nano-tera.ch
swiss scientific initiative in health / security / environment systems

IcySoC RTD 2013

FNSNF

Approximate Computing Units for an Ultra-Low Power Platform

Michael Gautschi[1], Antonio Pullini[1], Frank K. Gürkaynak[1], Luca Benini[1]

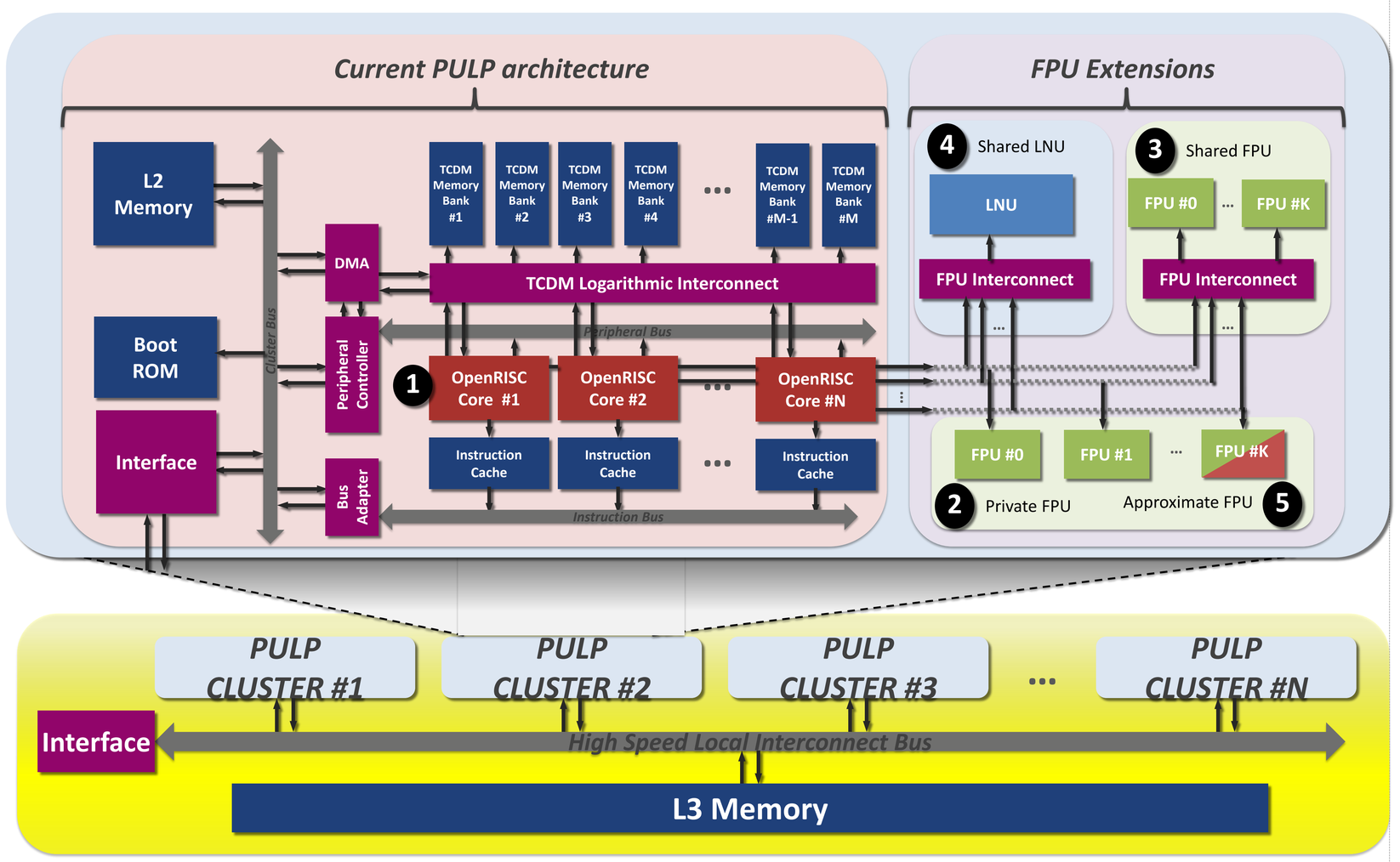
ETH Zürich

[1]Integrated Systems Laboratory, ETH Zürich

PULP Architecture

PULP stands for **P**arallel **U**ltra-**L**ow **P**ower Processor Architecture and is actively being developed by the Integrated Systems Laboratory of ETH Zürich. Our goal is to develop a system that has the same energy efficiency regardless of the computational load. We call this property **energy proportionality**. Our system works very well when there is little to do but is equally efficient when the work load increases. This is different form other processor systems which are optimized to work well at one corner, but do not scale well.

- Our many-core architecture is organized in clusters. Each cluster consists of simple RISC cores which have been optimized for the cluster. Our current core is based on the OpenRISC ISA from opencores.org.
- A shared L1 memory, so-called Tightly-Coupled Data-Memory (TCDM) is used to efficiently share data structures.
- To add floating point support we have designed an FPU which can be used in a private and a shared setting in combination with an interconnect.
- In addition, we have looked in FPU approximation and a logarithmic number system.

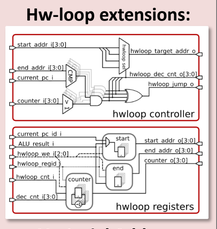


Instruction Extensions

1 **OpenRISC Instruction-Set Extensions:**
Several ISA-extensions have been analyzed and implemented in order to increase computational efficiency of the OR10N core.

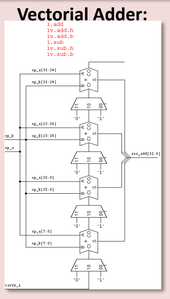
Hardware loops:
- Allow to get rid of loop overhead (branch, compare instructions) in regular loop structures, such as for-loops.
- A hardware loop is fully defined by:
  - Start address
  - End address
  - Number of iterations
- Setup possible in only one instruction

Hw-loop extensions:



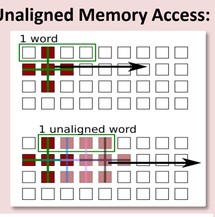
Pre-/post increment memory addressing:
- Effective memory address (EA) is computed:
  EA = rA + signext(offset)
- With auto-incrementation it is possible to store this address in the register file.
- Allows to update counters, addresses, etc. in parallel.

Vectorial Adder:

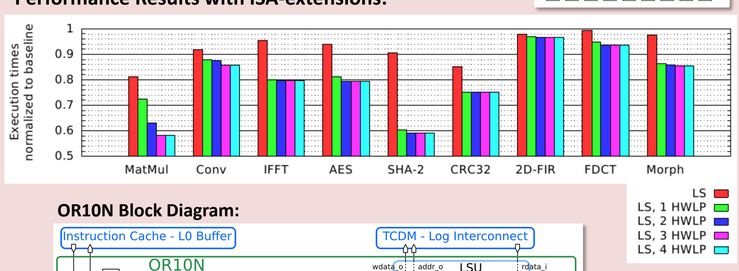


Vector ALU Unit:
- Possible to operate on byte, and halfword level
- Unaligned memory access implemented with two subsequent memory requests.
  => No hardware overhead

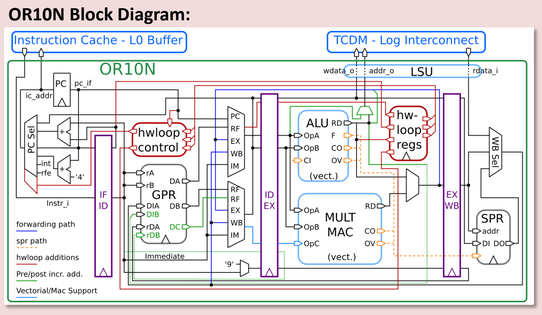
Unaligned Memory Access:



Performance Results with ISA-extensions:



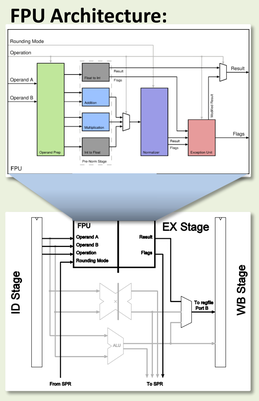
OR10N Block Diagram:



Floating Point Units

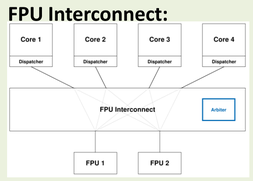
2 **FPU Design:**
- Implementation of a FPU to be integrated in the Or10n core.
- Supported FPU instructions: (IEEE single precision):
  - mult, add, sub, comp, type casts
- Fused data path for minimal hardware costs.

FPU Architecture:

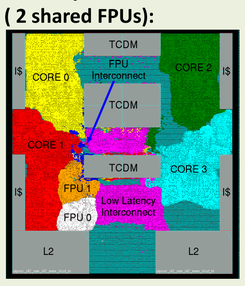


3 **Private vs shared FPU:**
- FPUs are not utilized 100% and therefore remain idle for a large number of cycles.
- Sharing FPUs in hardware leads to better utilization of the computation units, and is not seen by the application designer.
- *M* different FPUs can be shared among *N* cores using our developed interconnect which guarantees perfect and fair arbitration while minimizing its area and latency.

FPU Interconnect:



Final Layout of Hekate (2 shared FPUs):



5 **The case for approximate FPUs:**
- If full precision is not required it is either possible to save in area or delay.
- Together with ICLAB we have designed three different approximate FPUs and integrated them in a 65nm chip called Diana.

Summary:

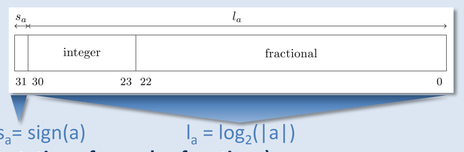
	FPU Area	Total Area	Timing	FPU-Latency
Artemis Private FPU	48 kGE (12kGE/FPU)	563 kGE	300 MHz	2 cycles
Hecate Shared FPU	22 kGE (11kGE/FPU)	557 kGE	300 MHz	3 cycles
Diana Approximate FPU	38 kGE (11/10/9/8) kGE	553 kGE	300 MHz	2 cycles

Logarithmic Number Unit

4 The logarithmic number system can be used to exploit a larger dynamic range. The format has its pros and cons, but leads to attractive results. We have developed a Logarithmic Number Unit (LNU) which achieves IEEE single precision and can be used as a replacement to a FPU.

Float representation:
$a = (-1)^{sa} * (1+m) * 2^{exp}$

LNS representation:
$a = (-1)^{sa} * 2^{la}$



s_a= sign(a) $l_a = \log_2(|a|)$

Advantages: (single cycle computation of complex functions)
Simple multiplication, division, powering which can be computed with the processors integer unit!

$l_{mul} = \log_2(x*y) = \log_2(2^{lx} * 2^{ly}) = l_x + l_y$

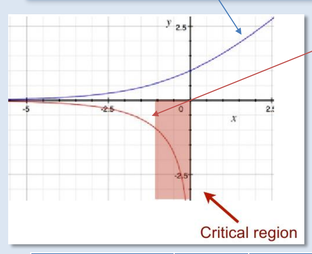
$l_{div} = \log_2(x/y) = \log_2(2^{lx} / 2^{ly}) = l_x - l_y$

powering: $x^{123.74} => l_x * 123.74$

Disadvantages: (non linear functions)
Addition and subtraction and type casts are non linear and have to be approximated!

Addition:
$l_{res} = l_x + \log_2(1+2^{ly-lx})$

Subtraction:
$l_{res} = l_x + \log_2(1-2^{ly-lx})$

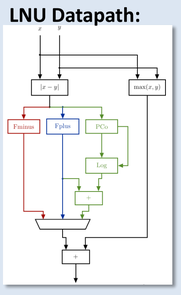


F_{plus}:
- 2nd order Taylor

F_{minus}(not critical):
- 2nd order Taylor

Fminus (critical):
- *Cotransformation* to eliminate the critical region

LNU Datapath:



	Timing	Area	Add/Sub	Typecast
ELM [1] 180 nm	13.15 ns	905'000 µm² 97 kGE	✓	
Rom-less LNS [2] 180 nm	14.6 ns	584'000 µm² 62 kGE	✓	
Selene, this work 65nm	4.35 ns	72'000 µm² 63 kGE	✓	✓

Can be efficiently shared in a cluster! -> Selene

[1] The European Logarithmic Microprocessor, J.N Coleman et.al, 2008
[2] ROM-less LNS, R.Che Ismail and J.N Coleman, 2011

Timeline for ASICs

as part of the IcySoC project ...