lowRISC - an open-source RISC-V SoC

Alex Bradbury, lowRISC/University of Cambridge

asb@lowrisc.org

@asbradbury @lowRISC

RISC-V workshop 2015/01/14

Beginnings and motivation

- Aim to produce volume silicon for a complete open-source SoC
- Started Summer 2014 as a non-profit project
- Previous experience with Raspberry Pi
- Why?
  - Teaching and research
  - Demand from industry
  - Startups and innovation

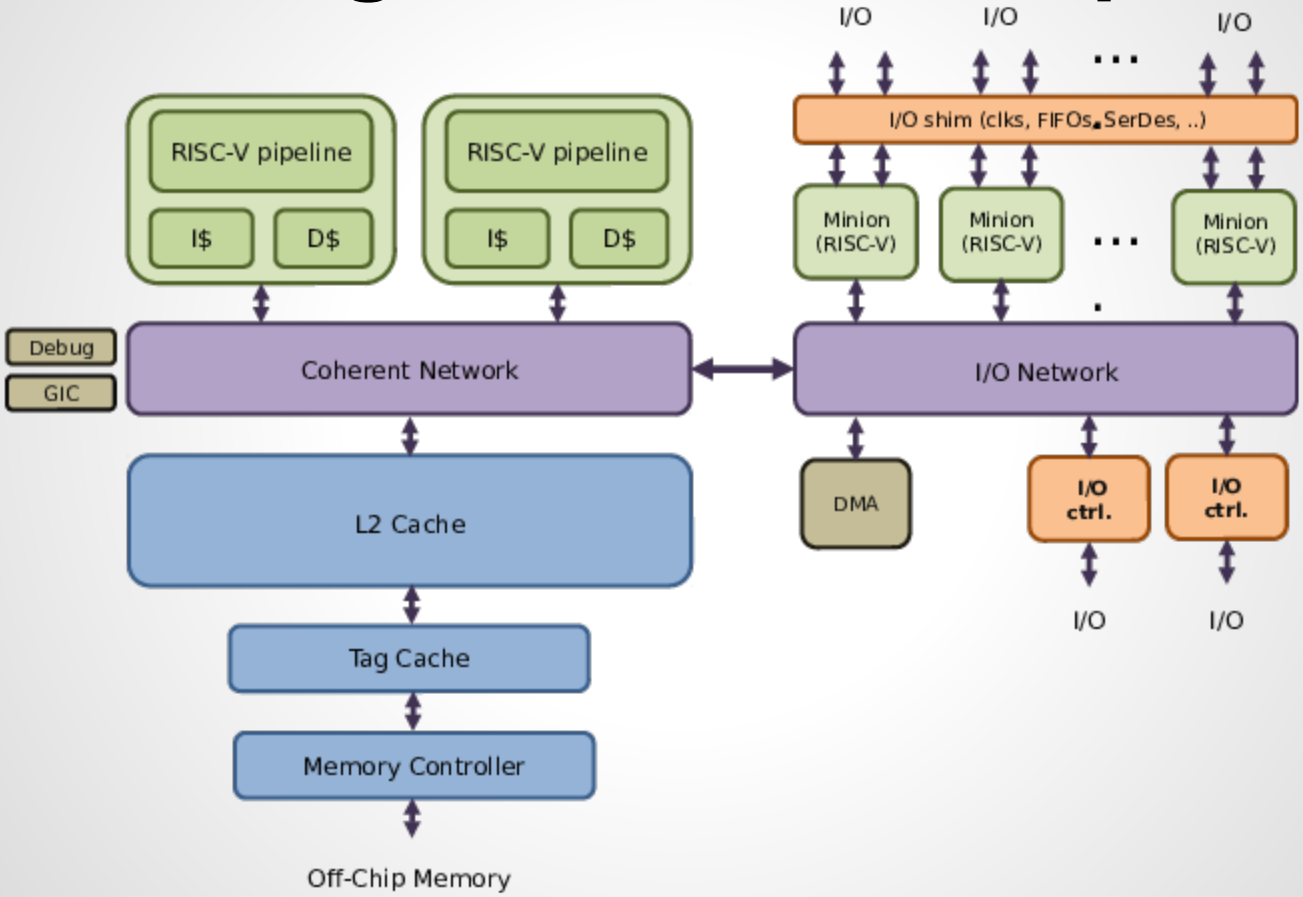
The opportunity

- Clean slate design
- Technology scaling is slowing
- Cores are free and customisable
- Free from commercial influences and release cycles. Aim to maximise functionality (no product range!)
- Open source community

Approach

- Simple reusable components rather than single-purpose solutions to problems
- Derive from Berkeley's Rocket RISC-V core
- Expose interesting new features (particularly security)
- Develop out in the open as much as possible
- Multiple volume silicon runs
- Initial volume target: a low-cost development board. 'Raspberry Pi for grownups'

System diagram for test chip



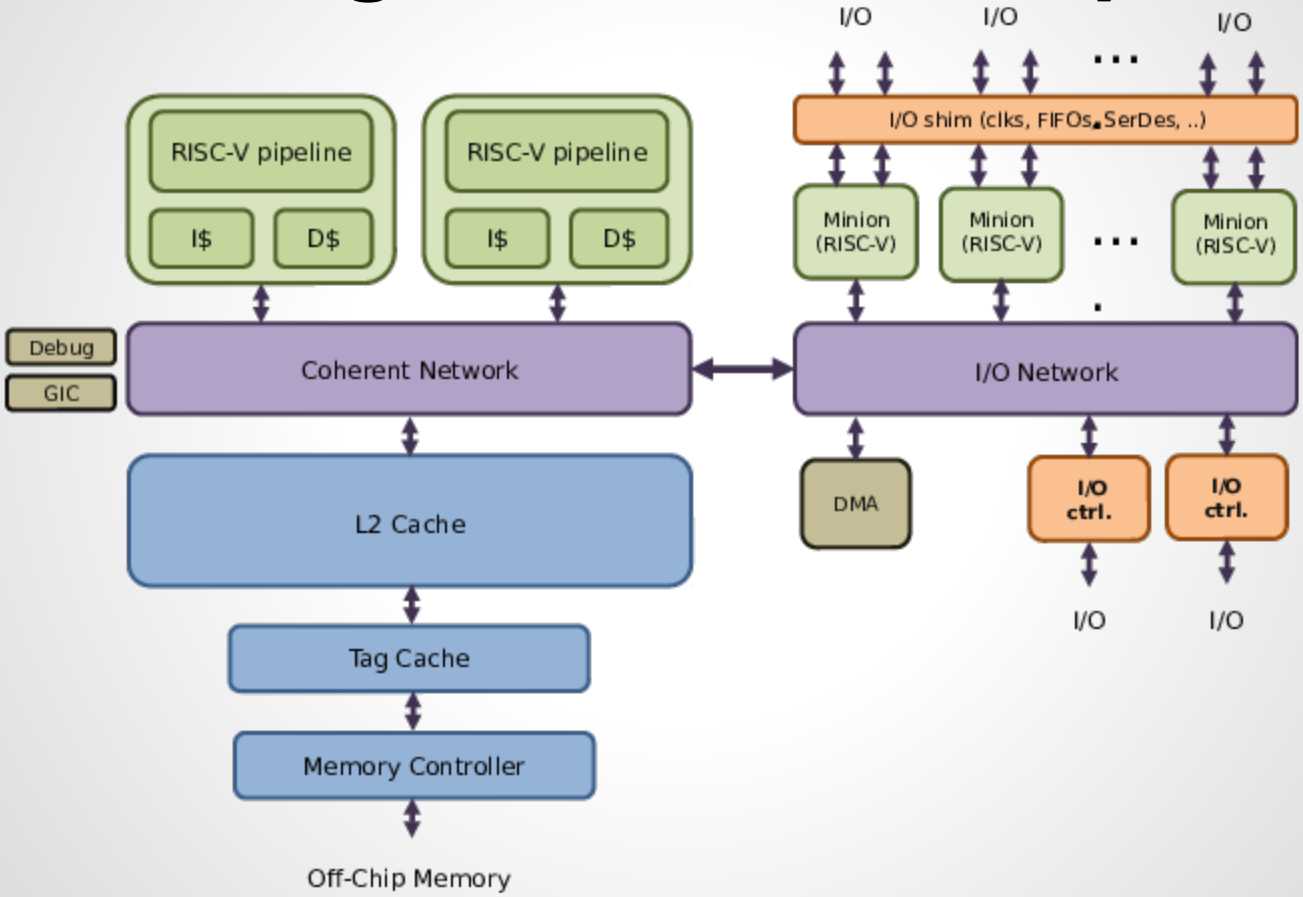
Tagged memory

- Associate tags (metadata) with each memory location
- Initial motivation is prevention of control-flow hijacking attacks (still a major attack surface)
  - Provide protection for code pointers. i.e. set tag bit = read-only
- Low overhead implementation. Tag bits copied to L1/L2 and on-chip tag cache
- Exploring 2-bit tags (~3% storage overhead)

Tagged memory - beyond security

- Infinite memory watchpoints
- Better version of traditional canaries
- Garbage collection
- Accelerate debug/performance tools (e.g. Google *Sanitizer)
- Per-word locks or full/empty bits for synchronisation
- Mark valid targets of indirect branches

System diagram for test chip



Minion cores

- Motivation
  - Soft peripherals
  - I/O preprocessing/filtering, wake-up main cores
  - Offload fine-grain tasks, e.g. security policies, debug, performance monitoring
  - Off-load tasks from main cores
  - Secure, isolated execution
- Not a new idea
  - CDC6600, TI PRUs, Ubicom IP3000, XMOS, NXP LPC4370, Motorola (e)TPU

Minion cores - architecture

- Predictable timing
- I/O ‘shim’
  - Logic to aid shift in/out, parallel load, buffer data, provide clocks, assign pins to minions
- Low-latency path between main cores and minions
  - May carry cache misses, branch mispredicts

Roadmap

- Q1 2015 - Tagged memory (FPGA)
- Q2 2015 - Minion cores (FPGA)
- End 2015 - Dual-core test chip with integrated memory PHY, minions. 28 or 40nm
- First volume run 2016/2017

Next steps

- Further software work
- Documentation
- Verification, formal methods
- Larger scale benchmarks. “Run Linux well”
- Programmable interrupt controller, performance counters, debug...

Cathedral and the bazaar

- Want to be a truly open-source project
- Case 1: source is open, but development happens behind closed doors
- Case 2: development happens in the open, but nobody else chooses to take part
- Case 3: open source + open development with a community of external contributors

Final notes

- See www.lowrisc.org for
  - Announcement list
  - Discussion list
  - Memo with many more details on tagged memory and minion cores
- Keen to receive feedback, explore collaborations etc. asb@lowrisc.org
- lowRISC at FOSDEM 2015 (31st Jan, Brussels)