The PULP Cores: A Set of Open-Source Ultra-Low-Power RISC-V Cores for Internet-of-Things Applications

Pasquale Davide Schiavone, Florian Zaruba

Davide Rossi, Igor Loi, Antonio Pullini, Francesco Conti, Florian Gasler,
Stefan Mach, Giovanni Rovere, Germain Haugou, Manuele Rusci,
Alessandro Capotondi, Giuseppe Tagliavini, Daniele Palossi, Andrea
Marongiu, Fabio Montagna, Victor Javier Kartsch Morinigo, Simone
Benatti, Lei Li, Eric Flamand, Frank K. Gürkaynak, Luca Benini

1Department of Electrical, Electronic and Information Engineering
2Integrated Systems Laboratory
Energy Efficient HW

Sense

MEMS IMU
MEMS Microphone
ULP Imager
EMG/ECG/EIT

100 μW ÷ 2 mW

Analyze and Classify

μController
IOs

1 ÷ 10 mW

Transmit

Short range, medium BW
Low rate (periodic) data
SW update, commands
Long range, low BW

Battery + Harvesting powered → a few mW power envelope

Idle: ~1μW
Active: ~ 50mW

Davide Schiavone | 07.12.17 | 2
# RISC-V cores under development

## 32 bit

<table>
<thead>
<tr>
<th>Low Cost Core</th>
<th>Core with DSP enhancements</th>
<th>Floating-point capable Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero-riscy</td>
<td>RI5CY</td>
<td>RI5CY + FPU</td>
</tr>
<tr>
<td>RV32IMC</td>
<td>RV32IMCX</td>
<td>RV32IMFCX</td>
</tr>
<tr>
<td>Micro-riscy</td>
<td>SIMD</td>
<td></td>
</tr>
<tr>
<td>RV32EC</td>
<td>HW loops</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit manipulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fixed point</td>
<td></td>
</tr>
</tbody>
</table>

## 64 bit

<table>
<thead>
<tr>
<th>Linux capable Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ariane</td>
</tr>
<tr>
<td>RV64IMC</td>
</tr>
</tbody>
</table>
RISCY Core for Energy Efficient Data-Computation

- 4-stage pipeline
  - RV32IM[F]C
  - 40.7 – 69.3 kGE
  - 30 logic levels of critical path
  - Coremark/MHz 3.19
- Includes various extensions
  - pSIMD
  - Fixed point
  - Bit manipulations
  - HW loops
- Working Chips in:
  - UMC65
  - GF28
- Versions taped out in:
  - TSMC40LP
- Tape out soon:
  - GF22 FDX

Different Options:
- **FPU**: IEEE 754 single precision
  - Iterative DIV/SQRT (7 cycles)
  - Pipeline MAC, MUL, ADD, SUB, Cast
  - Single cycle load, store, min, max, cmp etc
- **Privilege support**:
  - Support subset of privilege mode M and U

https://github.com/pulp-platform/riscv
RISCY – ISA Extensions improve performance

### Baseline

```assembly
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
```

11 cycles/output

### Auto-incr load/store

```assembly
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
```

8 cycles/output

### HW Loop

```assembly
lp.setupi 100, Lend
   lb  x2, 0(x10!)
   lb  x3, 0(x11!)
   add x2, x3, x2
Lend:  sb  x2, 0(x12!)
```

5 cycles/output

### Packed-SIMD

```assembly
lp.setupi 25, Lend
   lw  x2, 0(x10!)
   lw  x3, 0(x11!)
   pv.add.b x2, x3, x2
Lend:  sw  x2, 0(x12!)
```

1,25 cycles/output

Small Power and Area overhead → Energy reduction in NT >3x
Zero-riscy Core for Energy Efficient Control-Tasks

- 2-stage pipeline
  - RV32{I,E}[M]C
  - 11.6 – 18.9 kGE
  - 30 logic levels of critical path
  - Coremark/MHz
    - RV32IMC 2.44
    - RV32EC 0.91
  - Optimized for area
    - High resource sharing
    - 3/4 cycles mul
    - 1bit iterative divider
- Versions taped out in:
  - UMC65
  - TSMC40LP

- Different Options:
  - Zero-riscy:
    - 32 GP registers, HW multiplier and division
  - Micro-riscy:
    - 16 GP registers, no HW resources for multiplications and divisions

https://github.com/pulp-platform/zero-riscy
Ariane Core for Linux

- 6-stage pipeline
  - RV64IMC
  - 185 kGE
  - OoO execution
  - In-order Commit
  - M, S and U privilege modes
  - TLB
  - Tightly integrated D$ and I$
  - Hardware PTW
  - Branch Prediction
  - Scoreboarding
  - Coremark/MHz 2.01

- Optimized for speed
- Synthesized for 1.5 GHz@0.72V, SSG corner in GF22 FDX

SOON OPEN-SOURCE

Davide Schiavone | 07.12.17 | 7
Rich SW tools

Virtual platform implementation:
• C++ + Python
• Any ISS can be integrated (or1ksim, gdb simulator, riscv on-going)

Timing model:
• Fully-event based
• Includes timing models for interconnects, DMACs, memories…

Simulation performance:
• Around 1MIPS simulation speed
• Functionally aligned with HW
• Timing accuracy is within 10-20% of target HW

Debug with GDB:
• Supports RTL and virtual platform
• Uses a bridge to inject JTAG requests

Profiling with KCachegrind:
• Supports PC traces from RTL and VP
• Several events can be caught (PC, cycles, I$ miss, stalls…)
Recent effort in Core Verification

- Constrained Pseudo-Random Test in a perturbated environment (random interrupts, stalls)
- The uGP program generator tries to maximize the code coverage
- IIS and the RTL model receive the same random program and have to produce the same output
PULP/PULPino Users

+ Several Universities
PULP Platform: new controller

- New Int-Controller
- uDMA for efficient peripheral data transfer
- RISCY or Zero-Riscy
- New SDK
- Tape out soon: GF22