PULPino: A RISC-V based single-core system

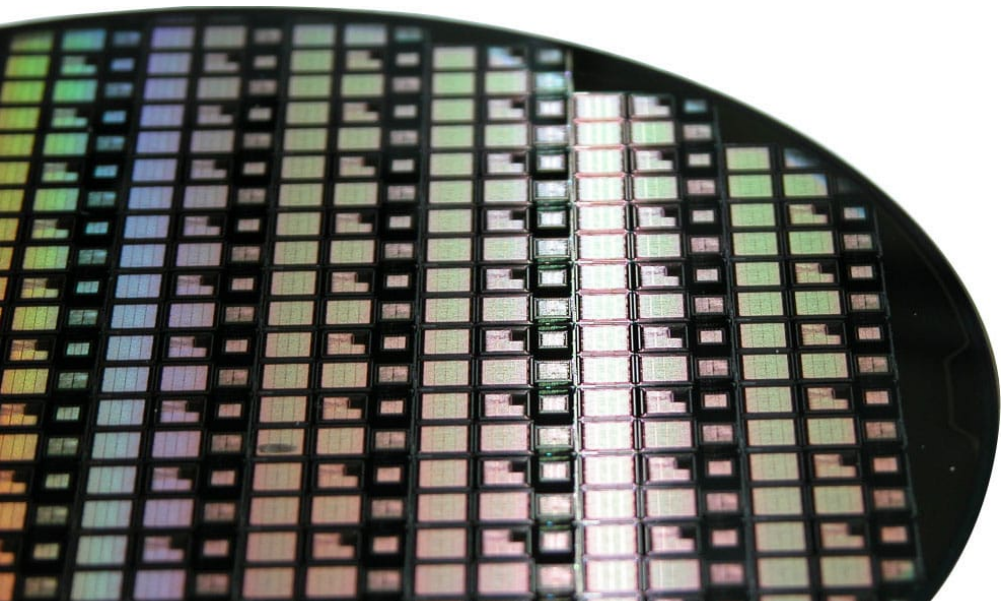
ORCONF 2015

Geneva, October 2015

Andreas Traber
Sven Stucki
Florian Zaruba
Michael Gautschi
Antonio Pullini
Igor Loi
Davide Rossi
Germain Haugou
Frank Kagan Gürkaynak
Luca Benini

ETH zürich
Integrated Systems Laboratory

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA



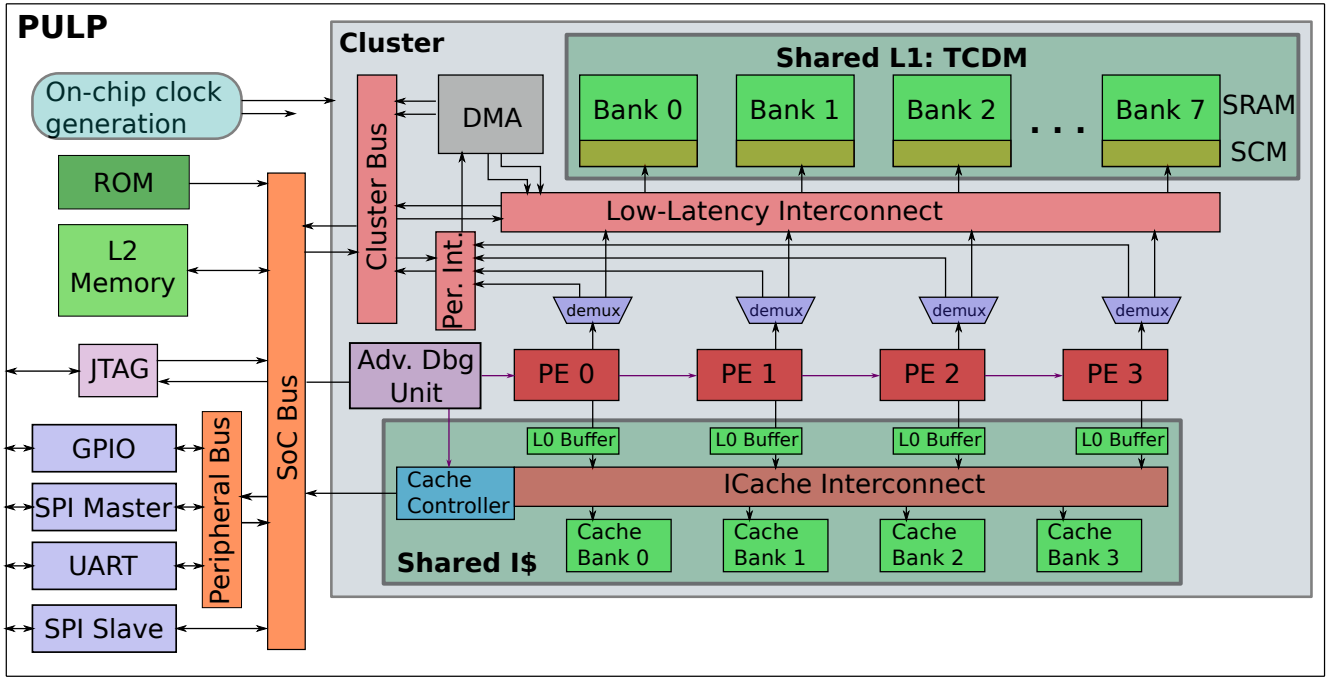
PULPino

A RISC-V based single-core system

- PULP: Parallel Ultra-Low power processing Platform
  - Energy efficient many core SoC
  - Software frameworks (OpenMP, OpenVX, ...)
  - Custom toolset (Virtual Platform, profiling tools, ...)
  - Large number of IPs
- Plan to publish PULP as open source
  But PULP is a huge project
- PULPino as a first step
  Back to basics, focus on core again

From PULP to PULPino

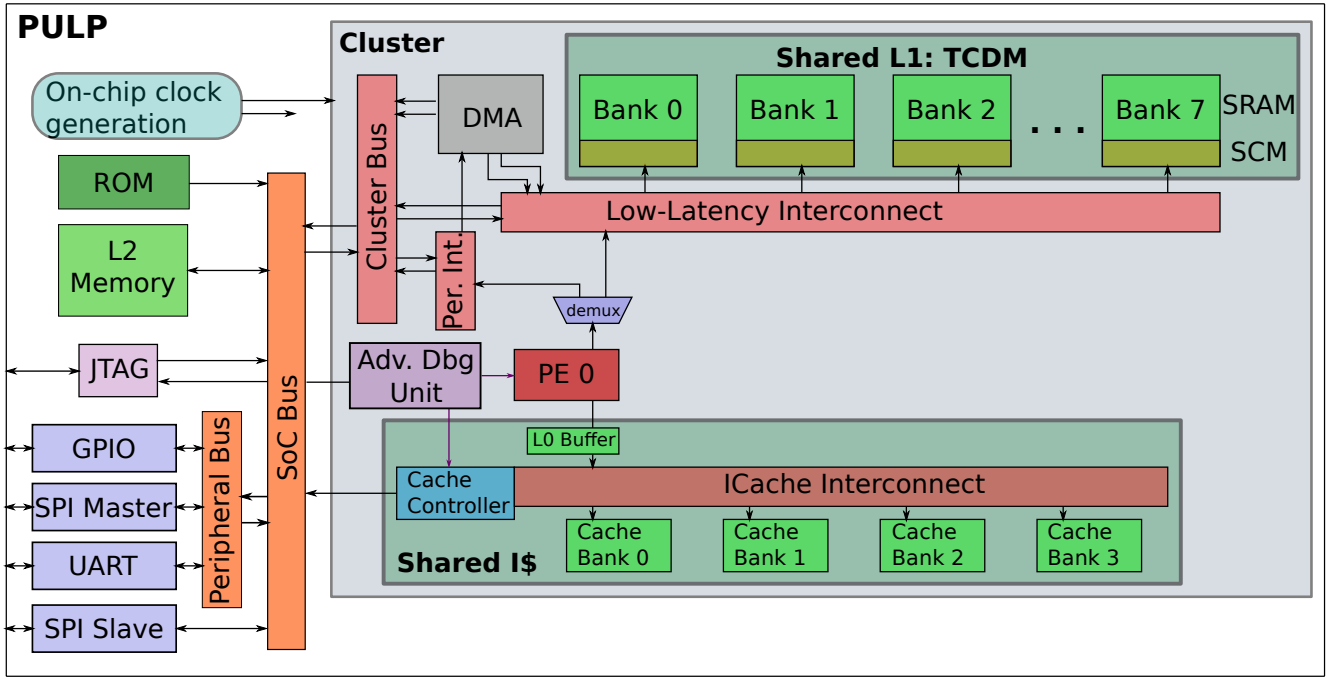
Background



- PULP: Parallel Ultra-Low power Processor
  Complex system that achieves extreme energy efficiency

From PULP to PULPino

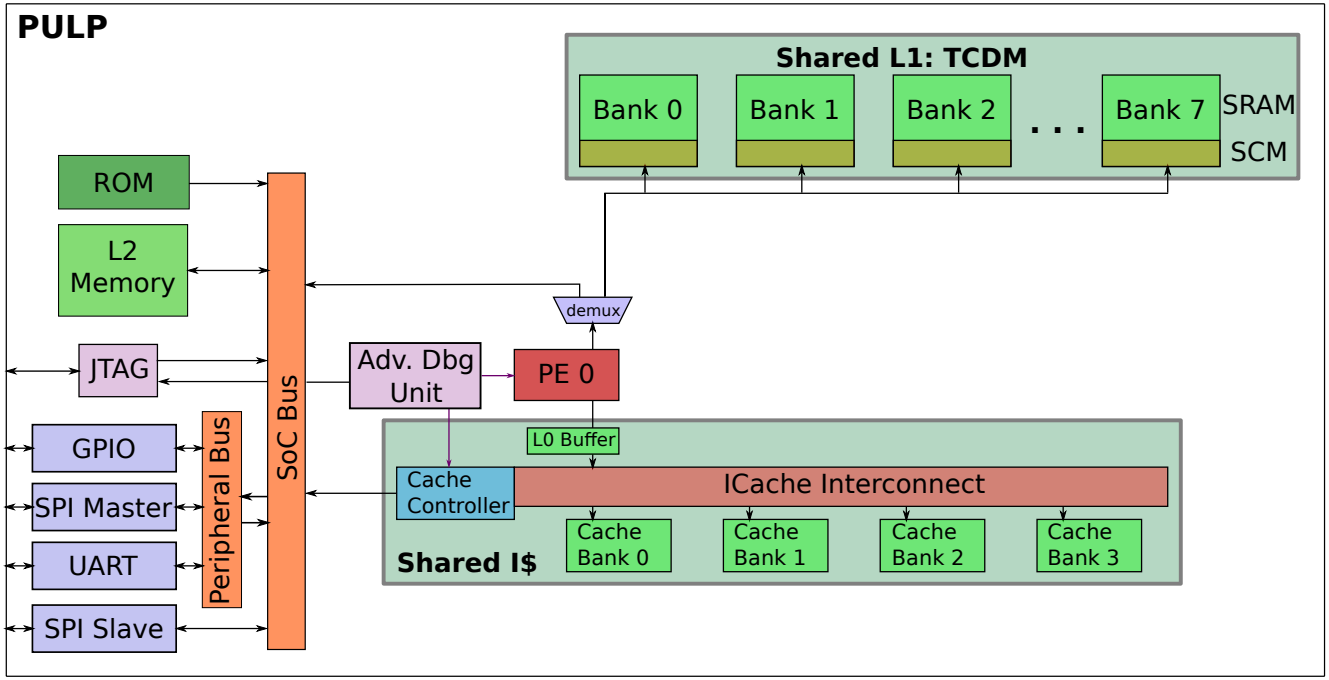
Background



- Move to a single core system

From PULP to PULPino

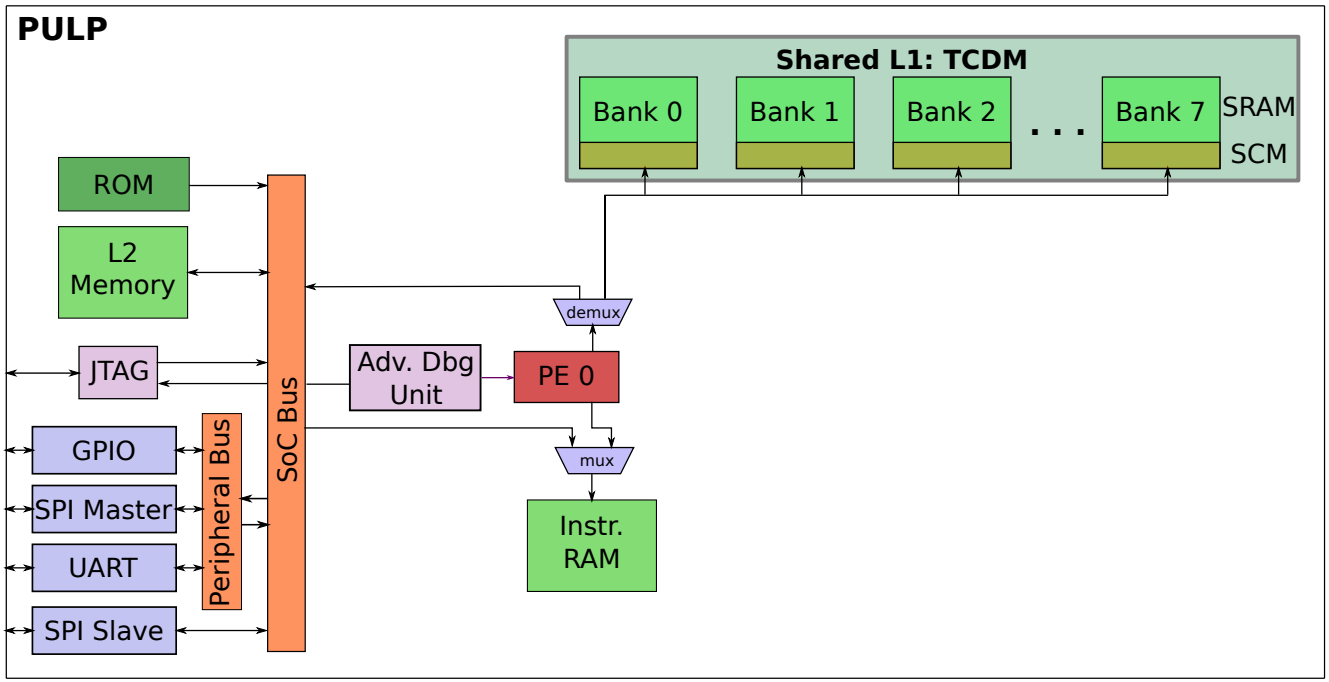
Background



- Remove cluster specific components
  No longer needed for a single core

From PULP to PULPino

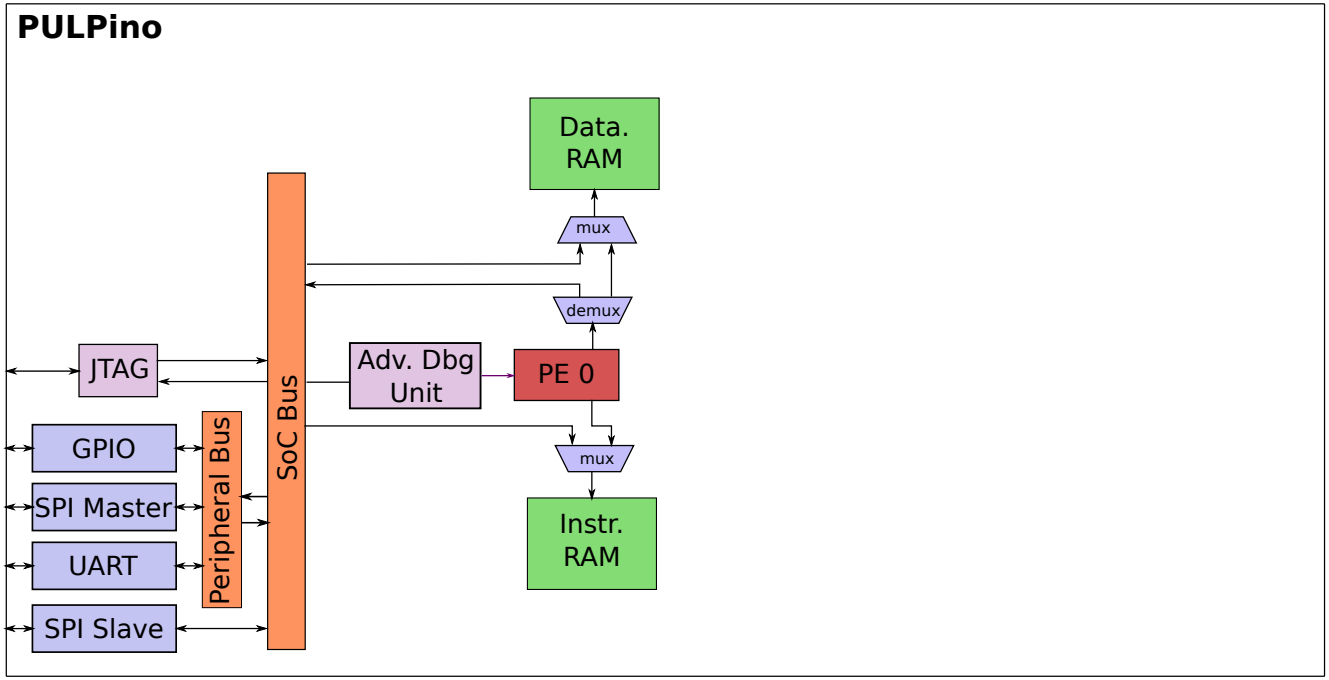
Background



- Replace instruction cache with instruction RAM
  Single-cycle instruction RAM access

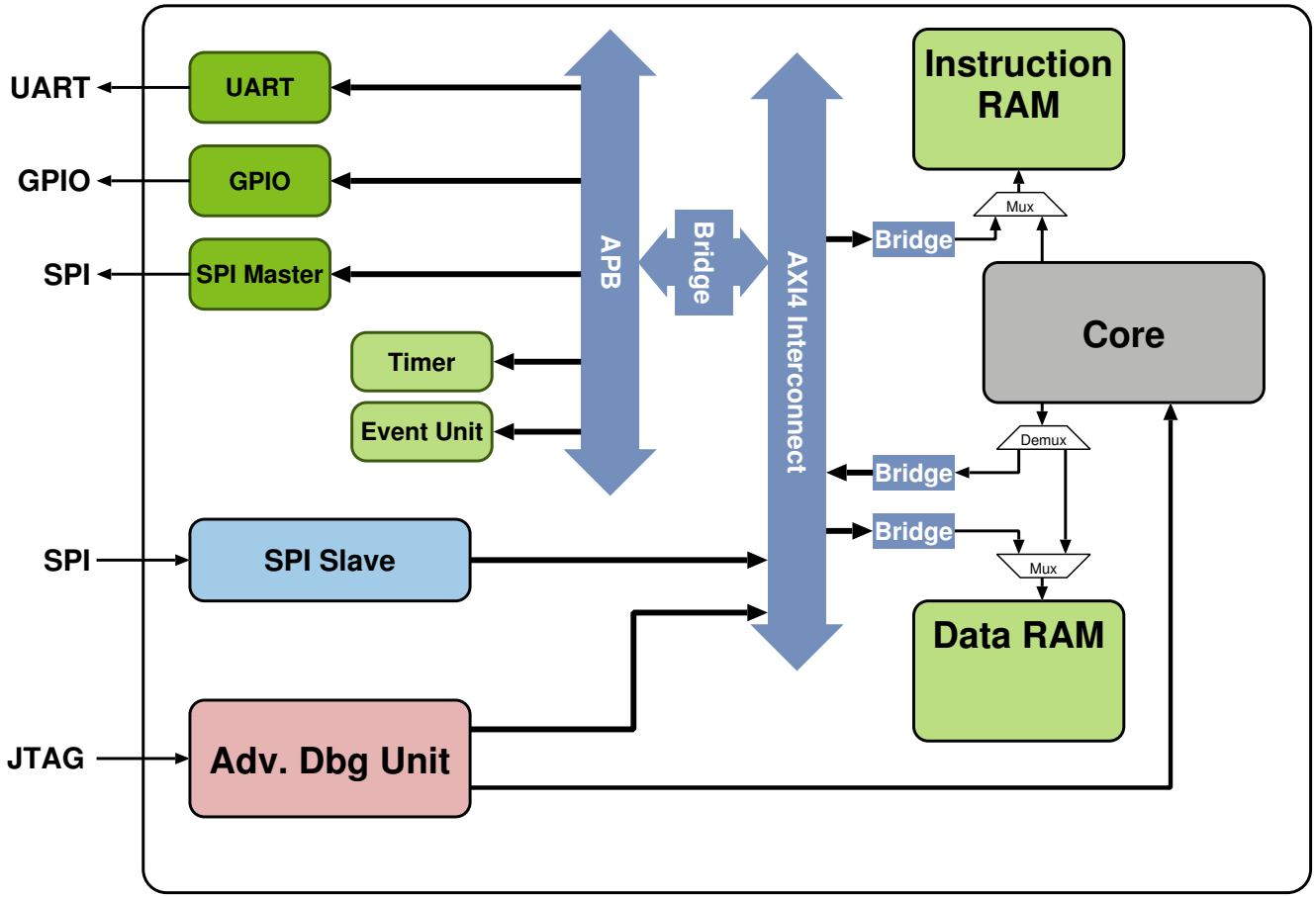
From PULP to PULPino

Background



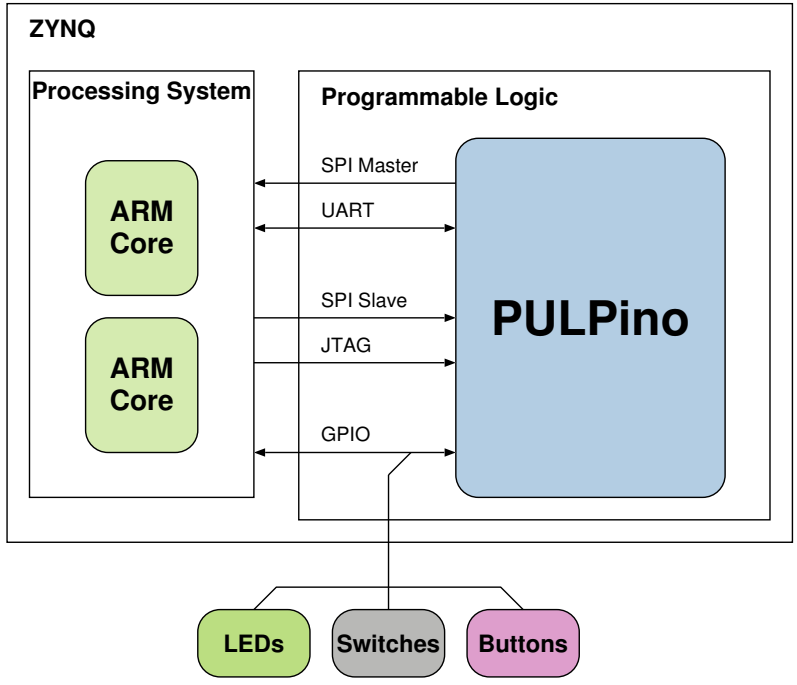
- Remove L2 and replace TCDM with data RAM
  Single-cycle data RAM access

PULPino uses IPs built for PULP



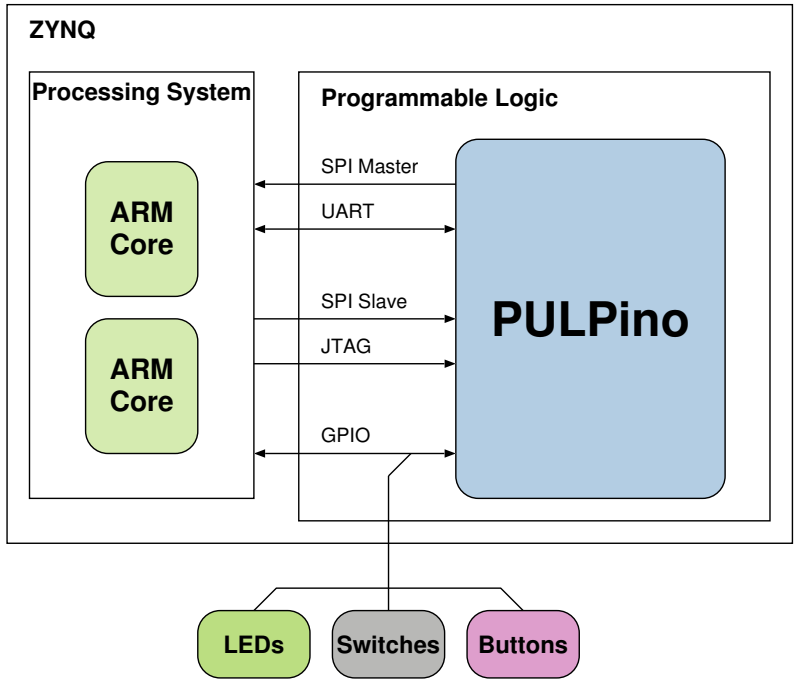
- Central AXI bus
  APB for peripherals
- Standard set of peripherals
  SPI, UART, GPIOs

PULPino for FPGA & ASIC



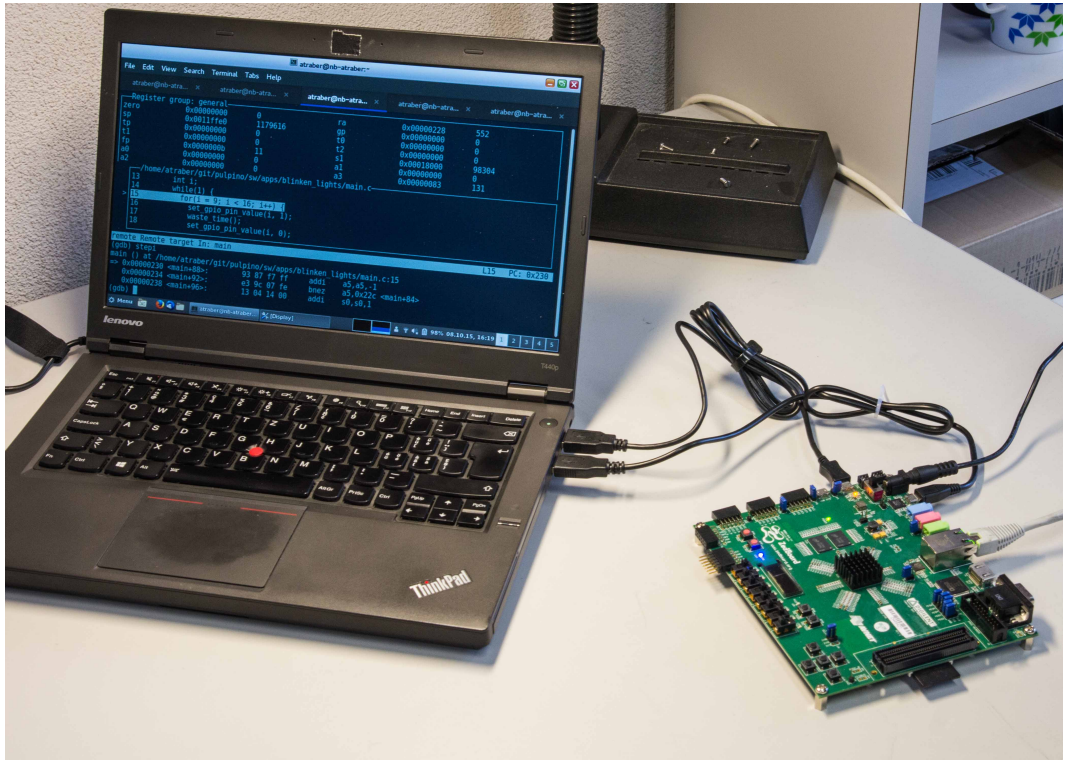
- FPGA implementation available
  Running on a ZedBoard
- Program PULPino via SPI or JTAG
  Communicate via UART or SPI

PULPino for FPGA & ASIC



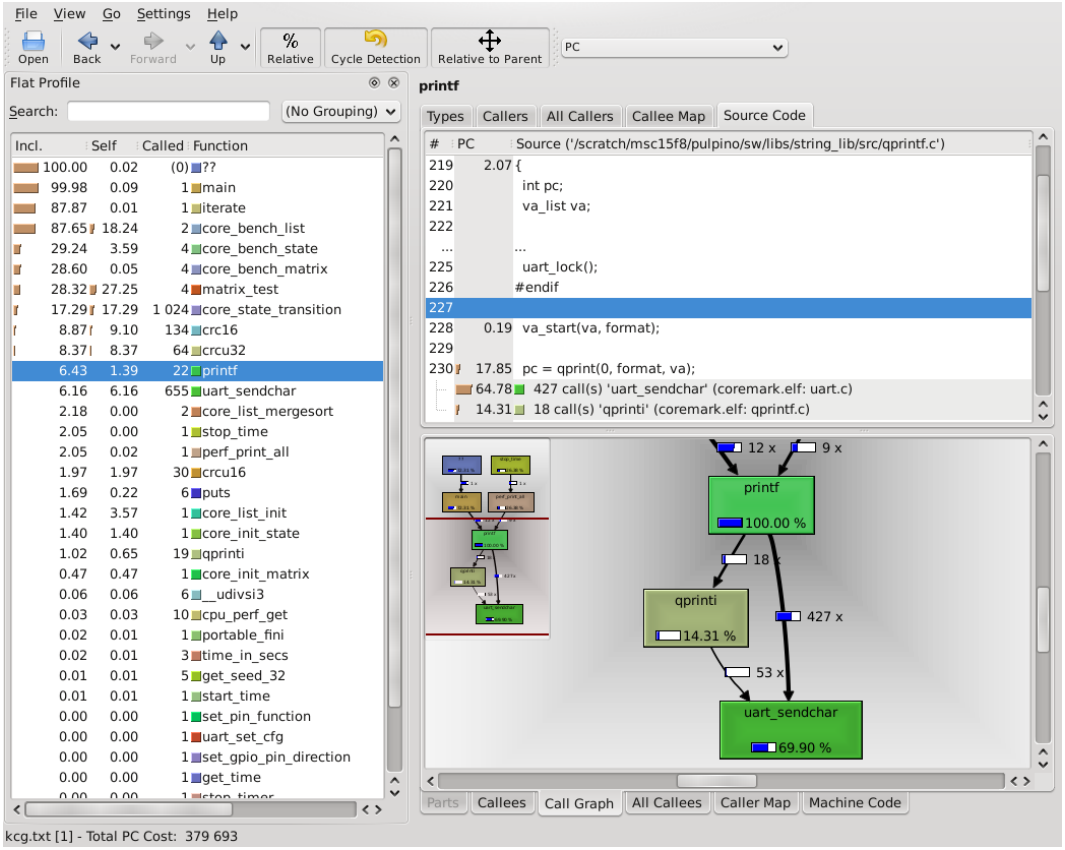
- FPGA implementation available
  Running on a ZedBoard
- Program PULPino via SPI or JTAG
  Communicate via UART or SPI
- ASIC tapeout planned for January 2016
  Student project using UMC 65nm

Interactive Debugging via GDB



- Debugging the core via JTAG on the FPGA
  JTAG is controlled by debug bridge running on the ZYNQ
- GDB can be run remotely and connects to the bridge
  Easy debugging on your laptop

Software Profiling using KCacheGrind



- Capture instruction traces during simulation
  Also includes performance counter values
- Analyze program execution in KCacheGrind
  Includes source and machine code views, call graphs, . . .

RI5CY: Optimized RISC-V core

- Developed at IIS for the use in PULP
- Explore the RISC-V ISA
  Compare to our existing OpenRISC core
- Focus on energy efficiency
  Keep it small and simple
- RI5CY is comparable to a Cortex M4
  In terms of area and performance
- Entirely written in SystemVerilog

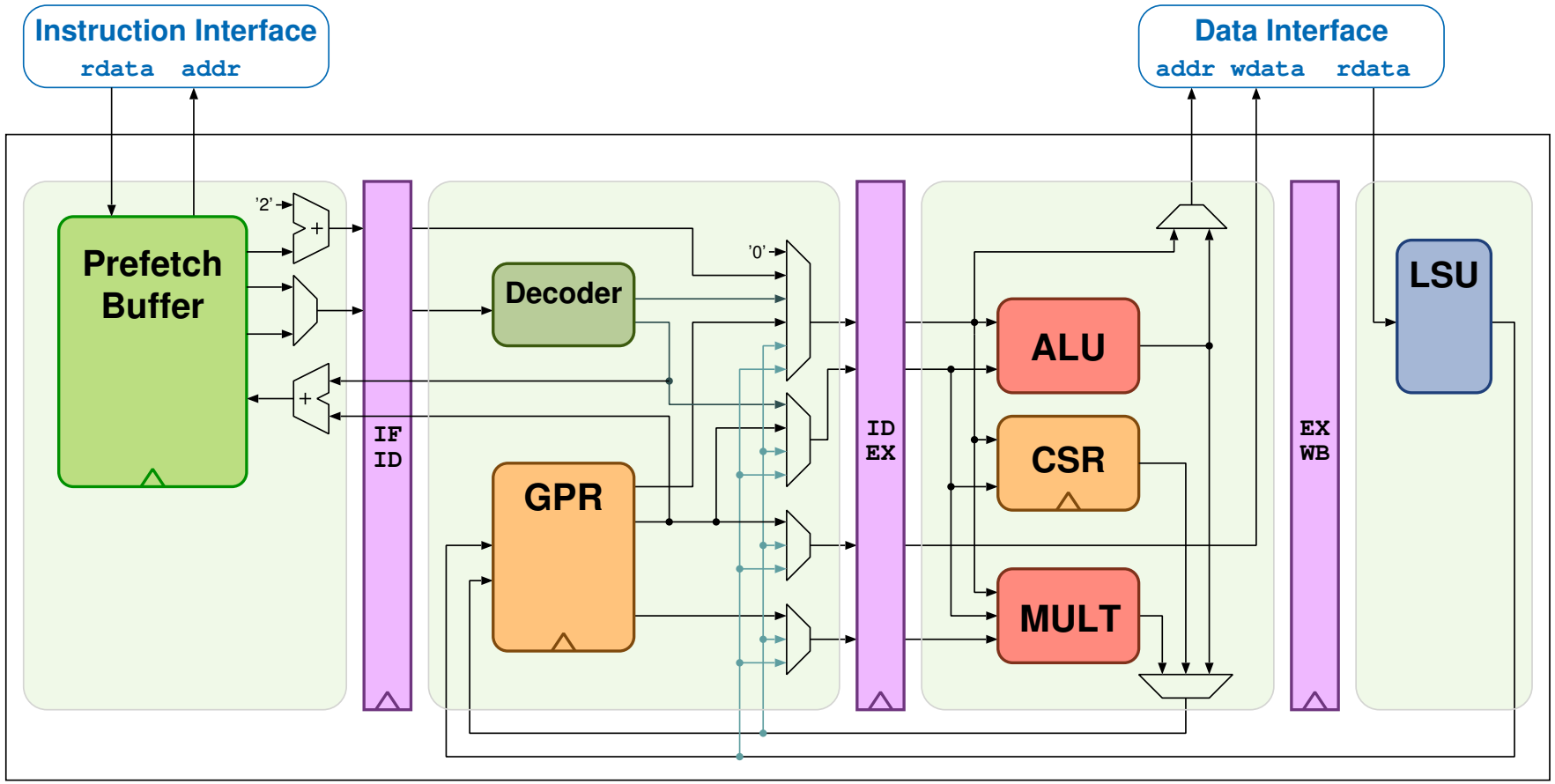
RI5CY: RISC-V Instruction Set Support

- Base 32-bit integer instruction set (RV32I)
  ALU operations, loads, stores, branches, jumps, ...

- Multiplications supported: Subset of "M" extension
  - Currently no divisions and no multiplications with upper 32 bit result
  - We are looking into multi-cycle divisions and micro-coded multiplications

- Compressed instruction set extension (RVC)
  - Smaller code size, less pressure on instruction cache
  - Requires support for 16-bit aligned memory access

RI5CY: 4-stage in-order pipeline



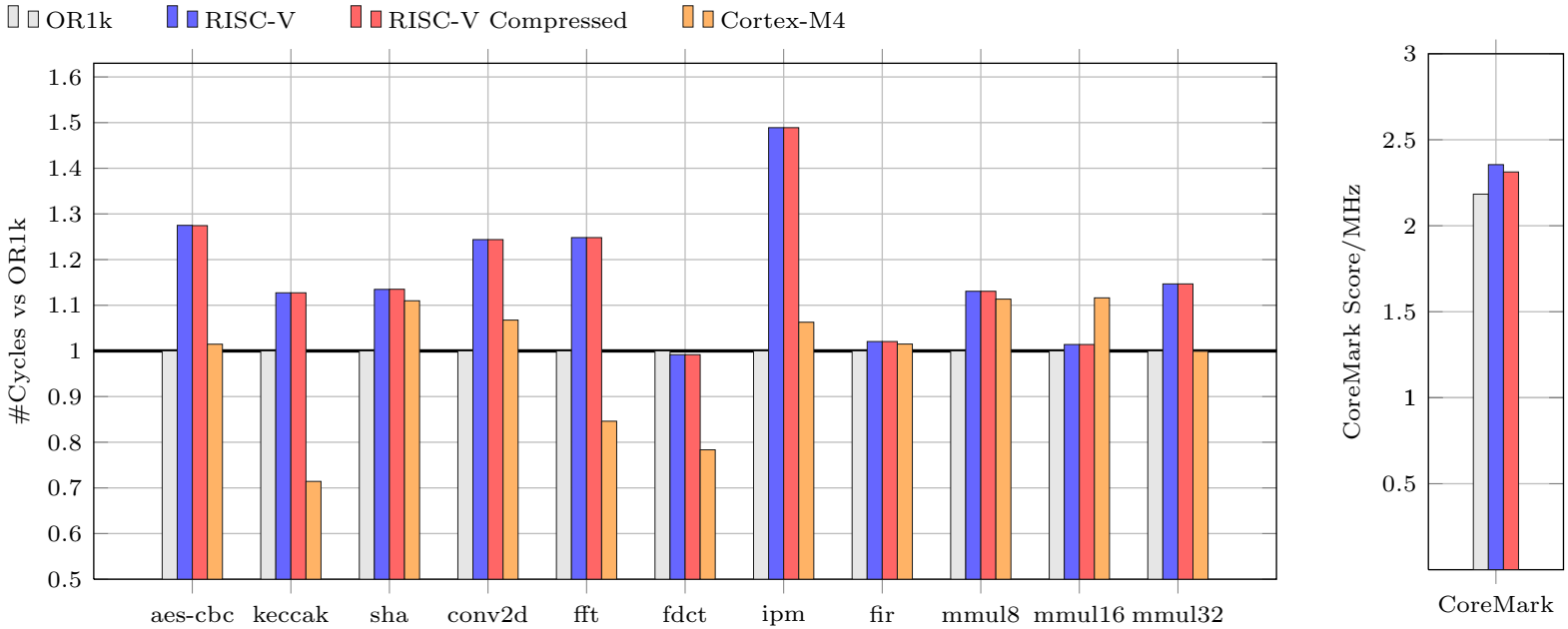
- Single cycle multiplications
- Critical path: Balanced between instruction and data access

RI5CY

Custom ISA extensions

- Register-register load and stores
  If offset is not available at compile time
- Post-increment load and store instructions
  Increase base register by offset after LD/ST done
- Hardware loops
  Hardware takes care of loop counter and branches automatically at end of loop
- Multiply-Accumulate ($d = a \times b + c$)
  MAC with three input operands

RI5CY: Speed



- Compilers used
  - ARM: Official GCC ARM Embedded toolchain (gcc 4.9)
  - OR1k: Custom GCC toolchain (gcc 4.9)
  - RISC-V: GNU toolchain from Berkeley (gcc 5.2)

RI5CY: Area

Component	Area [kGE]
Core	30.5
⊢ Register File	11.1
⊢ Decoder	1.0
⊢ Compressed Decoder	0.7
⊢ EX Stage	6.4
⊢ Multiplier	3.3

- Latch based register file
  Consumes only 35% of area

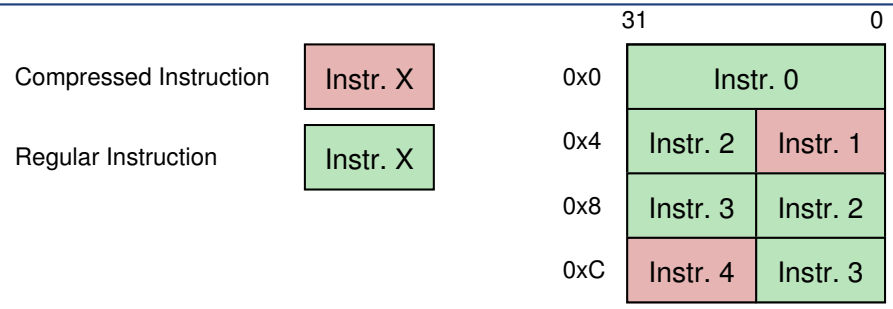
RI5CY

Branching is difficult

- No delay slot in RISC-V
- Jumps loose one cycle
  Next instruction after jump was already fetched
- Combined branches: No setflag instructions
  Branch decision computed with branching instruction
- Branch decision must be evaluated in EX stage
  Otherwise increases critical path
- Taken branches loose two cycles
  Since branch decision is available only late in the pipeline
- Not taken branches don't loose cycles

Compressed instructions

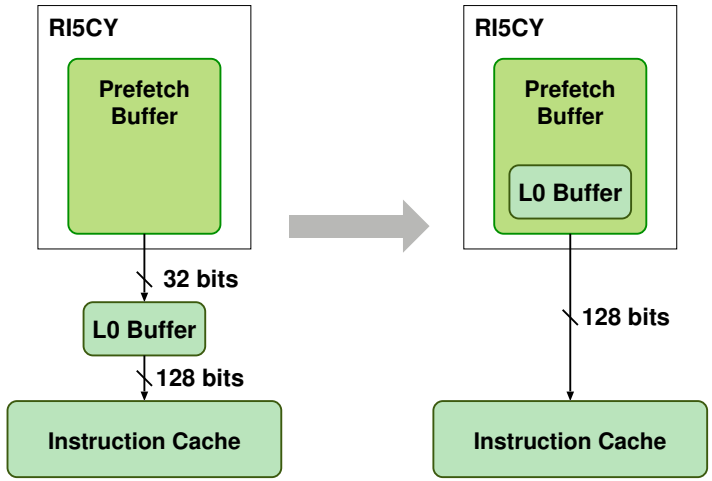
The need for prefetching



- Instruction memory is word-aligned
  Cannot send unaligned requests to it
- Cross-word instruction needs to be assembled from two words
  Need to store upper half word of last fetch
- If lower half word is compressed instruction, no need to fetch next word already
  But cannot wait for memory response to arrive before sending next request
- Solution: Prefetch buffer with 3 entries
  2 are not enough for 2 cross-word instructions back to back

Prefetching with an instruction cache

Merging with L0 Buffer



- L0 Buffer holds most recent cacheline
  Reduces contention on instruction cache access
- With the use of a prefetch buffer we cache most instructions twice
  Once in the L0 buffer and once in the prefetch buffer
- Merging the two removes this redundancy and improves performance

Upcoming RI5CY tapeouts

- Tapeout of a PULP cluster in Global Foundries 28nm
  November 2015

- Tapeout of the PULPino system in UMC 65nm
  January 2016

- More to come

Open Source Release

- What we release
  - PULPino platform
    Including FPGA build flow
  - RI5CY core
  - PULPino IPs: AXI interconnect, peripherals, ...
  - JTAG debug interface
  - Supporting software and tools
  - FreeRTOS port
- All IPs are silicon-proven in multiple technologies
- License will be very liberal
  We will align with LowRISC
- Release very soon
  Before end of 2015
- Check our website: pulp.ethz.ch

Open Source Release

Where we could use your help

- Compiler support
  Integrate ISA extensions & improve performance
- Applications of the PULPino
- Peripherals and software drivers
- Porting embedded operating systems to PULPino
- Boards for a PULPino ASIC
- Porting PULPino to other FPGA platforms

Summary

- First step in open sourcing PULP
- The PULPino platform: Micro-controller like system
- RI5CY: Small and optimized RISC-V core
- First tapeout of PULPino and RI5CY before end of 2015

Outlook

- Floating point support
  Already implemented in OpenRISC PULP core
- Improve branching speed with branch prediction
  Predict branch decision in ID stage
- Look into branch target buffer
  Could reduce branching delay to zero
- Evaluate ISA extensions for RI5CY
  cmov, fractionals, packed-SIMD, bit manipulation, ...


Questions?