Hardware Emulation: A New Approach to the Rapid Prototyping of Multiprocessors

Luiz André Barroso

RPM Project Group

Michel Dubois (PI)

Luiz André Barroso

Koray Öner

Jaeheon Jeong

Previous contributors

Krishnan Ramamurthy

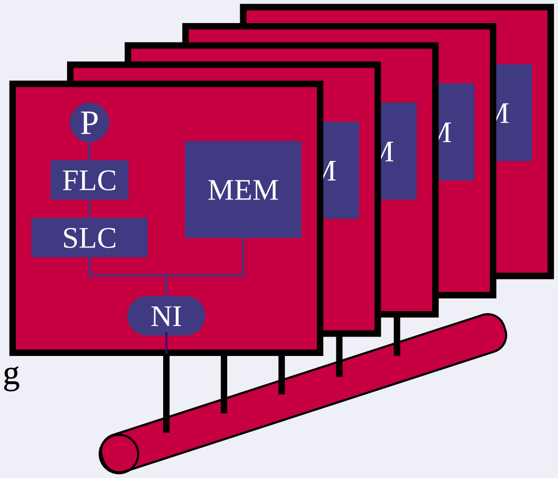
Sasan Iman

Jacqueline Chame

Per Stenstrom

Massoud Pedram

Introduction



Scope:

- Multiprocessor system design

Features:

- memory organization
- shared-memory vs message passing
- cache and protocol design
- consistency model
- interconnection fabric
- software/hardware trade-offs

Evaluate:

- performance
- cost / complexity
- correctness / validation

Methods:

- Software Simulation
- Prototyping
- Hardware Emulation

Introduction (cont.)

Software Simulation:

- relatively inexpensive
- versatile
- slow
- accuracy ?
- unable to handle real workloads (OS, system software, etc.)
- insight on actual design ?

Breadboard Prototyping:

- very expensive
- accurate
- fast
- few design points
- typically hard to observe

Introduction (cont.)

Hardware Emulation:

- actual implementation
- faster than software simulation
- allows the study of large applications, including operating systems
- allows the study of a large design space
- detailed monitoring
- less expensive than most prototypes

  time scaling

  the same hardware emulator is re-used

OUTLINE

- Introduction
- Software Simulation
- FPGA-based Rapid Prototyping Systems
- Hardware Emulation in RPM
- Measuring Performance
- Conclusion

Software Simulation

Breaking down the overhead of software simulation:

1. Overhead of simulating processor execution

2. Semantic gap

3. Need to keep a simulated clock

4. Target system speedup

Software Simulation (cont.)

Simulation of processor execution:

- Direct execution (Tango, WWT)
  - fast when target ISA is similar to host ISA
  - slow if instructions and private data activity is relevant
  - code has to be instrumented
- ISA simulation (CacheMire-2)
  - overhead of instruction decoding/execution

Semantic gap:

- Depends on how detailed the simulator is
- Example: CacheMire-2 8-processor SPLASH simulations
  - SLCacheAcess executes 210.5 instr./call
  - 1380 to 3130 simulator instructions/target instruction simulated

Software Simulation (cont.)

Handling simulated time:

- Event calendars
  - scheduling/context switching (61% in TangoLite)
  - hard to parallelize
- Activity scanning
  - no context switching
  - fixed overhead to scan for activities in every simulated cycle
  - even harder to parallelize

Target system speedup:

- Simulators are typically fast on hits and slow on other events
- Example: CacheMire-2 executing MP3D for 8 Processors
  - Over 80% of the simulator time is spent on references that miss

Software Simulation (cont.)

Parallel Software Simulation

- Makes use of existing high-performance parallel computers (WWT)
- Problem: how to preserve the order of target events ?
  - ⇒ distribute the event list and exchange time-stamped messages
- Conservative approach: periodic barriers (WWT)
- Optimistic approach: checkpointing/backtrack (time-warp)

FPGA-based Rapid Prototyping Systems

New technology: High-density in-circuit reprogrammable circuits

- 13,000 gates of reprogrammable gate arrays (Xilinx X4013)
- 1,024 pins Field-programmable interconnect circuits (Aptix FPICs)
- High-level design languages (VHDL)
- Improving synthesis tools

Typical configuration:

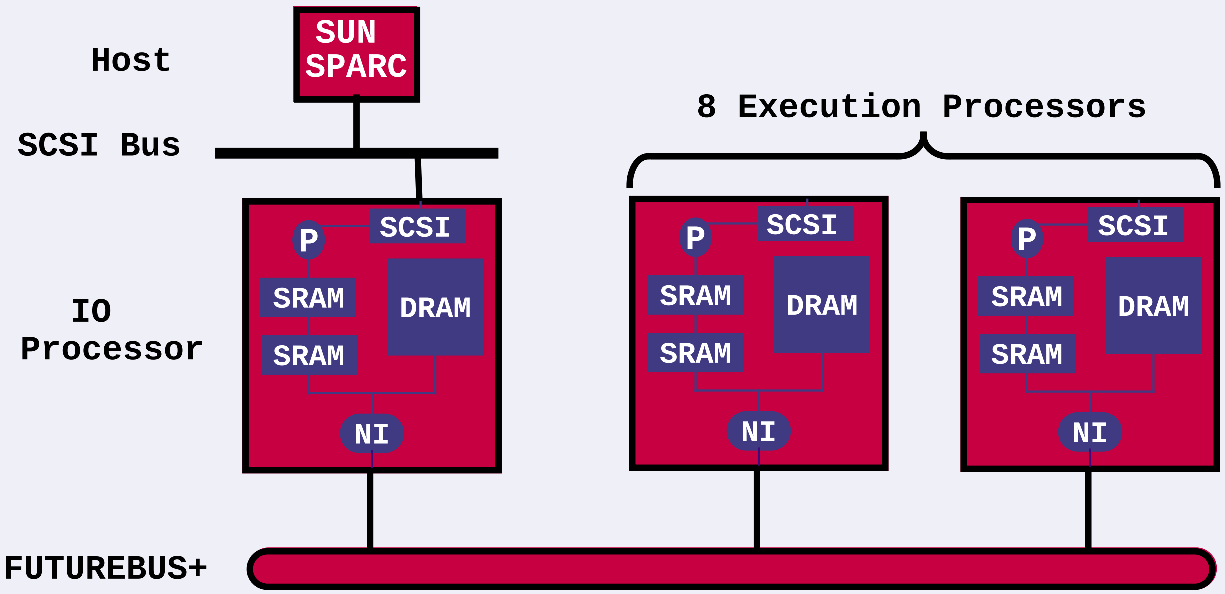
- Array of FPGAs on a board + interconnection logic + I/O

Example:

- *Quickturn* emulation system (RPMplus) (50 Kgates to 6 Mgates)

Hardware Emulation in RPM

RPM Architecture



Hardware Emulation in RPM (cont.)

Making cost-effective use of emulation technology:

- Restrict reconfigurable hardware to points of interest
- FPGAs are used for the caches and memory/directory controller only
- Use of mature technology in the hardware implementation
- Controllers are clocked faster than the processors
  - simplify controller circuitry
  - avoids saturation of the controllers and interconnects
  - adds flexibility
  - extra cycles are used for performance monitoring

Hardware Emulation in RPM (cont.)

Main features:

- Flexibility comparable to a software simulator
  - ❒ different cache sizes, block sizes, associativities, replacement schemes, coherence protocols, buffering strategies, consistency models
  - ❒ extensions to the ISA
  - ❒ COMAs and CC-NUMAs
  - ❒ support for message passing
  - ❒ almost every activity or event can be monitored
- Emulator is an actual computer
- Emulation is very close to actual implementation
- No time-stamps required. RPM timing emulation is based on ***Time Scaling***

Hardware Emulation in RPM (cont.)

Time Scaling

Every resource (interconnect, caches, memories, I/O units) is characterized by two performance measures: latency and bandwidth

The timings of all system components can be adjusted to take as many Pclocks in RPM as they would take in Pclocks of the target system

How it is accomplished:

- The processor is "clocked" once every 8 cycles
  ⇒ all controllers and data transfers are too fast (in Pclocks) with respect to all target systems of interest
- All controllers are artificially delayed to match the relative speed of the target system being emulated
- A "Delay Unit" delays the sending of messages in the system bus
- Emulated I/O can be delayed by software + standard interrupt timer

Hardware Emulation in RPM (cont.)

Performance of RPM

RPM vs. CacheMire-2 running on a SPARCStation10

Benchmark (#procs)	Number of references	simulator/ target instructions	Simulation Rate (CacheMire) (cycles/sec)	Speedup (RPM/CacheMire)
MP3D (8)	18.5 M	3130	3,786	330
WATER (8)	136.5 M	1380	3,960	315
CHOLESKY (8)	79.5 M	1718	3,426	365

Slowdown Factors Between Target and RPM

Target Uniprocessor Speed	50 MIPS	100 MIPS	200 MIPS	500 MIPS
Slowdown	40	80	160	400

Measuring Performance

COUNT MEMORY in each level of the memory hierarchy

- Software controlled event counting

  Access only one memory location at a time

Table 1: Example for counting events in FLC: Private, Shared, Read, Write, Hit, Miss

Counter Address	Private/ Shared	Read/ Write	Hit/ Miss	Basic Events
0	0	0	0	Shared-Write-Miss
1	0	0	1	Shared-Write-Hit
2	0	1	0	Shared-Read-Miss
3	0	1	1	Shared-Read-Hit
4	1	0	0	Private-Write-Miss
5	1	0	1	Private-Write-Hit
6	1	1	0	Private-Read-Miss
7	1	1	1	Private-Read-Hit

Conclusion

- ❐ Hardware emulation is a promising methodology
- ❐ Time scaling allows accurate emulation with inexpensive hardware
- ❐ Potential to largely outperform software simulation
- ❐ Possible uses:
  - Rapid prototyping of cache coherence protocols
  - Validation of hardware/software architecture schemes
  - Study general purpose application performance
  - Trace generation
  - Performance tuning of parallel programs

See Also

WWW page:

http://www.usc.edu/dept/ceng/dubois/RPM.html

Papers:

RPM: A Rapid Prototyping Engine for Multiprocessor Systems, IEEE Computer, February 1995

The Design of RPM: An FPGA-based Multiprocessor Emulator, FPGA'95, February 1995.