter

RISC-V core

RISC-V core

RISC-V core

Core Outputs
(request, status, ...)
On-Demand Redundancy Grouping

- **Core Inputs** (instructions, data, ...)
- **Core Outputs** (request, status, ...)

- **System**
  - **Memory**
  - **RISC-V core**
  - **Voter**
  - **0**

- **Voter**
  - **0**
  - **0**
  - **0**

- **RISC-V core**
  - **RISC-V core**
  - **RISC-V core**
  - **RISC-V core**
  - **RISC-V core**
On-Demand Redundancy Grouping

Core Inputs
(instructions, data, ...)

Core Outputs
(request, status, ...)

System

Memory

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core

RISC-V core

Voter

Voter

Redundant Mode
On-Demand Redundancy Grouping

Core Inputs
(instructions, data, ...)

System

Memory

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

Voter

0
0

Redundant Mode

Core Outputs
(request, status, ...)

Performance Mode
On-Demand Redundancy Grouping

- **Up to 3x speedup** for non-critical tasks
- **<60’000 cycles** to switch modes
- **<1% area overhead** for the cluster
  - ODRG for 3 cores requires ~11% area of a single core

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Cycles</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit 2D Convolution</td>
<td>150</td>
<td>2.77×</td>
</tr>
<tr>
<td>32-bit 24x24 MatMul</td>
<td>100</td>
<td>2.96×</td>
</tr>
<tr>
<td>32-bit 32x32 MatMul</td>
<td>50</td>
<td>2.63×</td>
</tr>
</tbody>
</table>

- soft-error tolerant
- performance
Leverage the PULP cluster
Leverage the PULP cluster
Leverage the PULP cluster
Memory Protection Options

- No Protection
- TMR
- Hsiao Error Correcting Codes:
  - Single Error Correction, Double Error Detection (SECDED)
  - 8-bit ECC
  - 16-bit ECC
  - 32-bit ECC

32-bit word
+200% +62.5% +37.5% +21.9%
ECC Load-and-Store

Byte Store

Byte Store with ECC
L1 Memory

CoreMark Store Instructions

- 32-bit ECC Used
- Buffer storage operation
  - Delay following transaction, not current transaction, to shift & reduce impact
- <1% cycle increase
  - Various tests, such as 8-bit Matrix-Matrix Multiplication
## ECC Scrubber

- **Scan Memory Bank**
- **Re-write faulty word if error is detected**
- **Defer permission to external accesses**

<table>
<thead>
<tr>
<th>Address</th>
<th>Hex</th>
<th>Hex</th>
<th>Hex</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0x0001</td>
<td>0x0002</td>
<td>0x0003</td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td>0x0005</td>
<td>0x0006</td>
<td>0x0007</td>
<td></td>
</tr>
<tr>
<td>0x0008</td>
<td>0x0009</td>
<td>0x000A</td>
<td>0x000B</td>
<td></td>
</tr>
<tr>
<td>0x000C</td>
<td>0x000D</td>
<td>0x000E</td>
<td>0x000F</td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>0x0011</td>
<td>0x0012</td>
<td>0x0013</td>
<td></td>
</tr>
<tr>
<td>0x0014</td>
<td>0x0015</td>
<td>0x0016</td>
<td>0x0017</td>
<td></td>
</tr>
<tr>
<td>0x0018</td>
<td>0x0019</td>
<td>0x001A</td>
<td>0x001B</td>
<td></td>
</tr>
<tr>
<td>0x001C</td>
<td>0x001D</td>
<td>0x001E</td>
<td>0x001F</td>
<td></td>
</tr>
<tr>
<td>0x0020</td>
<td>0x0021</td>
<td>0x0022</td>
<td>0x0023</td>
<td></td>
</tr>
<tr>
<td>0x0024</td>
<td>0x0025</td>
<td>0x0026</td>
<td>0x0027</td>
<td></td>
</tr>
</tbody>
</table>
Leverage the PULP cluster

Ext. Mem

Mem Cont

L2 Mem

RISC-V core

I/O

PULPissimo

interconnect

Tightly Coupled Data Memory

Mem

Mem

Mem

Mem

Mem

interconnect

DMA

Event Unit

HW accel

RISC-V core

RISC-V core

RISC-V core

RISC-V core

I$

I$

I$

I$
Ongoing Projects

- Fault-tolerant Host Processor
  - Triple-core PULPissimo
- Interconnect Protection
- Instruction Cache Protection
- Fault-tolerant Peripherals
- Adding a Watchdog Timer
Redundancy Cells

This repository contains various modules used to add redundancy.

On-Demand Redundancy Grouping (ODRG_unit)

The ODRG_unit is designed as a configurable bridge between three ibex cores, allowing for independent operation or lock-step operation with majority voting, triggering an interrupt in case a mismatch is detected. It uses lowrisc's regen tool to generate the required configuration registers.

Testing

ODRG is integrated in the PULP cluster and the PULP system. To test, please use the space_pulp branch.

Try it out on Github!

www.github.com/pulp-platform
Luca Benini, Alessandro Capotondi, Alessandro Ottaviano, Alessio Burrello, Alfio Di Mauro, Andrea Borghesi, Andrea Cossettini, Andreas Kurth, Angelo Garofalo, Antonio Pullini, Arpan Prasad, Bjoern Forsberg, Corrado Bonfanti, Cristian Cioflan, Daniele Palossi, Davide Rossi, Fabio Montagna, Florian Glaser, Florian Zaruba, Francesco Conti, Georg Rutishauser, Germain Haugou, Gianna Paulin, Giuseppe Tagliavini, Hanna Müller, Luca Bertaccini, Luca Valente, Luca Colagrande, Manuel Eggimann, Manuele Rusci, Marco Guermandi, Matheus Cavalcante, Matteo Perotti, Matteo Spallanzani, Michael Rogenmoser, Moritz Scherer, Moritz Schneider, Nazareno Bruschi, Nils Wistoff, Pasquale Davide Schiavone, Paul Scheffler, Philipp Mayer, Robert Balas, Samuel Riedel, Sergio Mazzola, Sergei Vostrikov, Simone Benatti, Stefan Mach, Thomas Benz, Thorir Ingolfsson, Tim Fischer, Victor Javier Kartsch Morinigo, Vlad Niculescu, Xiaying Wang, Yichao Zhang, Frank K. Gürkaynak, all our past collaborators and many more that we forgot to mention