Datacenter Extension

- Move lots of data
- Thousands of threads
- Cache isolation
- Measurement: focus on elapsed time, not CPU time

Dick Sites, Google

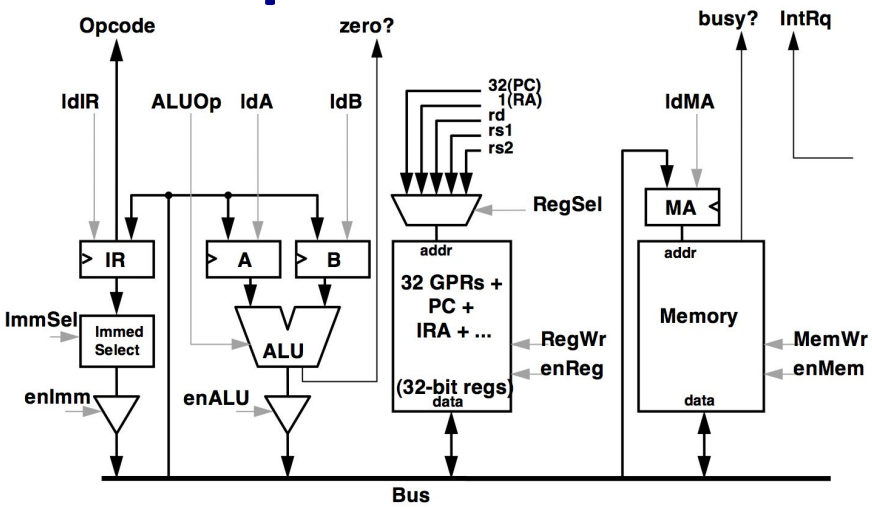
The Sodor Processor Collection: Teaching Computer Architecture, Chisel, and RISC-V with Concrete Examples

- Goals
  - use Sodor in undergraduate-level computer architecture class (CS152)
  - introduce users to Chisel, RISC-V
- Menagerie of RV32I processors
  - 1-stage, 2-stage, 5-stage
    i. (designed to match lecture slides)
  - 3-stage with synchronous scratchpad (RV32IS)
  - micro-coded bus-based design
- Sodor available at:

  github.com/ucb-bar/riscv-sodor/wiki

  riscv.org

  chisel.eecs.berkeley.edu



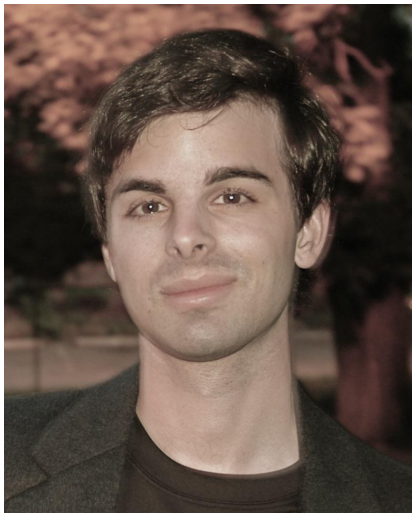
Chris Celio

advised by Krste & Dave

The Berkeley Out-of-Order Machine: Implementing and Comparing High-Performance Processor Designs using Chisel

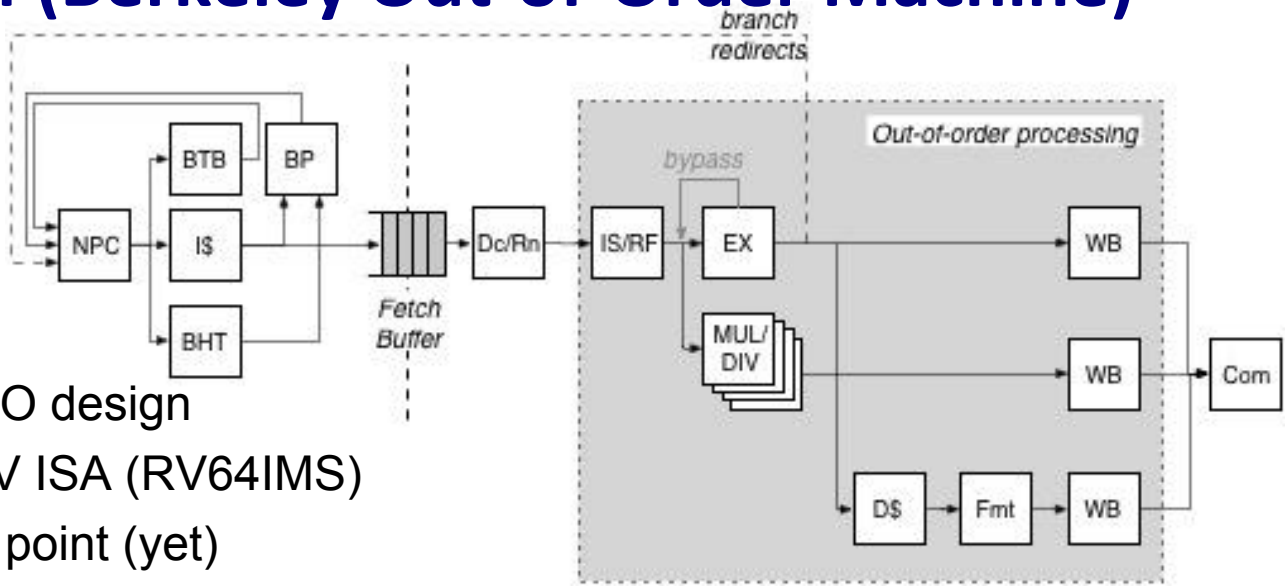
- Goals
  - Create a realistic baseline, strawman RISC-V OoO processor
  - Explore energy-efficiency in OoO designs using real RTL that can be pushed through to layout
  - provide an open-source implementation for both research and education
- Why RTL?
  - grounded in reality
  - Most other research uses software simulators and energy models
- Why Chisel?
  - It's more productive
  - more easy to express generators, parameterized hw

Chris Celio



advised by Krste Asanovic & Dave Patterson

BOOM (Berkeley Out-of-Order Machine)



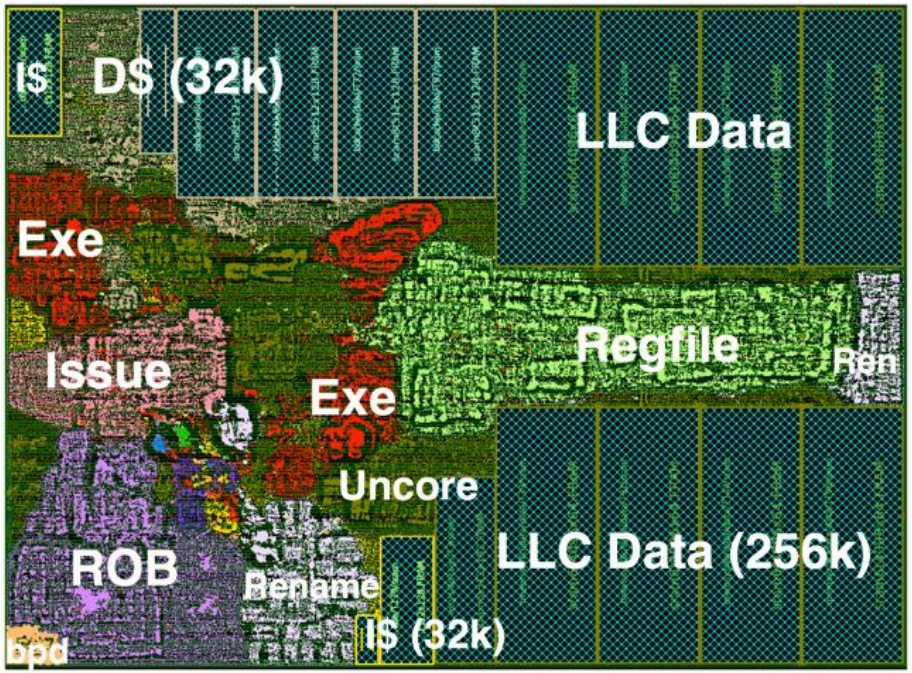
- Baseline, strawman OoO design
  - implements RISC-V ISA (RV64IMS)
    i. no hw floating point (yet)
  - superscalar (parameterizable widths)
  - full branch speculation (BTB/BHT/RAS)
  - load/store queue with store ordering
    i. loads execute fully OoO wrt stores, other loads
    ii. store-data forwarded to loads
- > 2.2 GHz @ 45nm (<30 FO4)
- 8,000 lines of Chisel
- fits into existing Rocket-chip stack
  - re-uses significant pieces of Rocket (uncore, caches, frontend, functional units, HTIF, parameter system, FPGA & VLSI flows, and more!)

Status

- Current
  - BOOM has been used in 3 semesters of Berkeley's undergraduate computer architecture class CS152
  - Runs on zedboard FPGA (>50 MHz)
  - executes dhrystone, Coremark, SPECINT test inputs
    (~10-100 Billion cycles each)

- Next Steps
  - Performance tune
  - Finish debugging BOOM
  - need story for debugging
    100B-1T cycles in
  - Explore design trade-offs
  - ???
  - Paper!

1.7mm^2 @ 45nm



2-wide BOOM layout.

ASPIRE UC Berkeley

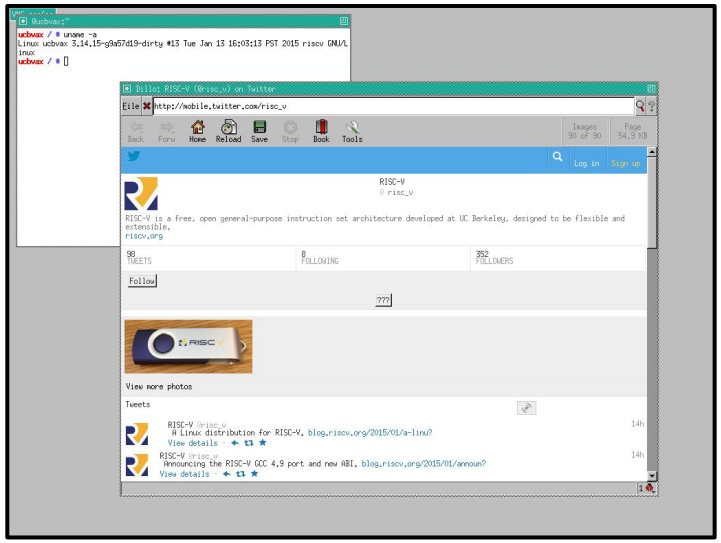
RISC-V

riscv-poky:
A Linux Distribution for RISC-V

- Compiling an application for RISC-V today:
  - Need to download and compile RISC-V toolchain + Linux
  - Download, patch and cross-compile application and dependencies
  - Create an image to run on QEMU or real hardware
- Problems with this approach:
  - **Error-prone**: Easy to corrupt the FS or get a step wrong
  - **Reproducibility**: Others can't easily reuse your porting work
  - **Rigidity**: If a dependency changes, need to do it all over
- We ported the Yocto Project:
  - A Linux Distribution generator: automate the above steps
  - Part I: **Hundreds of recipes** (scripts that describe how to cross-compile applications for different platforms)
  - Part II: **Build tool** to download, patch, cross-compile and package applications into RPM/DEB, images, SDKs
- Builds hundreds of packages out of the box!

Martin Maas

From riscv-poky to Gentoo Linux



- Why is this cool?
  - We can now easily compile a lot of complex applications, e.g., Python, Perl, LLVM, GCC, etc.
  - Bitbake is a fork of Portage, the Gentoo Linux build tool
  - Since yesterday, we have a working Gentoo port as well!

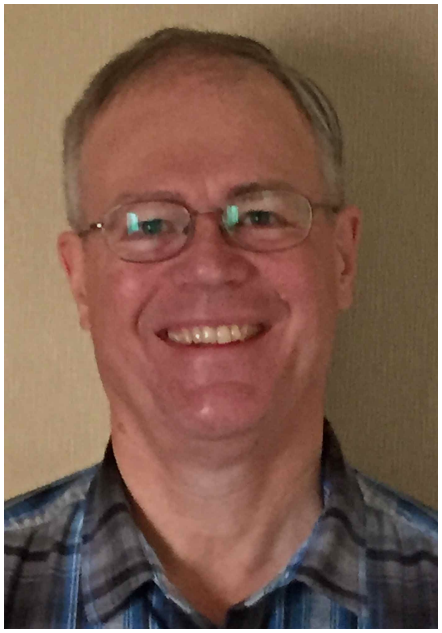
- If you port a package to RISC-V, **submit a pull request to riscv-poky**. We'll add it to Gentoo as well!
  - Want to grow the list of software available for RISC-V
- Gentoo running a web browser on RISC-V!
  → **Come to the demo!**

Palmer Dabbelt

ASPIRE
UC Berkeley
RISC·V

RICO: Fast Functional Instruction-Set Simulation

Craig Steele
Tautline LLC



Problem Area

- **Fast functional simulation for SW development for many-processor NoC-connected systems**

Current Approach

- **QEMU** (**Q**uick **EMU**lator)
  - **QEMU design seems resistant to multi-threading**
  - **Host binary translation code generator is arcane**
  - **Code base has lots of PC-related cruft**
  - **Would like to code modules in a newer language**

Our Approach

- **RICO: Code in C and Lua scripting language**
  - **Translate RISC Instructions into Lua**
  - **Let LuaJIT compiler optimize translated code**
  - **Comparable performance to QEMU for 32b RISC**
  - **Multithreaded implementation**
  - **Loadable Lua modules, Lua FFI ifc. to C libraries**

Bluespec demo of BluROCS/Flute on FPGA

- We'll connect remotely to a host+FPGA board in Massachusetts
  - Host: Linux laptop
  - FPGA Board: Xilinx VC707
  - With a PCIe connection
- On FPGA: BluROCS/Flute SoC
- On host:
  - Console I/O
  - Tandem verification with Cissr
  - riscv-gdb:
    - Connecting to BluROCS/Flute
    - Loading an ELF file
    - Setting a breakpoint
    - Running the program (with console I/O)

Rishiyur Nikhil
(with Darius Rad)

FlexPRET: A Predictable Processor for Mixed-Criticality Systems and Precision-Timed IO

Michael Zimmer

Research Area

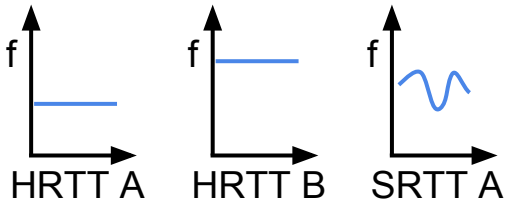
- Real-time embedded systems, focussing on **timing and safety critical tasks** that interact with sensors and actuators
  - e.g. automotive, avionics

Current Approaches

- RTOS running on traditional embedded processors
  - **unpredictable**: interrupts, unknown processor state
  - **not isolated**: test and verify entire system

Our Approach

- FlexPRET: 5-stage, fine-grained multithreaded RV32I processor
  - **Hard real-time HW threads**: constant rate (configurable)
  - **Soft real-time HW threads**: steal unused cycles
  - Open source, implemented in Chisel, deployed on Xilinx FPGA (Virtex-5, Spartan-6, Zync)



Practical RISC-V Random Test Generation using Constraint Programming

Edmond Cote

Abstract

A proof-of-concept random test generator for RISC-V ISA is presented. The test generator uses constraint programming for specification of relationships between instructions and operands. Example scenarios to cover basic instruction randomization, data hazards, and non-sharing are presented. The tool integrates the RISC-V instruction set simulator to enable the generation of self-checking tests. The tool is implemented in Python using a freely-available constraint solver library. A summary of problems encountered is provided and next steps are discussed.

Results

- I will be happy to discuss results

Objective

- Learn about RISC-V architecture ahead of workshop
- Learn about low-level constraint programming
- Demonstrate proof-of-concept random test generator
- Complete interesting technical project outside of work

PULP
Parallel processing Ultra-Low-Power platform

pj/op – mW power envelope, parallel platform for IoT apps

- Tightly-coupled clusters of simple processors operating in near threshold
- Achieve extreme energy efficiency on a wide range of operating points
- Two chips prototyped in 28nm FD-SOI technology (RVT/LVT)

But also…

- Scalable: signle-cluster (<10 mW) <<>> multi-cluster (10+ GOPS)
- Programmable: OpenMP, OpenCL, OpenVX
- Open: SW & HW

Objective: RISC-V + PULP heterogeneous platform

- *RISC-V based host*
- *PULP many core accelerator (64+ cores, 8+ clusters)*

Davide Rossi

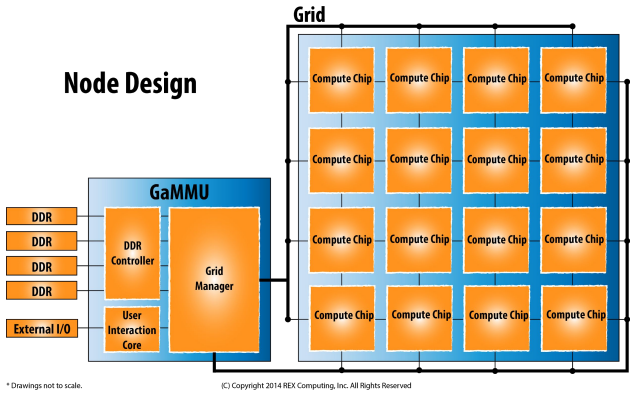
Prof. Luca Benini

REX Computing Open HPC Processor

http://rexcomputing.com

Thomas Sohmers
Paul Sebexen

- HPC Requirements
  - Floating Point and Integer
    - Very high memory bandwidth (For max throughput, 4 words per flop)
  - 50 DP GFLOPs/Watt at system level for Exascale
    - Current state of the art is ~5GFLOPs/Watt
- Processing styles
  - SIMD/Vector doesn't scale for most problems (Amdahl's Law)
  - MIMD/MPMD can (Gustafson's Law… PGAS/SHMEM)
  - Systolic Array + DSP style manycore processing
- Key decisions
  - No (managed) caches: **~30%+ area savings**
    - "You can't fake memory bandwidth that isn't there." --Seymour Cray
  - Global flat memory map for whole chip; can be extended across multiple chips in a node
  - Very simple on-chip mesh network
    - Ideal use would be similar to a Systolic Array, but user is not limited to that.
  - Parallel chip to chip interface
    - Eight 64-bit parallel interfaces per compute chip.
  - Redundant functions are centralized to a separate chip.
- (Hopeful) Results
  - 256 core compute chip (128k scratchpad per core)
  - 256 DP GFLOPS; 85 GFLOPs/Watt chip level
  - 53 GFLOPs/Watt node level
  - 384 GB/s aggregate compute chip bandwidth



Near future:
Full spec: March 2015
Cycle accurate simulator: ~March 2015
Prototype (16 core + interfaces) tapeout ~Q4 2015.

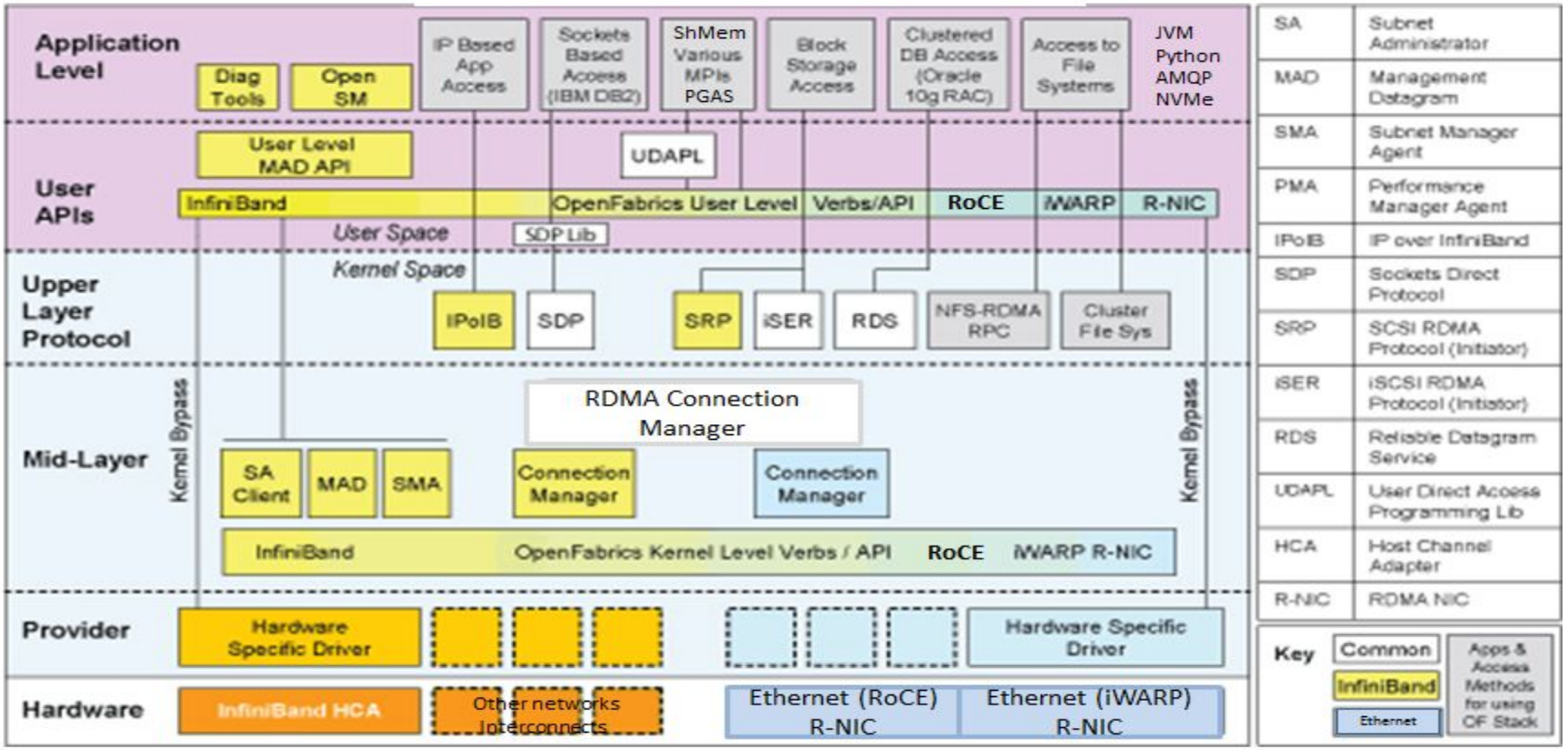
ASPIRE
UC Berkeley
RISC-V

bill.boas@openfabrics.org

OPENFABRICS ALLIANCE

Deployed on ~ 50% of Top500 machines

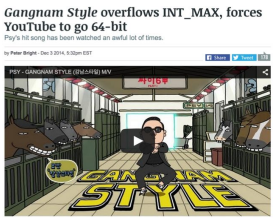
OpenFabrics Software - Linux Stack

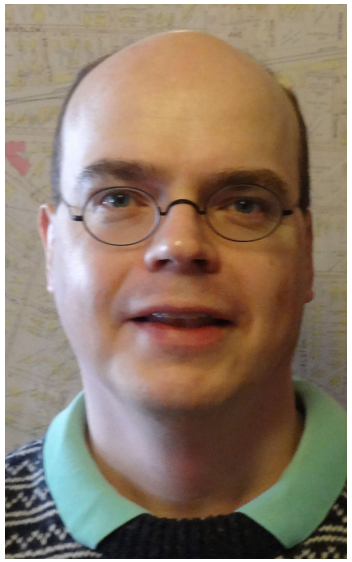


See www.openfabrics.org for OFS/OFED Linux distribution methodology or ask your Linux distributor.

Universal Integers

- C int type
  - counting objects in memory ✔
  - arbitrary external quantities ✘
  - favors performance over correctness





David Hossack

- Goal: Add **unbounded integers**
  - Small Integers ***as efficiently as C int***
  - Big Integers in software eg GMP
- Why?

Understanding Integer Overflow in C/C++

Will Dietz, Peng Li, John Regehr, Vikram Adve

Proceedings of the 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, June 2012

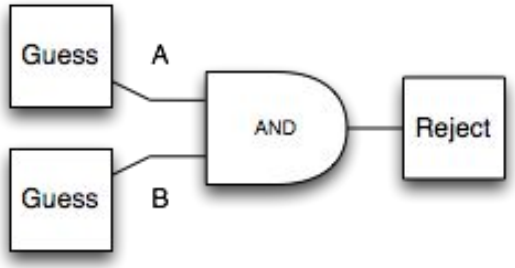
- How?

  Encoded tagged union { small_int, pointer }

- Student project?

Probabilistic Programming with ISA-Level Primitives

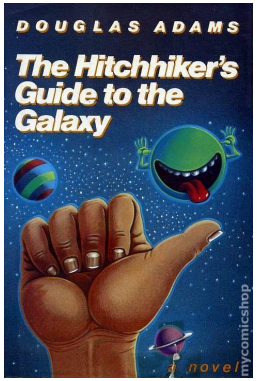
2 new elementary logic gates!
(one that will annihilates the universe)



Adam M. Smith
@rndmcnlly

A probabilistic processor design!
(impossible to physically construct)

Programming model based on fictional technology!
(*Infinite Improbability Drive*)



RISC-V + probabilistic instructions will save the day!
(ARMv6 sorta worked, but it was messy)

ASPIRE UC Berkeley

RISC-V

Probabilistic Programming with ISA-Level Primitives

- **Probabilistic programming is a thing these days**
  - using programs to define probability distributions
  - fancy algorithms answer questions about them
- **PP languages are still out of reach for most programmers**
  - unfamiliar statistics vocabulary
  - weak tool support
  - black-box model: poor search/inference performance
  - white-box model: severe restrictions on computation
- **Dream: black-box flexibility + white-box efficiency**
- **Idea: implement a probabilistic processor**
  - expose new instructions with C wrappers
  - reuse existing languages and compilers to produce machine code
  - apply symbolic simulation to reason over all possible executions (a twist on probabilistic model checking)
- **Prototypes**:
  - An ARMv6 probabilistic processor (barely works)
    - can generate and solve math textbook exercises given a student's mental model described in C
  - Python library for rapid prototyping of probabilistic software
- **Future:**
  - switch to RISC-V; use a real HDL; apply logic minimization; ...
  - think like a processor designer, not like an AI/PL/stats researcher
  - interesting new uses for an open ISA; fresh ideas for AI/PL/stats

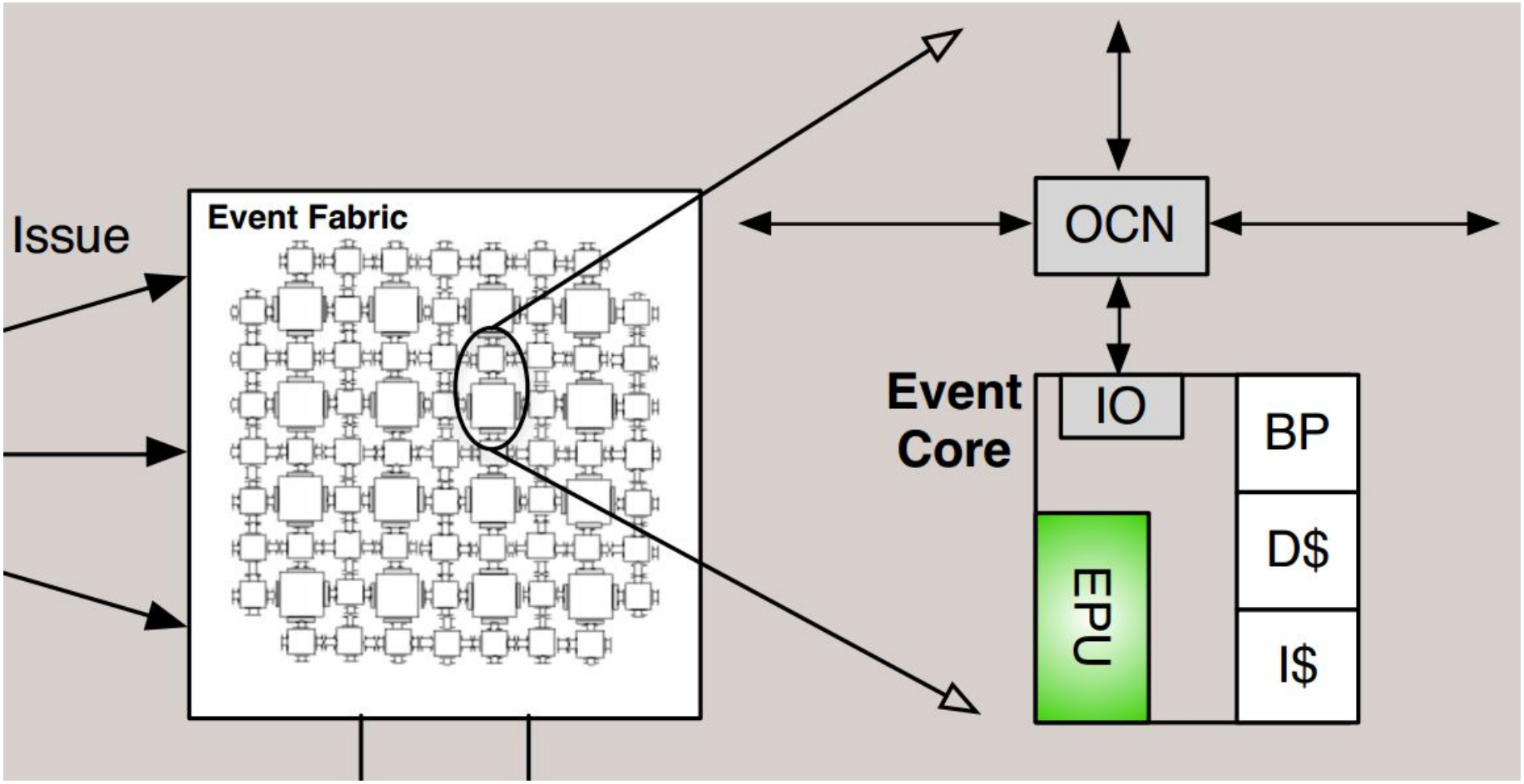
Adam M. Smith
@rndmcnlly

Eve: An Event-Driven Architecture For Asynchronous Programming Models

- Observation:
  - Architecture innovations driven by applications characteristics and software programming model upheaval (e.g., GPUs)
- Event-driven model as a new programming model and application class
  - IoT, sensor-rich environment
  - Mobile devices
  - Cloud (e.g., Node.js)
- General-purpose CPUs are ill-suited
  - Locality
  - Parallelism

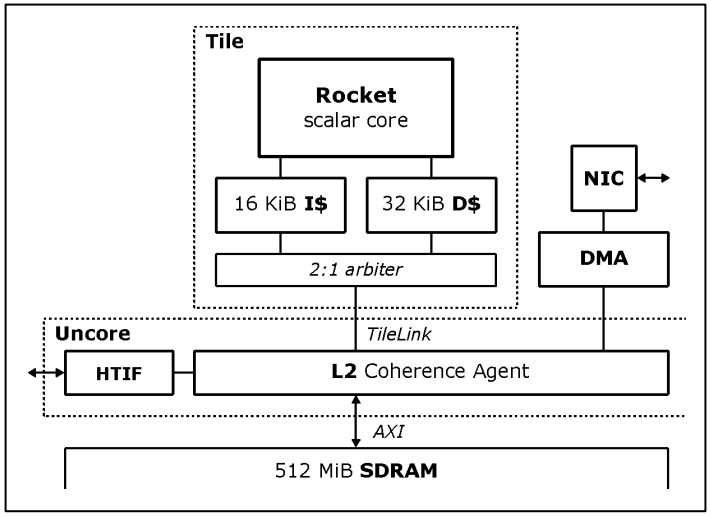
Yuhao Zhu
yuhaozhu.com
UT Austin

Eve: An Event-Driven Architecture For Asynchronous Programming Models



Hardware Acceleration of Key-Value Stores

- Datacenter apps, path through CPU/kernel/app ≈ 86% of request latency
- Goal: Serve popular Key-Value Store GET requests without CPU
- Soft-managed cache attached to NIC, RoCC CPU interface
- Benchmarking on FPGA:
  - RISC-V Rocket @ 50 MHz
  - NIC from TEMAC/PCS-PMA + 1 Gb SFP
  - Hardware KV-Store Accelerator
  - Traffic Manager, DMA Engine
- Written in Chisel

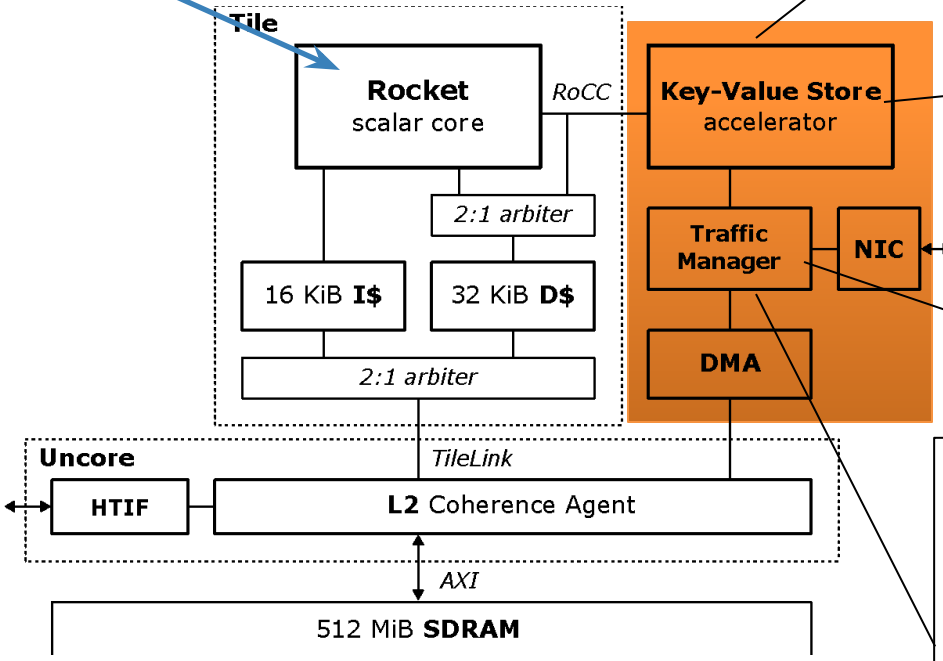


Base System: First Physical RISC-V System with Networking Support

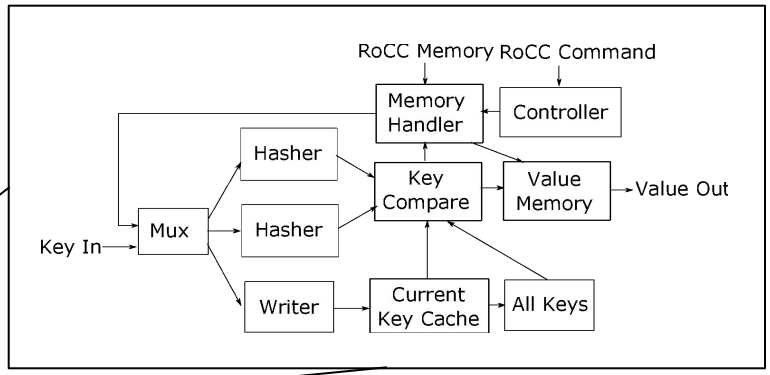
Sagar Karandikar
Howard Mao
Albert Ou
Yunsup Lee
Soumya Basu
ASPIRE Lab, UC Berkeley

Hardware Acceleration of KV Stores - System Design

Runs `memcached`

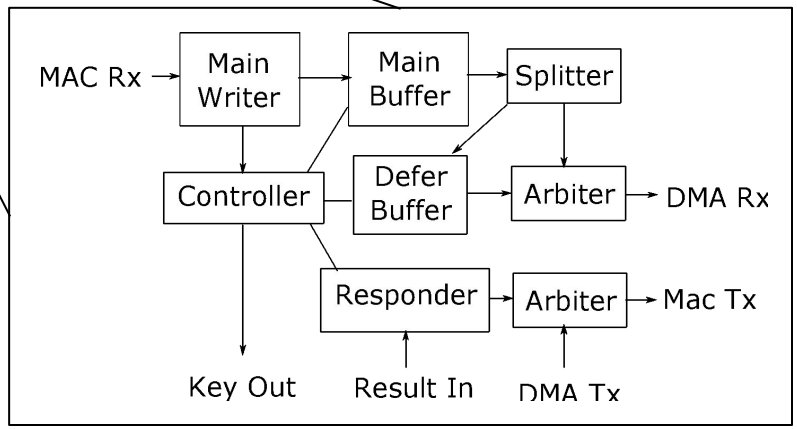


1 Gb SFP Module, Network



Accelerator

Traffic Manager



Implemented on Xilinx ZC706 Evaluation Platform

Hardware Acceleration of KV Stores - Evaluation

- Raw Performance: ~10x Latency Reduction
  - Software: ~1700 µs
  - Accelerator: ~150 µs
- Replacement policy is software-tunable
- Realistic benchmark based on appx. of internal Facebook traces:
  - Handled 40% of requests @ 10x latency reduction