Crossing The Line: HW/SW Snapshot Interface

A proof of concept by

Giorgio Ajmone, Shanqing Lin and Ayan Chakraborty

EPFL

Simulation Bottleneck

Today's simulators are slow

- Timing simulators are 20~250 KIPS
- Trace-driven simulators are 1~4 MIPS

Real hardware is fast ~ 2 GIPS (250-8000x faster)

- FPGAs offer the possibility to model any architecture
- Programming FPGAs is an order of magnitude more complex than writing software
- We cannot observe what is happening inside the hardware

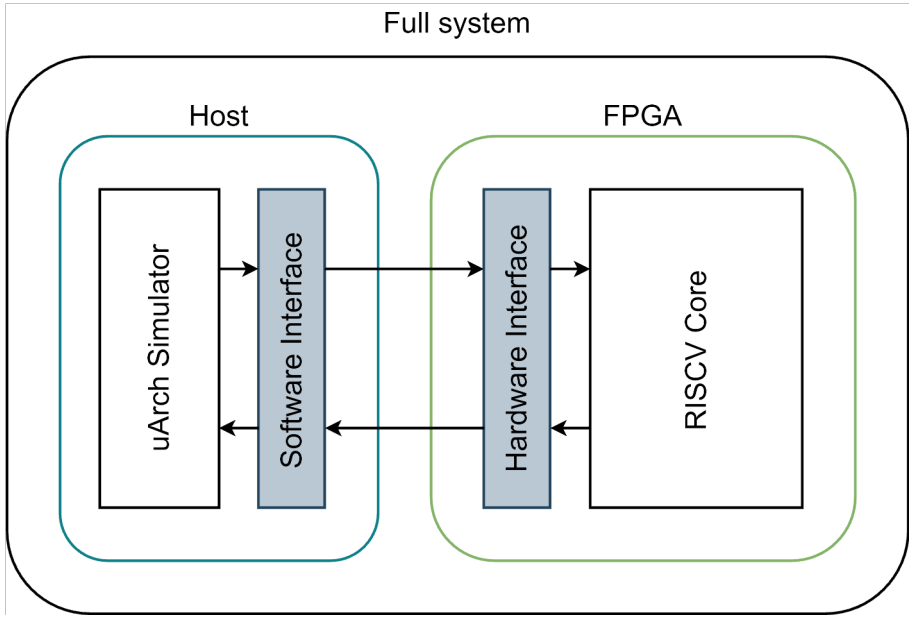
Statistical Sampling [Wenish'06]

- Apply statistical sampling to μArch simulation
- Sampling simulation requires warming
  - Generate microarchitectural state (functional warming) with traces
  - Collect a checkpoint library
  - Run detailed/timing simulation over individual checkpoints

Proposal

Seamlessly integrate real hardware on FPGA with timing simulation

- Perform statistical sampling, combining the speed of FPGA simulation with the instrumentation offered by µArch simulators



RTL- µArch Simulator Integration

RTL requirements

- Provide an interface to stop/restart the execution, canonicalize the system and read/write the hardware state
- Export/import a snapshot of the full system with its software and microarchitectural state

µArch simulator requirements

- Modify the code to load and export a compatible snapshot
- Model correctly the components on FPGA

Hardware-Software Interface

Stop/restart the execution

- User needs to be able to freeze and restart the hardware
- Hardware needs to be able to request for a system halt

Canonicalize the system

- To export/import a snapshot, the system needs to be in a stable state (e.g., fully-drained pipeline and no in-flight operations in the cache)

Read/write the hardware state

- We need to expose the hardware state to the user (e.g., read/write the content of the cache, memory and register file/PC)

Prototype

Core specifics

- In-order with 32-bit register file
- 2-way set-associative L1i and L1d caches (8+8 KB)
- 4-way set-associative L2 cache (16 KB)
- 64 MB main memory
- UART8250 Interface

Software Interface

- Connectal-based interface to import/export JSON snapshots and control the hardware simulation

Status

Achieved

- Stop the execution and freeze the hardware
- Restart the execution without perturbation
- Canonicalize the system
- Extract the hardware state of the various components
- Import the hardware state coming from a different machine

Future challenges

- Complete the integration with gem5 simulator to transfer snapshots
- Export main memory using Connectal DMA

Validation

- Regression testing with the course tests
  - Export/import snapshots of the tests between different machines
- Ad-hoc tests designed to verify the correctness of the interface
  - FPGA requests the Host to halt the simulation, export a snapshot and restart the system
  - Import the snapshot on a different machine and continue the execution starting from the moment the snapshot was taken
- Export a booted Linux system and run it inside a timing simulator
  - Avoid the necessity to boot Linux inside a timing simulator

Demo

Acknowledgements

- Bugra Eryilmaz for its processor and **awesome** caches

- Florian Hofhammer and Sankalp Gambhir for their support on Linux booting and UART interface

QUESTIONS?