OpenRISC

... and related projects

University of Amsterdam
Raphael ‘kena’ Poss – Oct 9th 2013

Background: OpenCores

- **OpenCores = “open source hardware community”**
  - But also the name of the web site and component library
- Since: 1995-1999
- What: open source “IP blocks” in Verilog/VHDL
  - processors, memory controllers, I/O interfaces, buses, accelerators
- Who: 150.000 registered users (est. 2012)
- Why: foster reuse and standardization

Roots of OpenRISC

- Flagship project of the OpenCores community
- **Defined by interface** in the **OpenRISC spec:**
  - 32/64-bit ISA: inspired by Alpha, optional delay slots
  - “layers” of features, first is mandatory but simple
  - Also a system architecture: specs for ABI, exception handling, virtual memory management etc.
- Multiple implementations **possible and encouraged**

Target use cases

- Research/industry:
  - Embedded control systems
  - Fast FPGA prototyping of utility SoCs
  - Fast prototyping of ISA extensions
- Example applications:
  - Samsung for the DTV SoC
  - Allwinner for power controllers
  - NASA for the control system of TechEdStad (ISS)

Target use cases (cont.)

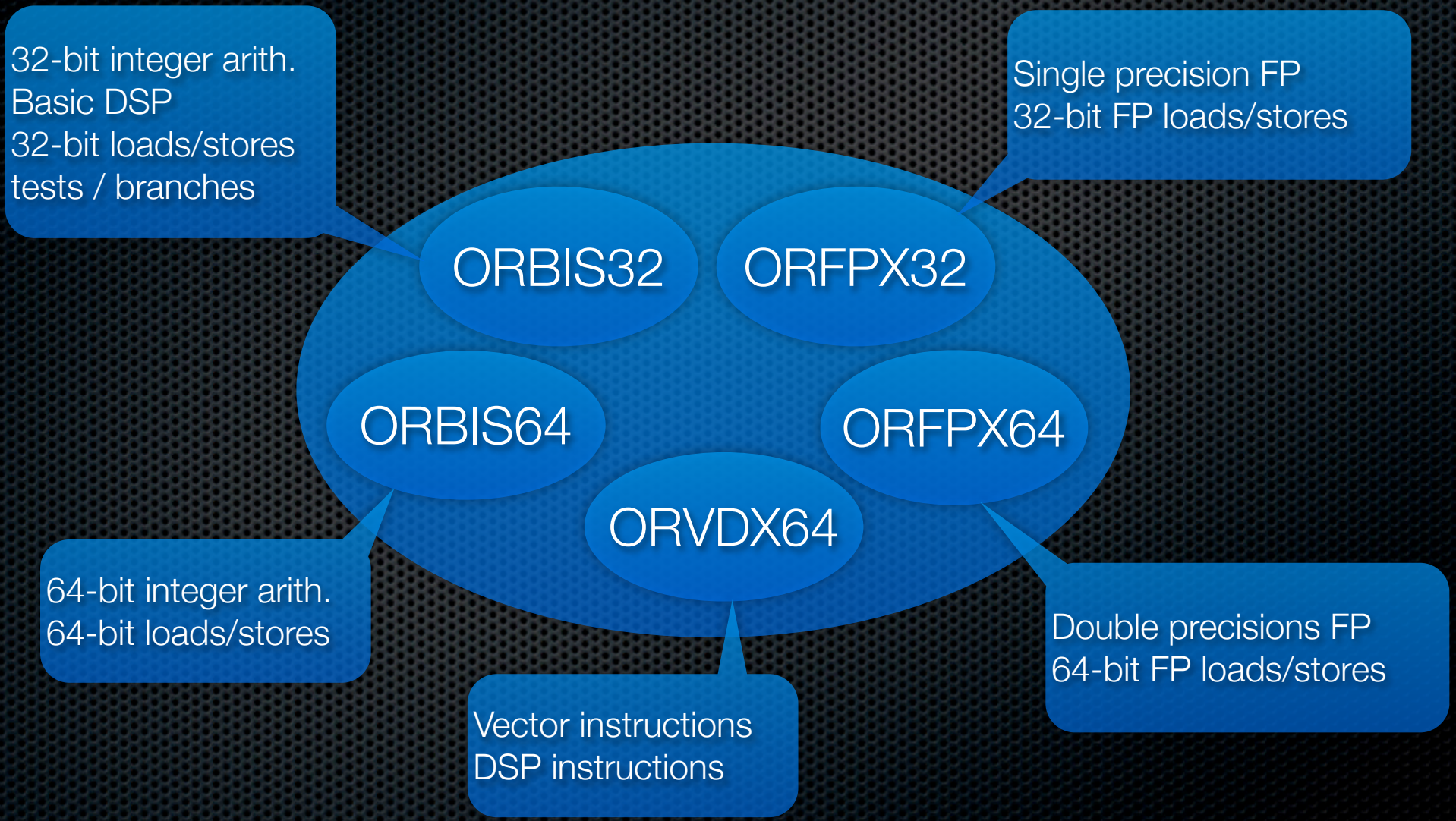
- Research/education:
  - Assembly programming
  - Study & implementation of operating systems
  - Compilers and code generation
  - Simulation
  - Fast SoC prototyping and FPGA programming

OpenRISC 1000 (Architecture specification)

Overview

- First edition 2000, latest 2012, multiple contributors
- **Regularly updated** to reflect the state of latest implementations; edits by consensus in community
- Specifies: addressing modes, register set, instruction set, exception model, memory model, virtual memory management, cache model & coherency, debugging, performance counters, power management, interrupt controller, programmable timer, ABI
- Overall a **showcase of an open, modular standard spec** geared towards real hardware implementations

OR1K ISA Sub-sets



OR1K Software support

- GNU binutils: assembler, linker - since 2000
- Custom GCC since ~2000, up-to-date with v4.9
- Preliminary support in LLVM
- uClibc / newlib (embedded C libraries)
- Standard in Linux 3.1, RTEMS, FreeRTOS, eCos

OR1K Emulators

- “Official” emulator: **or1ksim**, single core
  - Stable has ORBIS32, dev ORBIS64/ORFPX64
- jor1k (Javascript) http://s-macke.github.io/jor1k/
- QEMU
- Simulators: SysC via **Verilator**, soon MGSim!
  - (side note: Verilator = Verilog to SysC translator for cycle-accurate simulation, also free & open source)

OR1K instruction classes

- Class I : mandatory instructions
  - ORBIS32/64: arith, logic, ldX/stX, **sfX**, jal, **sys**, etc.
  - ORFPX32/64: add/sub, ftoi/itof, **mul**, sfX, etc.
  - ORVDX64: arith/logic, all, avg, cmp, pack, etc.
- Class II: optional instructions
  - ORBIS32/64: **mul**, mac/s, **div**, ror, rol, trap, psync, sfi
  - ORFPX32/64: **div**, **rem**, madd

OR1K exception model

- Synchronous (internal) / asynchronous (external)
- Async maskable (irq, timer) / unmaskable (buserr, reset)
- PC saved to EPCR, new PC set to addr [256 x ENUM]
  - Reset has ENUM 1 so boot code always at 0x100
- Exception mode disables virtual address translation
- Optional fast context switching with reserved registers

OR1K fast context switching

- **Optional feature, but fast is mandatory by spec**
  - special register reports to OS if feature available or not
- Special control register **“CID” = context ID, 4 bits** (16 contexts)
- CID is a pointer in the (bigger) register file: each context has its own set of registers; read stage offsets by CID
- CID changed on exceptions or by supervisor code
- CID is automatically prefixed to memory addresses
  - program sees 32(or 64) bits, cache/MMU sees 36(or 68)

OR1K MMU

- 32-bit or 64-bit “effective” addresses (used by program)
- 36-bit or 68-bit virtual addresses (input to MMU)
- Software-managed TLB, hardware optional (2 levels)
- Page sizes: 8KiB, 16MiB, 32GiB
- Page flags:
  - usual suspects: r/w/ux/sx, dirty, accessed, mapped
  - **cache policy**: uncacheable, write-back/write-through
  - **weakly ordered** (if not: fetch/LSU must run lock-step)
  - **coherent** (if not: MP caches do not need to synchronize)

OR1K memory model

- Super simple!
  - Memory accesses must be aligned on their size
  - Big endian
  - Assumes reordering can happen, **msync** to barrier
  - weakly ordered and coherent flags in MMU
  - cache control: flush/invalidate, optional prefetch/lock

OR1K Debug & perf

- Debug unit:
  - 8 watchpoints (trigger exceptions), 8 trace masks
  - Can reroute exceptions to external debugger
- Performance counters:
  - 32-bit width
  - Separate user / supervisor counters
  - Counters for: loads/stores/fetches, cache misses, TLB misses, load/store/fetch/branch/dependency stalls, watchpoints

OR1K Power management

- 16 frequency levels (logarithmic reduction via divider)
- **Doze**: wait on timer or external interrupts
- **Sleep**: same as doze, all internal clocks gated
- **Suspend**: halts the core; reset signal to resume
- “Dynamic clock gating” - hardware decides per cycle

OpenRISC implementations

OpenRISC 1200

- Implemented in Verilog, license LGPL
- 5-stage, single-issue, in-order pipeline, with delay slot
- ORBIS32 and optional ORFPX32
- 64-entry TLBs (configurable 16-256), Wishbone interface
- Example ASIC perf: 250MHz at 0.18µm tech, 1W max power envelope
- FPGA:
  - 7K core cells (1100 FFs, 4 block RAMs) at 35MHz on Actel ProASIC3
  - 2.4K LUTs, 1 block RAM at 125MHz on Xilinx Virtex 5
- **Mature but monolithic and relatively hard to maintain**

AltOr32

- “Alternative Lightweight OpenRISC”
- ORBIS32 only, no MMU, no delay slot
- Tailored for low-end FPGAs - minimizes area cost
- Simple and well-structured Verilog, LGPL licensed
- Also used to test / showcase Verilator

mor1kx

- Recent project
- Modular sourcebase, with 3 “main” configurations:
  - (Pronto) Espresso: 3-stage pipeline, no MMU
  - Cappucino: 6-stage pipeline, MMU, branch predictor
- Wishbone interface, supports debug interface
- Cappucino runs Linux
- Can fit ~4 Cappucinos on my DE0_nano Altera Cyclone 4 clocked at 50MHz

ORPSoC

- ORPSoC: “OpenRISC platform system-on-chip”
- Intended to serve as **reference platform** for OS development - towards standard software distributions
- Also an open source, modular project
- Features OR1K core(s), SDRAM interface, SD flash, JTAG debug, UART, VGA adapter, mostly tailored towards Wishbone
- Supports extending with other FPGA IP blocks

OpTiMSoC

- Multi-core, tiled OpenRISC-based research platform
  - With ISA extensions for direct point-to-point communication
- Tailored towards customizability (for research)
- Simulation & dev env available as VirtualBox image
- Runs on (big) FPGA boards
- PhD project of Stefan Wallentowitz & 5 others
  - Teschnische Universität Munchen
  - Official spec http://arxiv.org/abs/1304.5081

Between the lines

Known unknowns and unknown unknowns

- Known unknowns:
  - No implementation yet of ORBIS64, ORFPX64 and ORVDX64 - how do they behave in practice?
- Unknown unknowns:
  - Who exactly is using OpenRISC in industry?
    What challenges are they facing?
    Few dare to admit they use OpenRISC because of the open source stigma

Ongoing/future work

- mor1kx is intended to slowly replace OpenRISC 1200 as a “reference” implementation
- Another architecture spec “**OpenRISC 2000**” is in the works:
  - tailored to “deep embedded”
  - 16/32-bit only, no 64-bit
  - Mixed 16/32 instruction set (like ARM Thumb)
  - Preliminary discussions only, no implementation yet

Take-away

What you can do with it

- Open-source, open and customizable **replacement for MicroBlaze** / LEON3 in your SoCs / simulations
- Open and modular Verilog **eases customization** and **lowers barrier to entry** in architecture research
- Education: **clean, orthogonal and extensible ISA**
  - No register window crazyness, exotic addressing modes, “oh wait we forgot this” x86-style and ARM-style ISA extensions
  - Delay slots optional (in contrast to MIPS)
  - With current hardware platforms (in contrast to Alpha)

Questions?