ProtoFlex Tutorial:
Full-System MP Simulations Using FPGAs

Eric S. Chung, Michael Papamichael, Eriko Nurvitadhi,
James C. Hoe, Babak Falsafi, Ken Mai

PROTOFLEX

Computer Architecture Lab at

Carnegie Mellon

Our work in this area has been supported in part by NSF, IBM, Intel, Sun, and Xilinx.

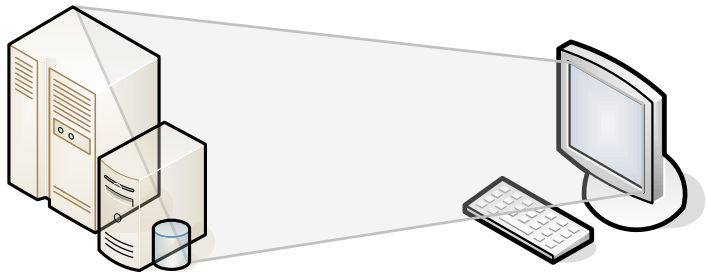
Overview

- **Part I: Basic ProtoFlex Concepts**
  - FPGA Virtualization techniques
  - BlueSPARC FPGA-accelerated simulator
- **Part II: Hands-on Technology Preview**
  - Access CMU's remote FPGA infrastructure
  - Compile and run code within an FPGA-accelerated simulation of 16-CPU UltraSPARC III server
  - Run real-time CMP cache simulations

Part I: Basic Concepts

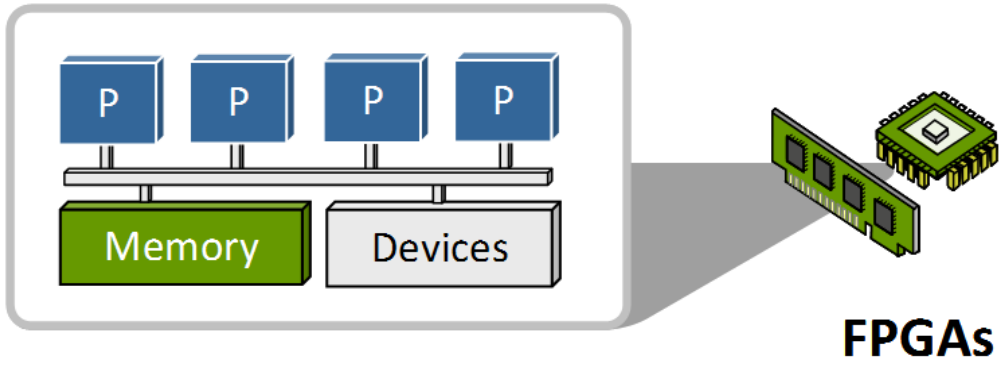
Full-system Functional Simulators

- **Convenient exploration of real (or future) HW**
  - Can boot OS, run commercial apps



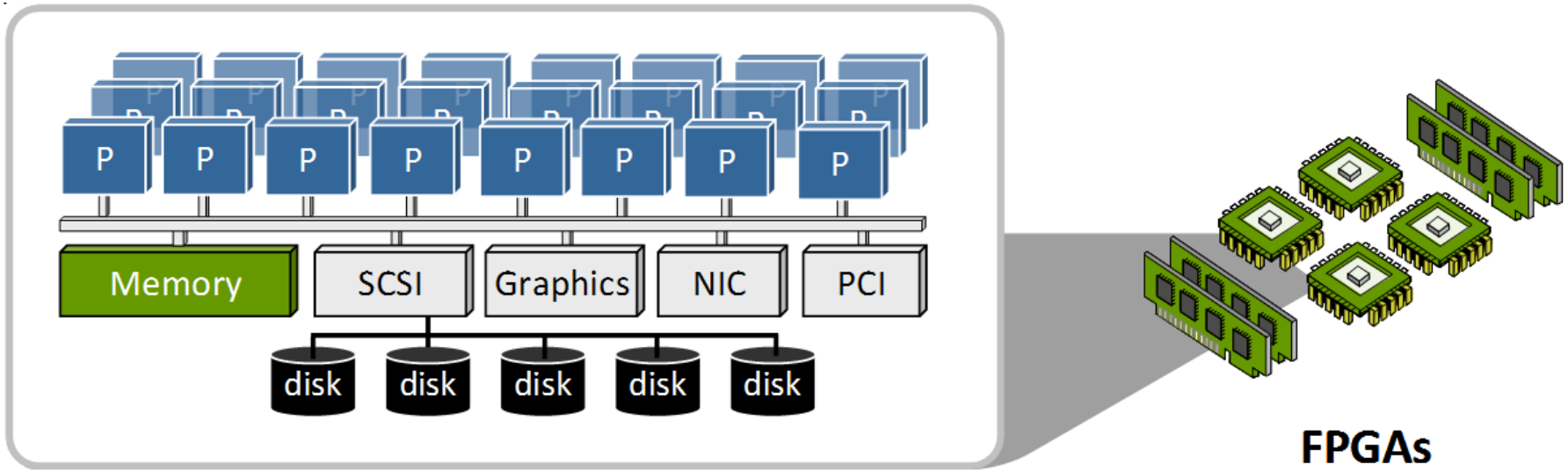
- **But too slow for large-scale (>100-way) MP studies**
  - Existing functional simulators single-threaded
  - High instrumentation overhead

FPGA-based simulation



- **Advantages**
  - Fast and flexible
  - Low HW instrumentation performance overhead

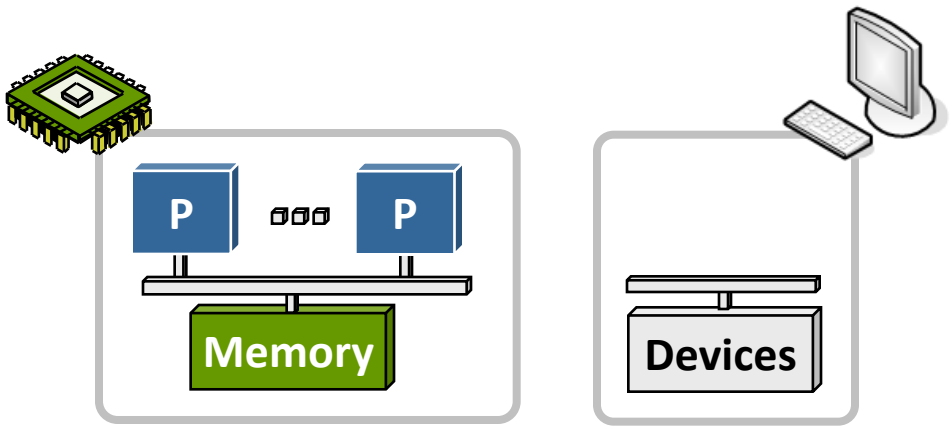
FPGA-based simulation



- **Caveat: simulation w/ FPGAs more complex than SW**
  - Large-MP configurations (>100) ➔ *expensive to scale*
  - Full-system support ➔ *need device models + full ISA*

Reducing Complexity

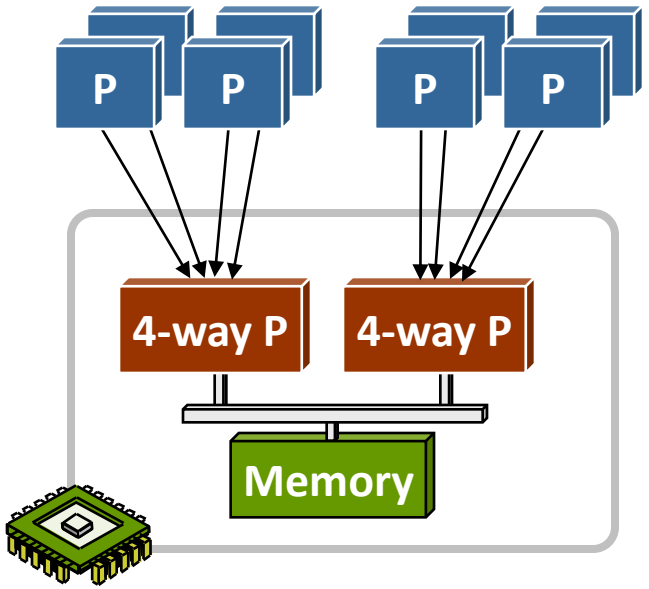
① Hybrid Full-System Simulation



Common-case behaviors

Uncommon behaviors

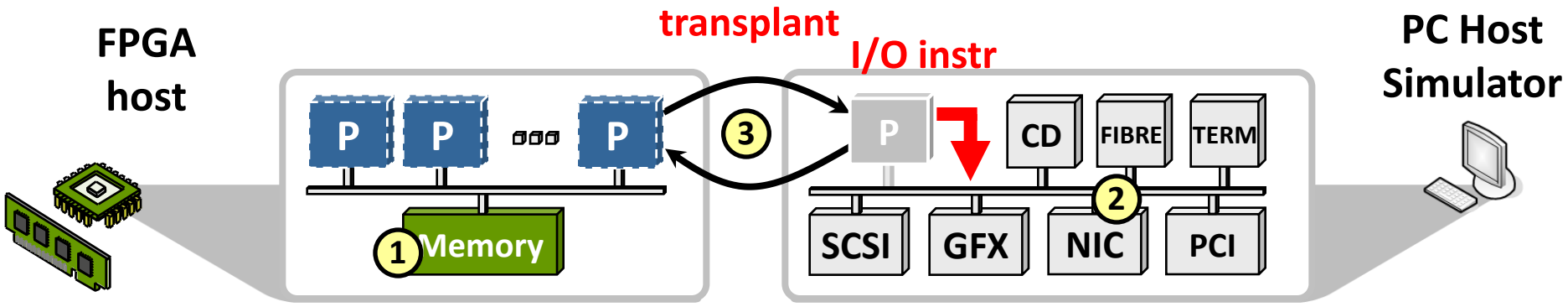
② Multiprocessor Host Interleaving



Outline

- Hybrid Full-System Simulation
- Multiprocessor Host Interleaving
- BlueSPARC Implementation
- Conclusion

Hybrid Full-System Simulation



- **3 ways to map target components**

  FPGA-only (1) Simulation-only (2) Transplantable (3)

- **CPUs can fallback to SW by "transplanting" between hosts**
  - Only common-case instructions/behaviors implemented in FPGA
  - Complex, rare behaviors relegated to software

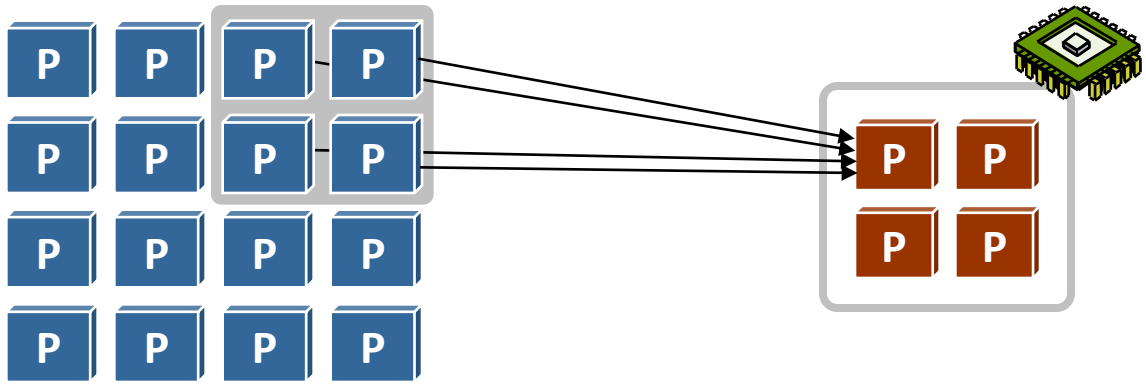
Transplants reduce full-system complexity

Multiprocessor Host Interleaving

Advantages:

- Trade away FPGA throughput for smaller implementation
- Decouple logical simulated size from FPGA host size
- Host processor in FPGA can be made very simple

4-to-1 host interleaving



FPGA Host Processor

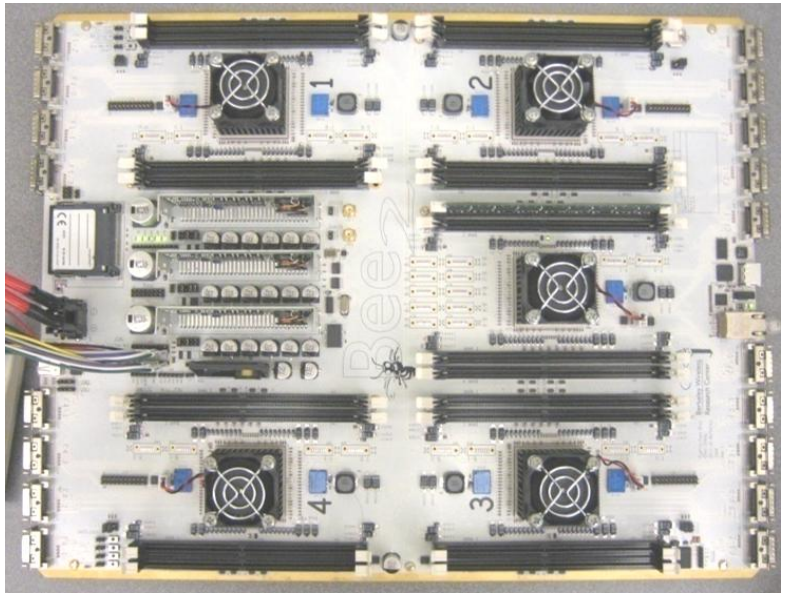
- **Architecturally executes multiple contexts**
  - Existing multithreaded micro-architectures are good candidates
- **Our design: Instruction-Interleaved Pipeline**
  - Switch in new processor context on each cycle
  - Simple, efficient design w/ no stalling or forwarding
  - Long-latency tolerance (e.g., cache miss, transplants)
  - Coherence is “free” between contexts mapped to same pipeline

Outline

- Hybrid Full-System Simulation
- Host MP Interleaving
- **BlueSPARC Implementation**
- **Conclusion**

BlueSPARC Simulator

- **Functionally models 16-CPU UltraSPARC III Server**
  - HW components are hosted on BEE2 FPGA platform

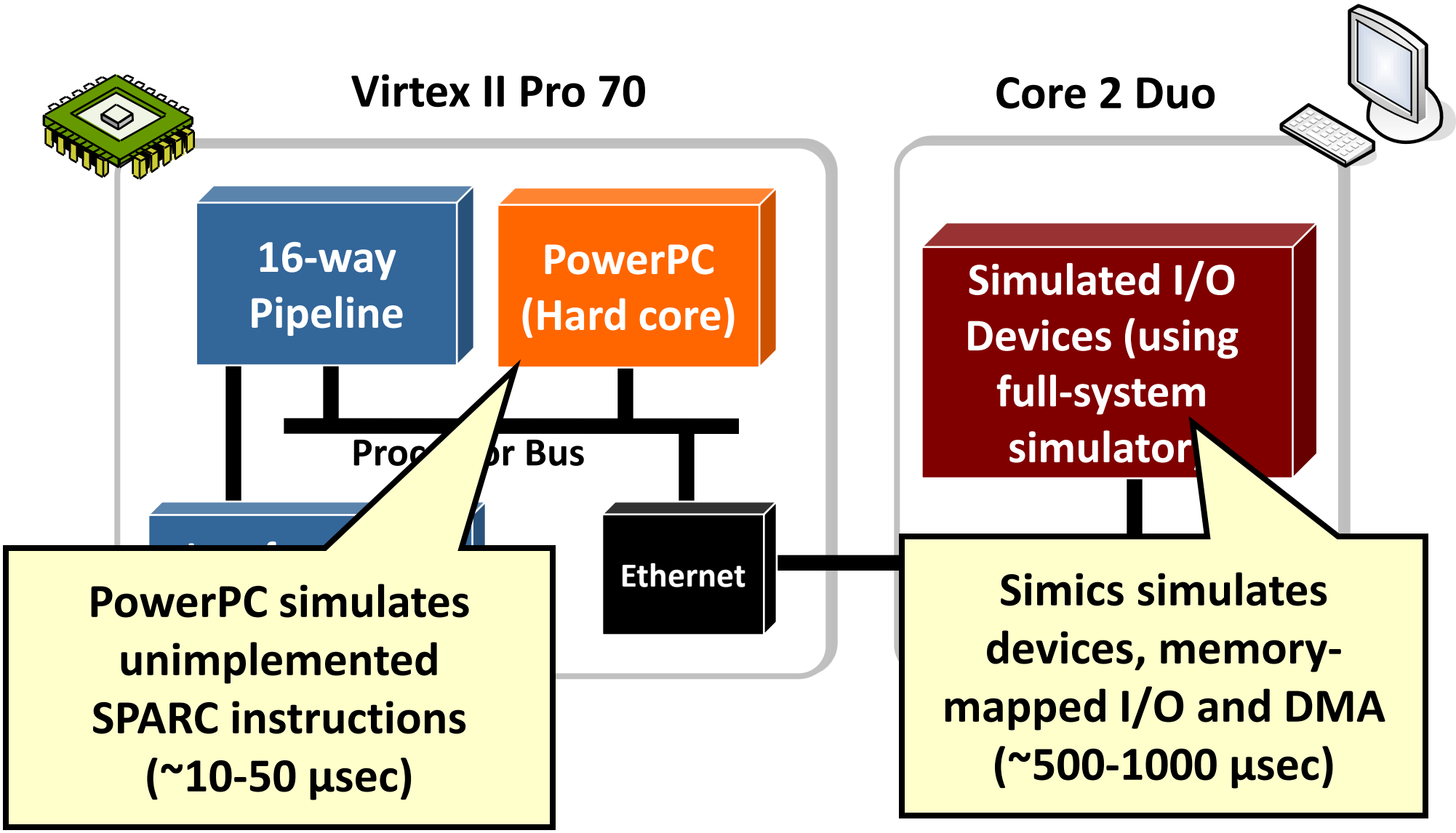


Berkeley Emulation Engine 2

- 5 Virtex-II Pro 70s (~66 KLUTs)
- 2 embedded PowerPCs per FPGA
- Only use 2 out of 5 FPGAs (pipeline + DDR controllers)

- **Virtutech Simics used as full-system I/O simulator**

BlueSPARC Simulator

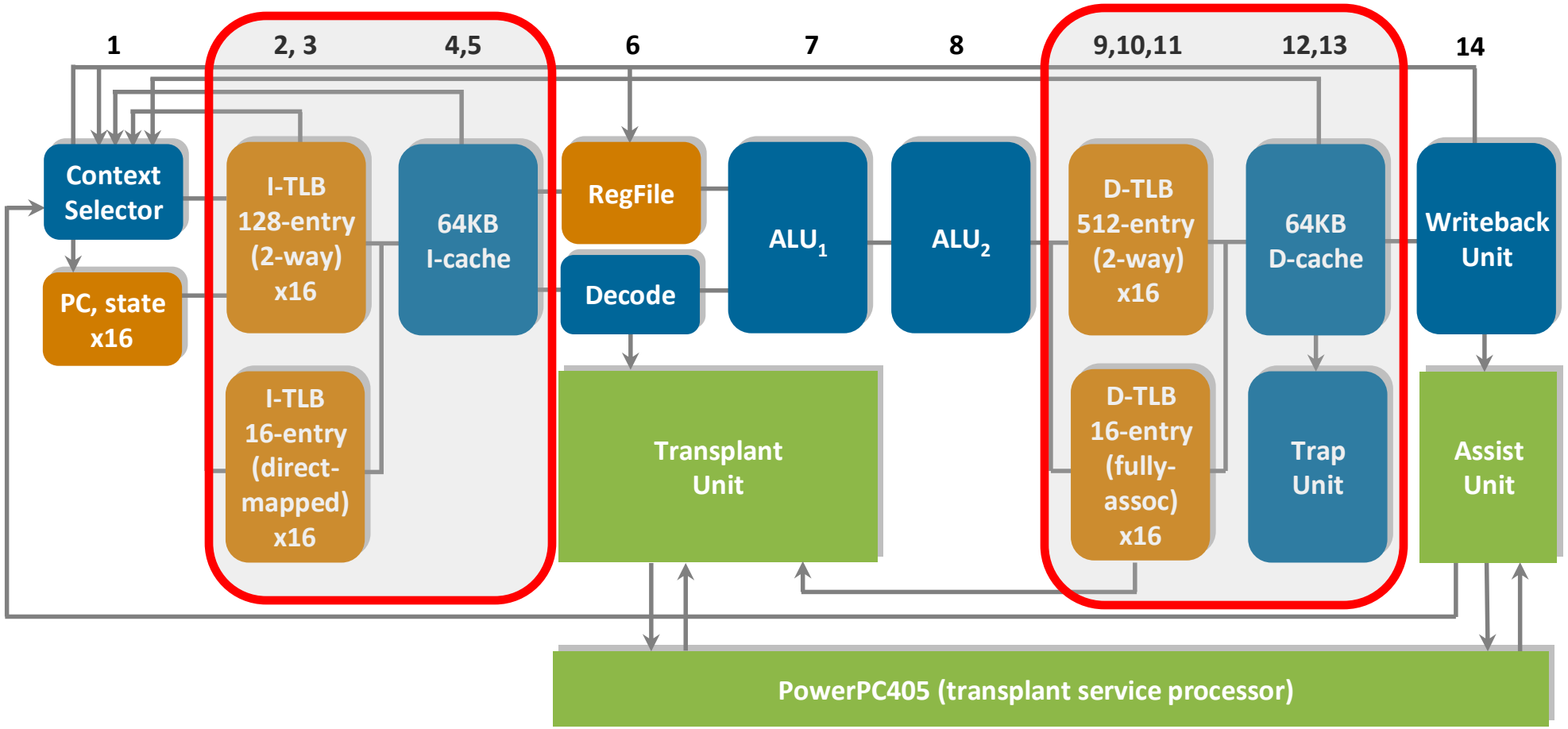


Hybrid host partitioning choices

BlueSPARC	PowerPC405	Simics on PC
• Integer ALU • Register Windows • Traps/Interrupts • I-/D-MMU • Memory + Atomics • 36% special SPARC insns	• Multimedia • Floating Point • Rare TLB operations • 64% special SPARC insns	•PCI bus •ISP2200 Fibre Channel •I21152 PCI bridge •IRQ bus •Text Console •SBBC PCI device •Serengeti I/O PROM •Cheerio-hme NIC •SCSI bus

Implementation = 60% of basic ISA,
36% of special SPARC insns, 0% devices

BlueSPARC host microarchitecture



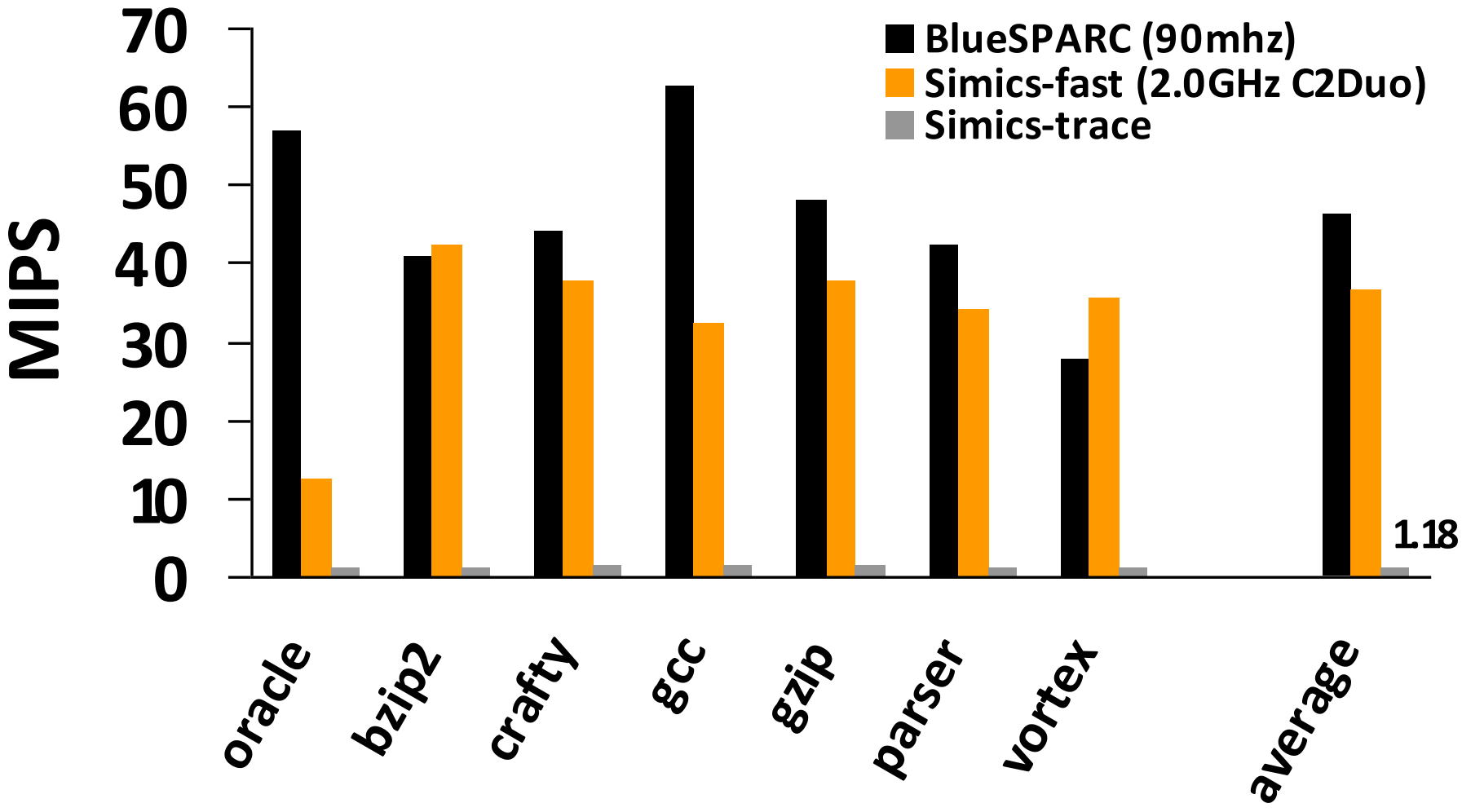
64-bit ISA, SW-visible MMU, complex memory

➔ high # of pipeline stages

BlueSPARC Simulator (continued)

Processing Nodes	16 64-bit UltraSPARC III contexts 14-stage instruction-interleaved pipeline
L1 caches	Split I/D, 64KB, 64B, direct-mapped, writeback Non-blocking loads/stores 16-entry MSHR, 4-entry store buffer
Clock frequency	**90MHz on Xilinx V2P70**
Main memory	4GB total
Resources (Xilinx V2P70)	33.5 KLUTs (50%), 222 BRAMs (67%) w/o stats+debug 43.2 KLUTs (65%), 238 BRAMs (72%)
Instrumentation	**All internal state fully traceable** **Attachable to FPGA-based CMP cache simulator**
Statistics	25K lines **Bluespec HDL**, 511 rules, 89 module types
Software	**Runs unmodified Solaris + closed-source binaries**

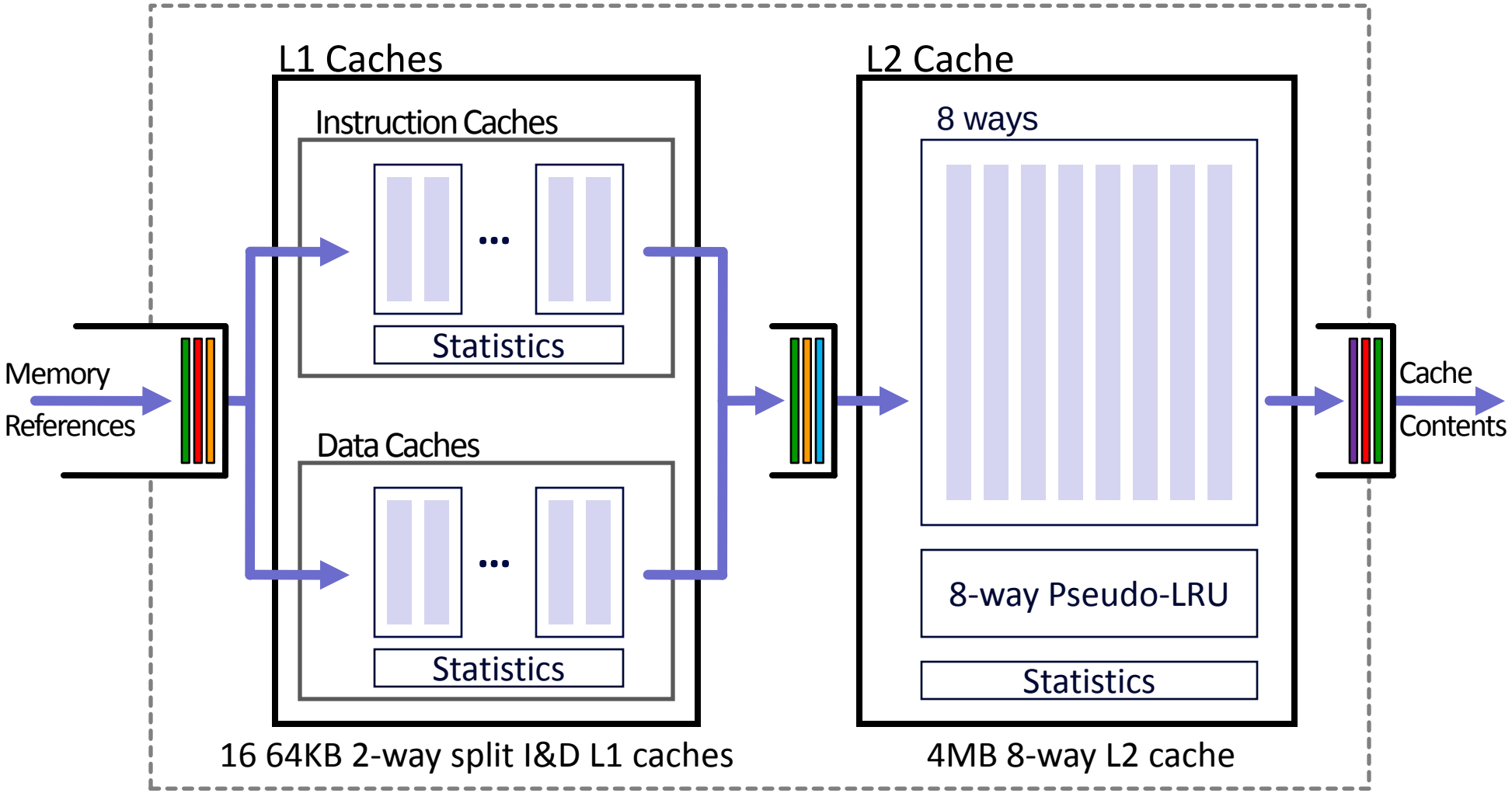
Performance



39x speedup on average over Simics-trace

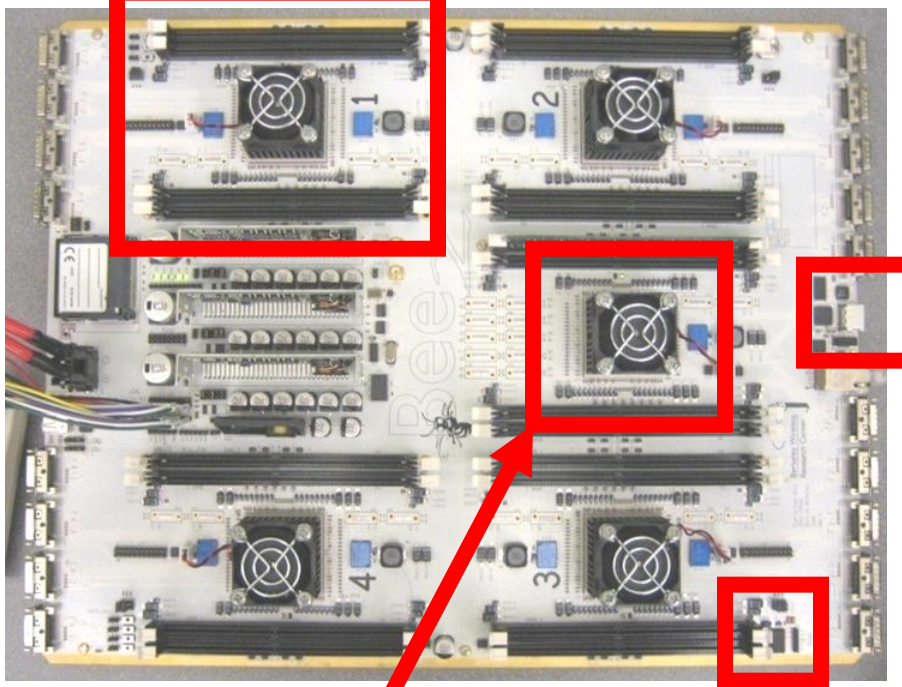
One more thing…

- **Our latest application w/ BlueSPARC: Trace CMP**



BlueSPARC Demo on BEE2

- **Demo application**
  - On-Line Transaction Processing benchmark (TPC-C) in Oracle
  - Runs in Solaris 8 (unmodified binary)
  - FPGA + Memory directly loaded from Simics checkpoint

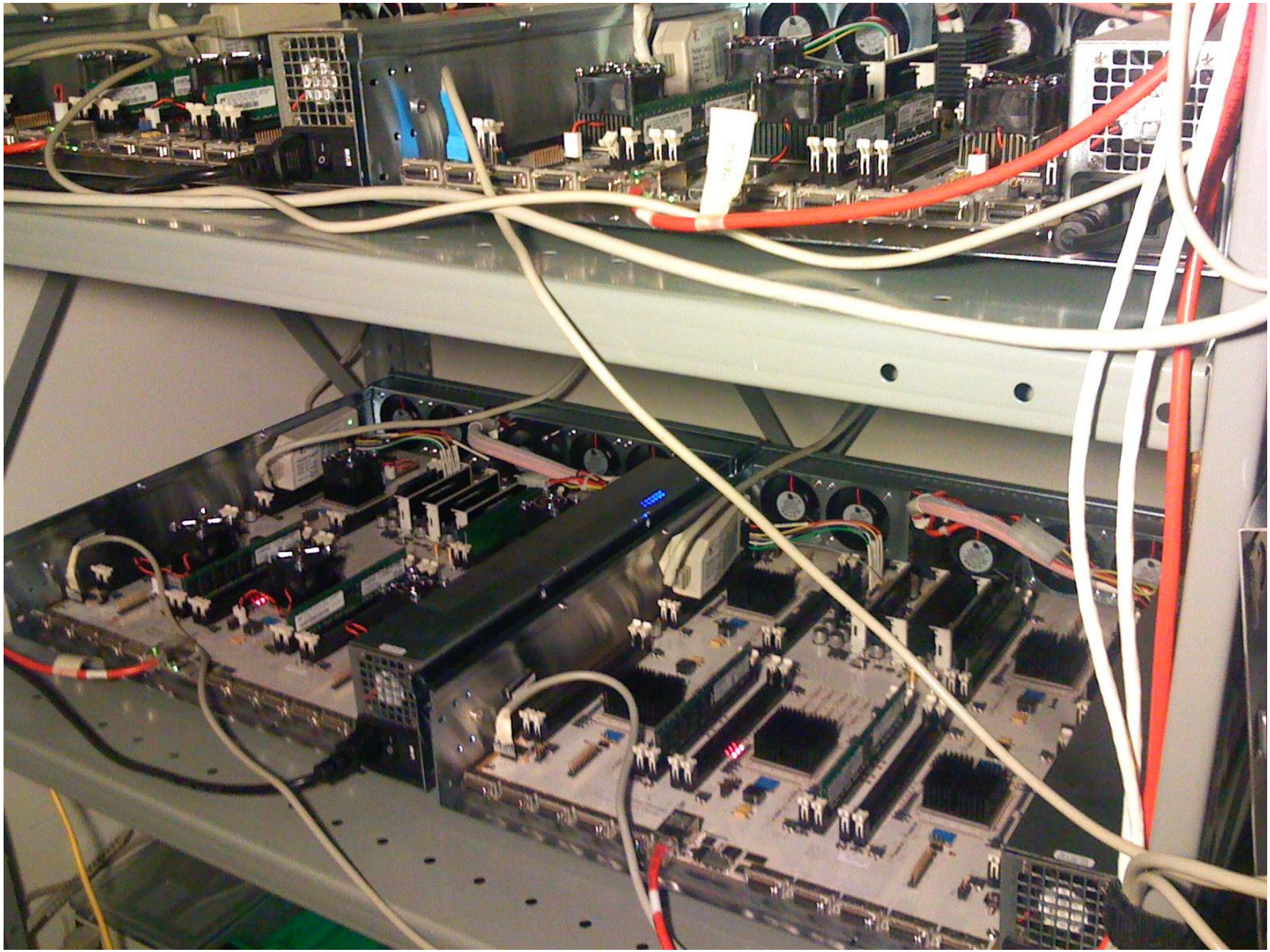


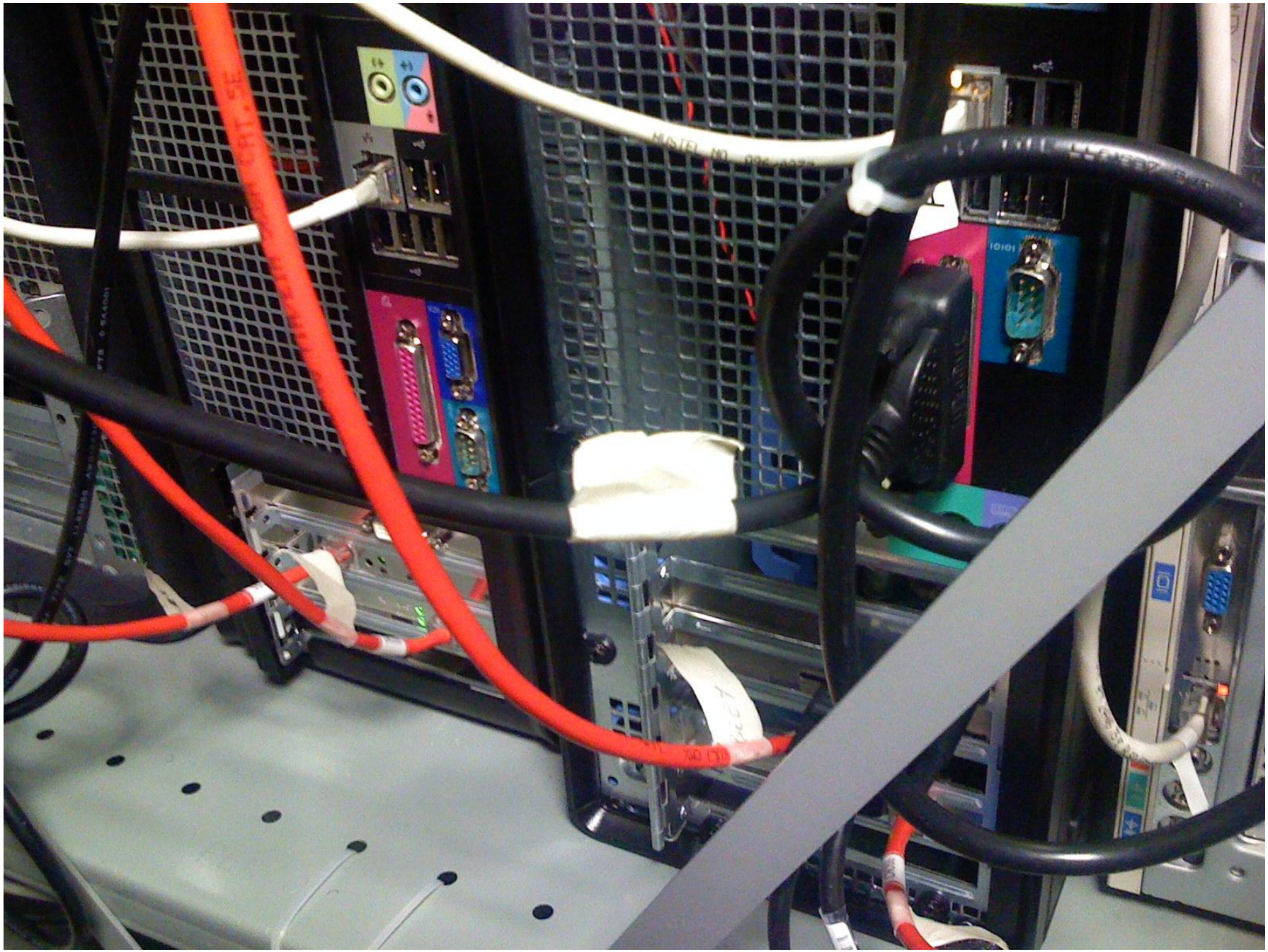
BEE2 Platform

Summary

- **Two techniques for reducing complexity**
  - Hybrid full-system simulation
  - MP host interleaving
- **Future work**
  - Timing extensions
  - Larger-scale implementation (hundreds of CPUs)
  - Run-time instrumentation tools

Part II: Hands-on Tutorial





Setup Procedures

- **Divide up into FOUR groups**
- **Each group needs 1 laptop with:**
  - Remote Desktop Connection for Windows XP
- **Instructions:**
  - http://www.ece.cmu.edu/~protoflex/wiki
  - user/login: ramp/ramp

Live CMP Cache Statistics

- **Statistics updated in real-time on webpage:**
  - http://www.ece.cmu.edu/~protoflex/asplos
- **Click on tabs to watch statistics for any of the four BEE2 boards as they run simulations**

Thanks! Any questions?

echung@ece.cmu.edu
http://www.ece.cmu.edu/~protoflex

Acknowledgements

We would like to thank our colleagues in the RAMP and TRUSS projects.