lowRISC: an open-source RISC-V SoC

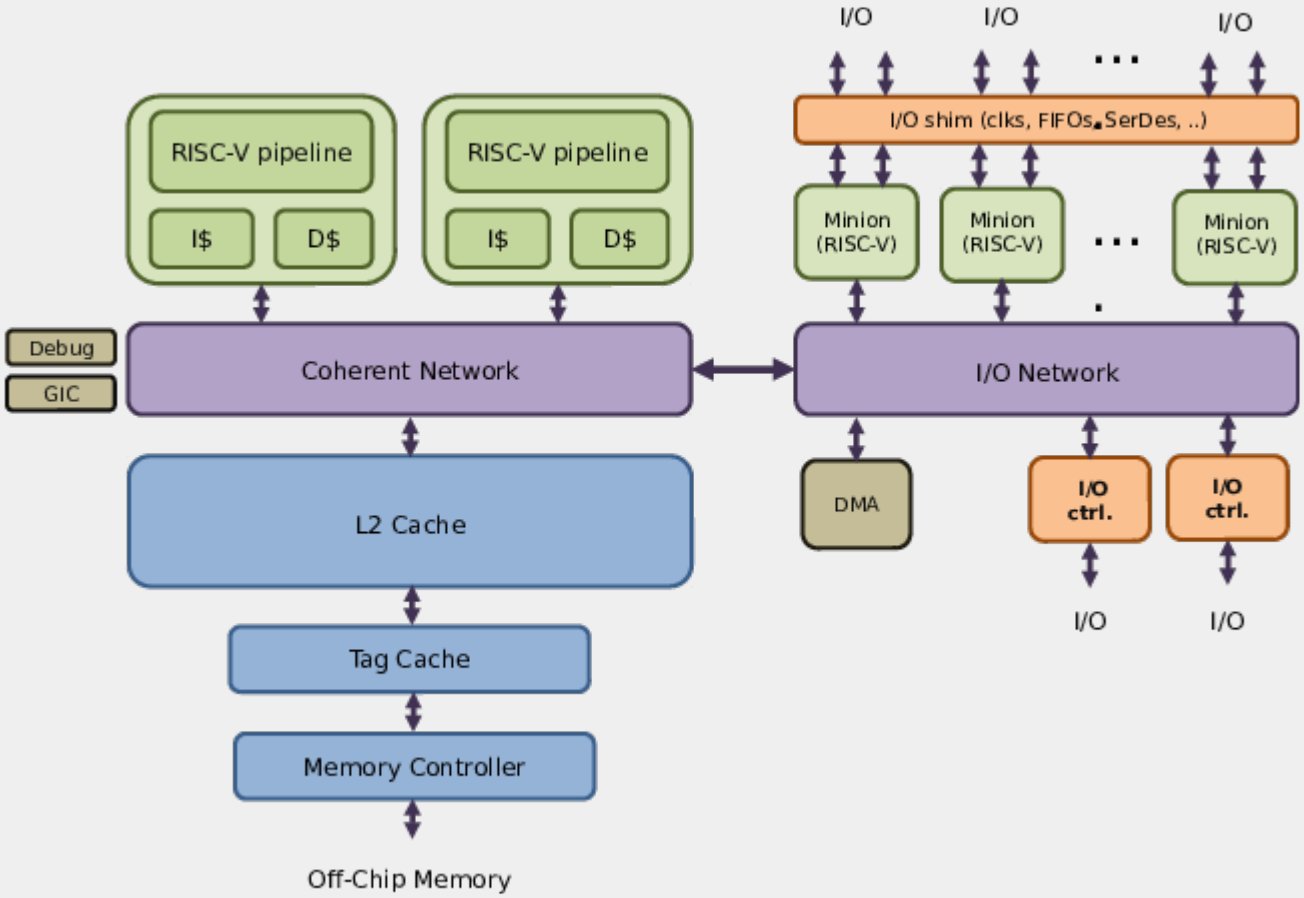
lowRISC

Alex Bradbury
asb@lowrisc.org @asbradbury @lowRISC

2nd RISC-V workshop 29/06/15

lowRISC in brief

- Aim: Bring the benefits we see in open source software to the hardware world
- Produce a complete open-source reference SoC design for others to build upon
- Linux of the hardware world
- Produce volume silicon and low-cost development boards (Raspberry Pi for grown-ups!)
- Not-for-profit. Based at University of Cambridge Computer Lab.



System diagram for test chip

Tagged memory

- Associate tags (metadata) with each memory location
- Aim to prevent control-flow hijacking
- Many other potential uses (GC, memory watchpoints, full/empty bits for synchronisation, …)
- Download code+tutorial at lowrisc.org
- **See Wei's talk tomorrow for much more detail**

Minion cores

- Motivation
  - Soft peripherals, I/O preprocessing/filtering
  - Offload fine-grain tasks e.g. security policies, debug, performance monitoring
  - Secure, isolated execution
  - Virtualized devices
- Everything old is new again
  - CDC6600, TI PRUs, XMOS, NXP, ...
- Long-term vision: minions distributed throughout the SoC

Core roadmap

- Q3 2015: ‘Untethered’ SoC design (FPGA)
- Q3 2015 - Minion cores (FPGA)
- Q1 2016 - Dual-core test chip with integrated memory PHY, minions, 28nm or 40nm
- First volume run 2016/2017

Current focus: add necessary peripherals to boot Linux on a standalone FPGA platform (no HTIF, no ARM)

Google Summer of Code

- Porting seL4 to RISC-V
- Porting jor1k to RISC-V
- An online Verilog IDE based on Yosys
- TCP offload to minion cores using rump kernels
- Extending the Tavor fuzzer to support directed generation of assembly test cases
- Implementing a Wishbone to TileLink bridge

Other developments

- Recently completed Master’s dissertations (uses of tagged memory, ‘shim’ for minion cores)
- New PhD students and staff starting in October
- Local interns at the lab
- Discussions and collaborations with startups, established companies, and academic groups

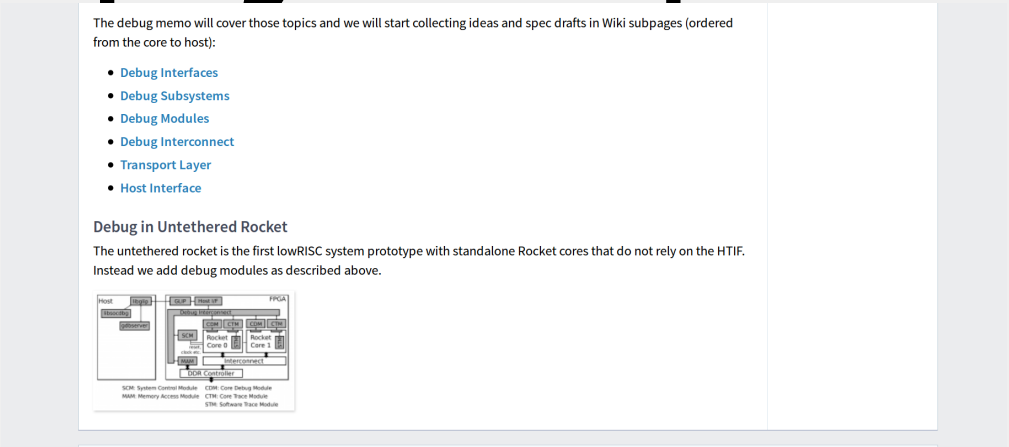
Security features

- Tagged memory, but also hopes for…
- Secure boot
- Crypto accelerators
- Encrypted off-chip memory
- Isolated execution
- Virtualisation
- …

Provide well documented, auditable (open source!) primitives to empower users to protect their privacy.

An open SoC security specification?

Developing in the open: debug



The debug memo will cover those topics and we will start collecting ideas and spec drafts in Wiki subpages (ordered from the core to host):

- Debug Interfaces
- Debug Subsystems
- Debug Modules
- Debug Interconnect
- Transport Layer
- Host Interface

Debug in Untethered Rocket

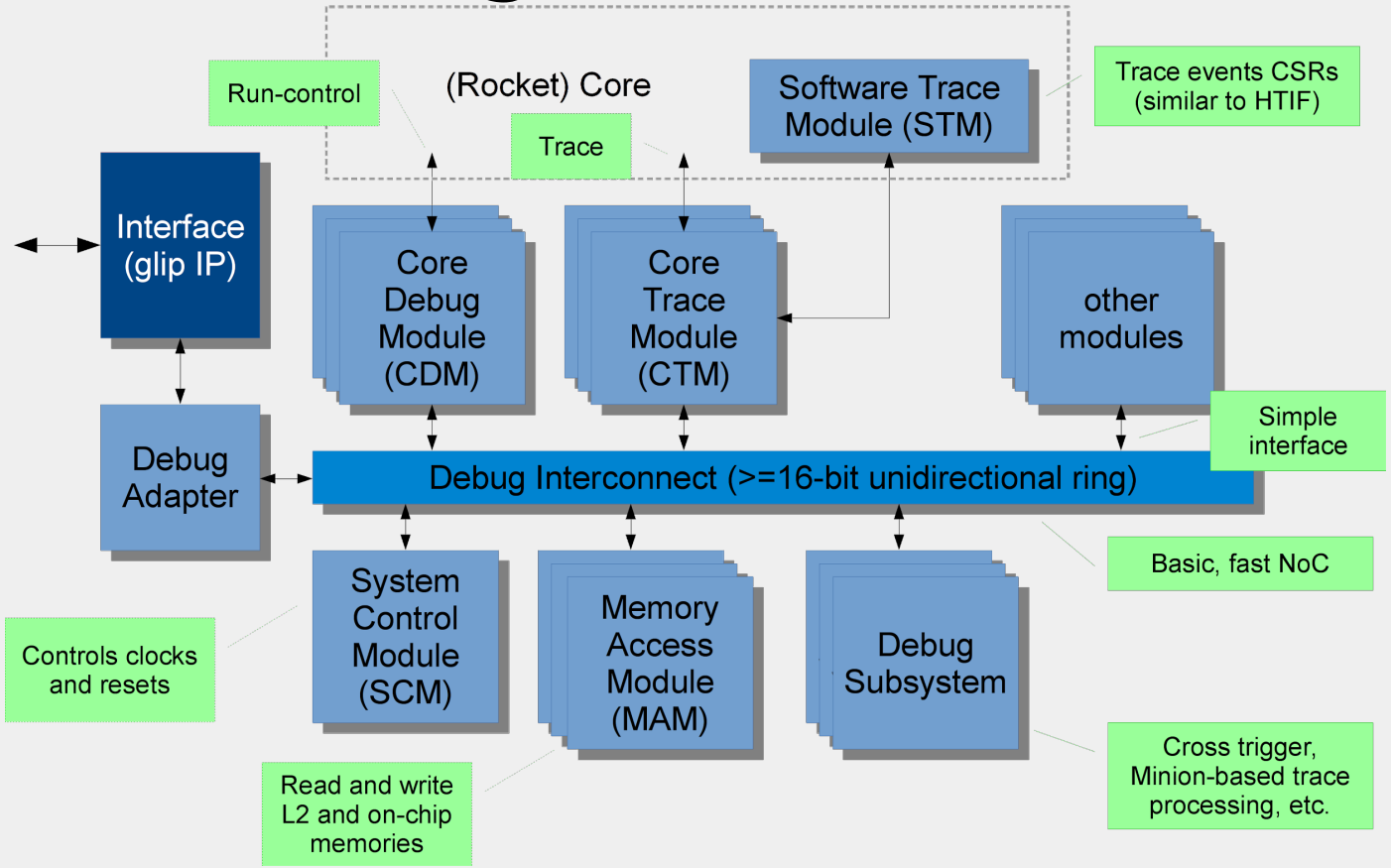
The untethered rocket is the first lowRISC system prototype with standalone Rocket cores that do not rely on the HTIF. Instead we add debug modules as described above.

- Goal: debug memo end of July
- Follow/contribute at phab.lowrisc.org
- Adopt prior trace debug work from OpTiMSoC
- Consolidate in new Open SoC debug project (http://opensocdebug.github.io)
- Major thanks to Stefan Wallentowitz+co-conspirators

lowRISC debug modes

- Run-control debug
  - Standard gdb-like debug mode
  - Step through code, set breakpoints etc
- Trace debugging
  - System generates trace events during execution
  - Filtering, triggering, and processing on chip
- Shared infrastructure for both

lowRISC debug infrastructure

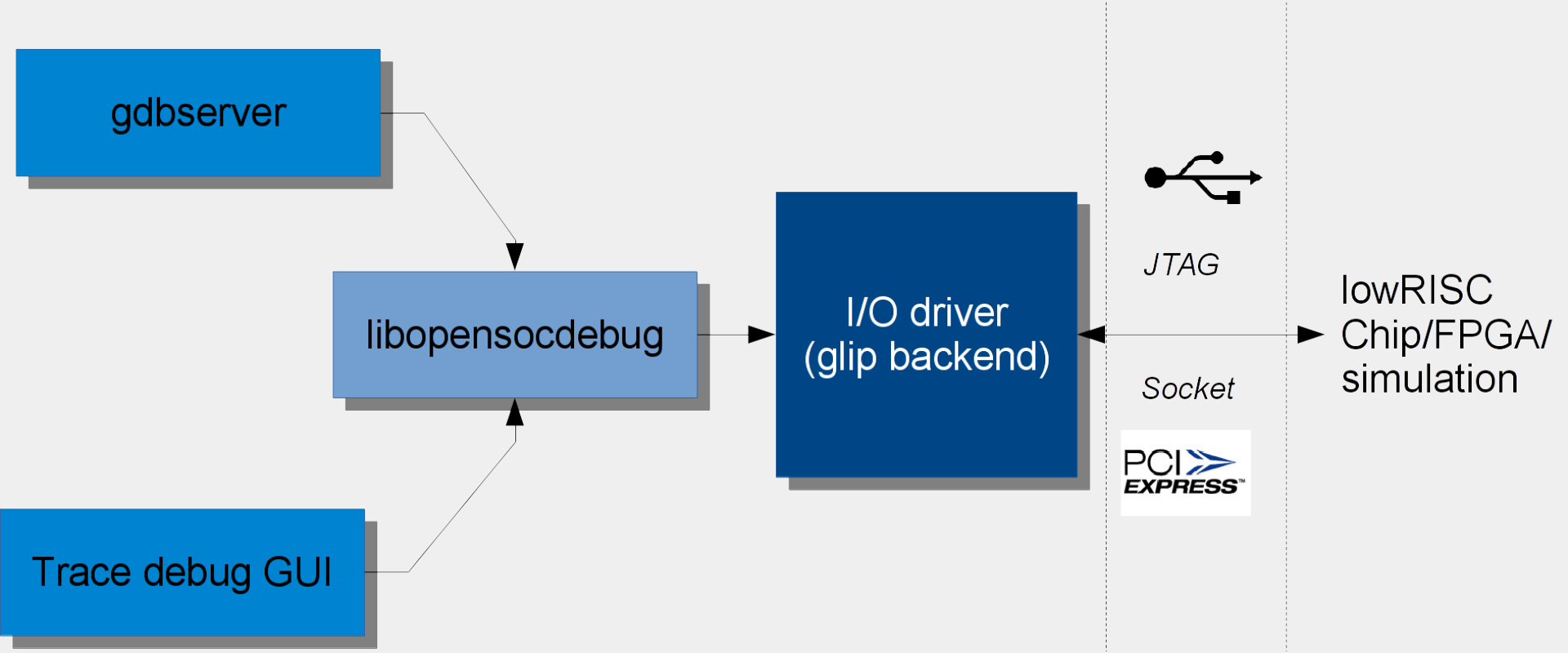


Get involved

- See www.lowrisc.org for
  - Announcement + development discussion lists
  - Memo on tagged memory and minions
  - Blog
  - phab.lowrisc.org bug-tracker and Wiki
- irc.oftc.net #lowRISC
- Keen to collaborate rather than duplicate effort!
- Give feedback, explore collaborations etc: asb@lowrisc.org @asbradbury @lowRISC
- European lowRISC/RISC-V event in October?

Overflow

lowRISC debug host software



Rocket debug interfaces

- Run-control
  - Not specified yet. Any existing work to build on?
- Trace
  - Instruction traces, register file accesses
  - (optional) branch predictor hits, cache hits, etc.
- Software trace
  - Add extra CSRs with software trace events
  - Minimally intrusive software instrumentation