Coyote: A RISC-V simulator for large scale HPC-like architectures

Borja Pérez
borja.perez1@bsc.es

05/05/2022
MEEP Goals

• What type of HPC Exascale accelerator would you like build in 2025?
  • MEEP is a flexible FPGA-based emulation platform that will explore hardware/software co-designs for Exascale Supercomputers
    • Software Development Vehicle
    • An evaluation platform of pre-silicon IP and ideas, at speed and scale
      • Efficiently Map RTL to FPGAs (not gate2gate)

• Flexible and Reusable
  • Enable other SW and HW projects @ in the EU
    • Physical platform
    • Emulation IP
    • Open SW and HW IP
MEEP in a Nutshell: A Co-Design Laboratory

1. Software Ecosystem
2. Hardware (RTL)
3. FPGAs

Full Stack Exascale Design
- HPC/DA Applications
- HPC/DA Runtimes
- Virtualization (Containers)
- OS
- FPGA Emulated HW

Accelerator Architecture & RTL

10 x

Software Ecosystem:
- Python
- C/C++
- Java

Hardware (RTL):
- Host Server
- FPGA Shell
- Accelerator IP Package
- HBM

Virtualization (Containers):
- OS

HPC/DA Applications:
- Fully Connected topology

HW/SW Tools: LLVM, GDB, Profiling, Performance Monitors
The ACME architecture (Accelerated Compute & Memory Engine)
MEEP Simulation needs

- A simulation infrastructure for **design space** exploration of HPC architectures.
  - RISC-V support.
    - Both scalar and vector instructions.
  - Strike a balance between fidelity and simulation throughput
    - Focus on the memory hierarchy and data movement.
  - Flexible and easy to extend.
  - Scalable.
  - Leverage existing tools.
    - No simulation infrastructure is capable of the needed scalability/flexibility out of the box.
    - Let’s integrate two of them.
Enter Coyote

• **Coyote**, a simulation infrastructure for design space exploration of HPC architectures based on Spike and Sparta.
  • Spike is the golden standard for RISC-V functional simulation.
  • Sparta is a modeling framework developed by SiFive.

• Let’s put them together.

• But...
  • The samples provided with Sparta are trace-driven and focus on the core.
  • Spike is an ISS. It does not know about timing or events.
Implementation approach

- Spike and the model built with Sparta are two independent entities.
Current modeling capabilities

- Scalar core + L1 caches
  - In-order core
  - Configurable L1 geometry (Spike)
  - Vector instruction support (Spike)
    - Version 0.8
  - Tracking of RAW dependencies
  - Instruction Latencies
  - MSHRs
  - CGMT
    - Modeled as several actual spike cores sharing an L1, but having just one executing at a time
    - Configurable thread switch penalty
Current modeling capabilities

• L2 Slices (Can be used as an L3 too)
  • Configurable geometry
    • Associativity, #Banks, line size...
  • Configurable hit and miss latencies
  • Different data mapping policies
    • Set interleaving
    • Page to bank
  • Different cache sharing policies
    • Tile private
    • Fully shared

• Crossbar interconnecting L2 and NoC router
Current modeling capabilities

- Three different NoC simulation models for different fidelity/simulation throughput
- Functional
  - Fixed latency for any message sent
  - Sensitivity tests
- Simple
  - Takes into account the placement of the source and destination
  - Heatmaps
- Detailed
  - Uses the booksim simulator
  - Models contention, congestion...
  - Tested for a mesh topology
Current modeling capabilities

- Memory controller
  - Configurable memory geometry
    - #Banks, rows, columns...
  - Configurable DRAM latencies
  - Supports a subset of DRAM commands
    - PRECHARGE, ACTIVATE, READ and WRITE
  - Configurable command and memory request schedulers
  - Configurable address mapping
    - Row Bank Column with the Bank Group Interleave
    - Row Column Bank
    - Bank Row Column
    - ...
Supported applications

• Coyote currently supports baremetal applications
  • Limitation of the underlying spike
  • Avoid syscalls; data needs to be static...

• Compiled with the standard riscv-gnu-toolchain

• Current applications:
  • Stream benchmark
  • Scalar matrix multiplication
  • Vector matrix multiplication
  • Vector SpMV
  • Vector somier
  • Vector axpy

• Simulation throughput of ~6 MIPS aggregate (modeling up to 128 cores).
  • Running @ laptop, 5th gen. Intel i7, 16GB RAM
Applications, outputs and performance

• Outputs common statistics:
  • Cycles
  • Miss rates
  • Number of stalls
  • Loads
  • Queue occupancies
  • Latencies
  • Number of bytes read/written
  • … and many more

• Produces a human-readable trace that can be parsed into the paraver format*

* Paraver: https://tools.bsc.es/paraver
Example visualization of a trace

- Evolution of the total number of NoC packets received per destination tile. Tiles labelled in yellow are memory tiles. Green means fewer, blue means more.
Future (& current) work

• To be released shortly under Apache 2.0 license.

Version 1.0
• All the above
• Coherence
• Back pressure

Version 1.1
• Virtual memory
• Vector processing revisited

Version 2.0
• Syscalls
• Vector extension 1.0