FACS

FPGA-Accelerated Multiprocessor Cache Simulator

Michael Papamichael, Wei Yu, Yongjun Jeon, Eric Chung,
James C. Hoe

papamix@cs.cmu.edu, {wy, yongjunj, echung, jhoe}@ece.cmu.edu

CALCM

PROTOFLEX

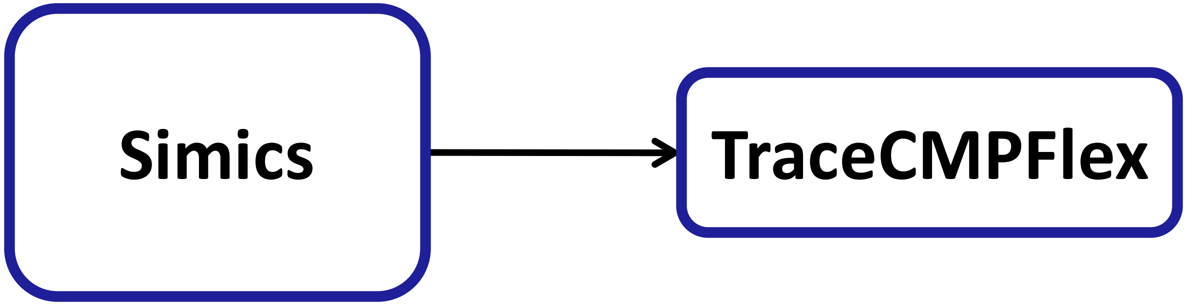
Computer Architecture Lab at
Carnegie Mellon

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

SUN Visit, Santa Clara, CA, January 18th 2008

Motivation

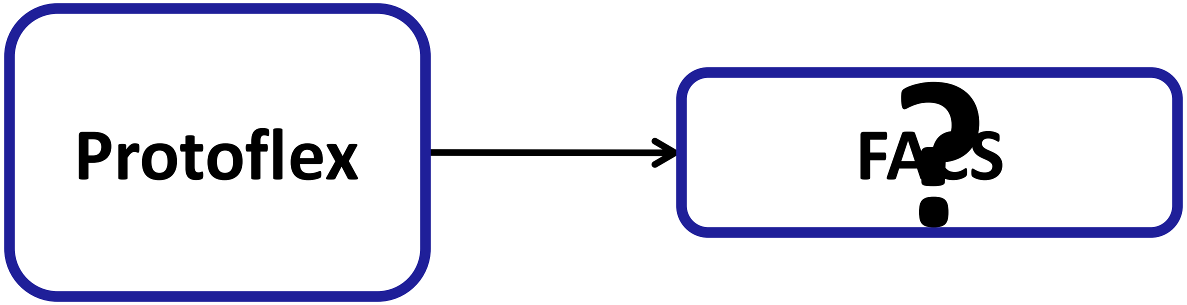
- Current SW-based simulation (e.g. Simics) slow



Simics

TraceCMPFlex

- Hardware FPGA-based simulation



Protoflex

FACS

?

FACS in a Nutshell

- **Functional HW Multiprocessor Cache Model**
  - Piranha-based 2-level Cache Coherence Design
  - Pipelined Implementation
  - Fully Parameterizable
- **Runs on BEE2 board @ 100 MHz**
  - High Throughput: 100 million references/sec
  - PowerPC Interface
  - Traces reside on DRAM or CF cards
- **Precise replication of SW Cache Model**
  - But **200x** faster!

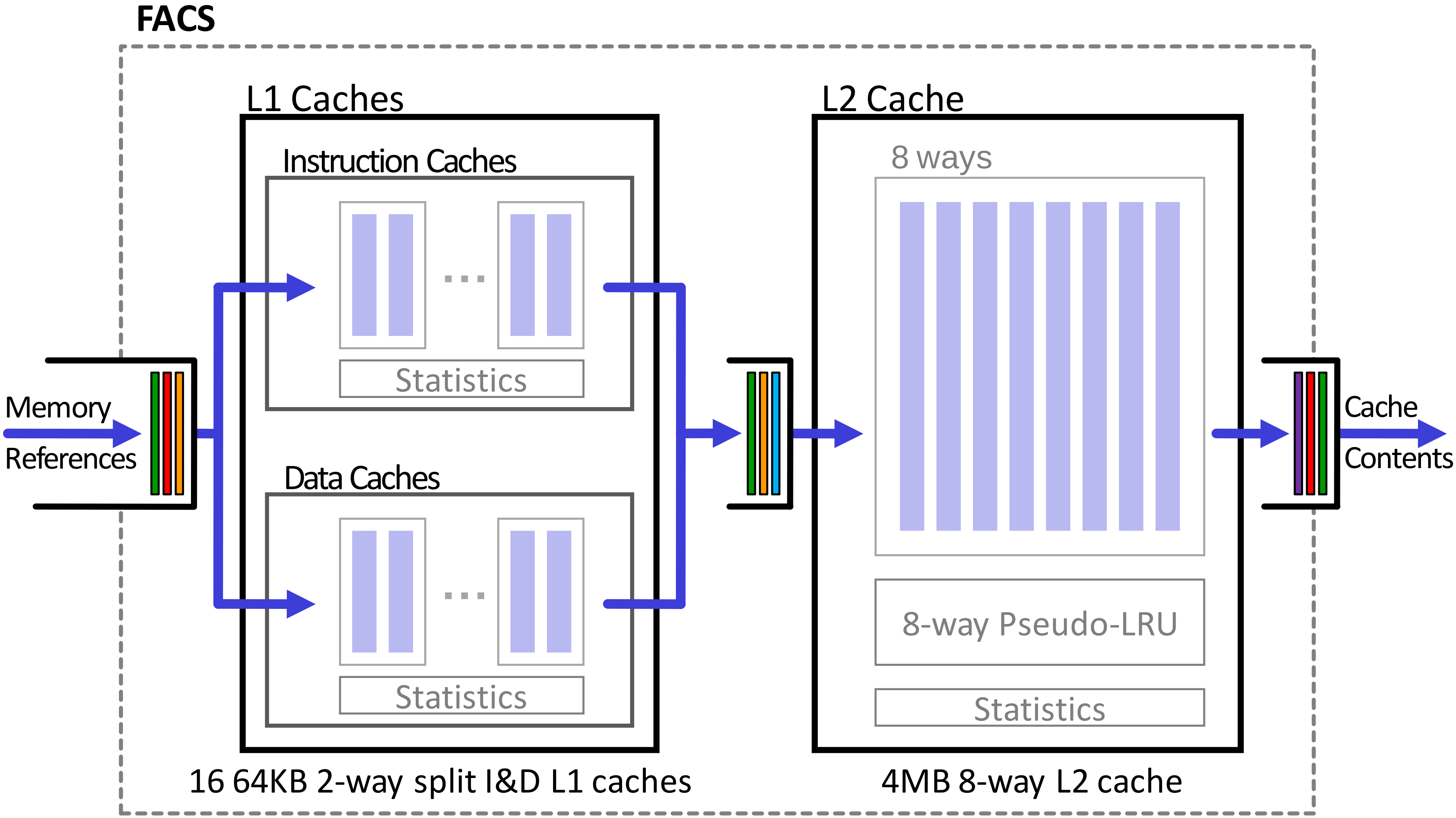
Implementation Details

- **2500L of Verilog code**
- **Functional Model**
  - Only tags + status bits stored and updated
- **L1 Caches**
  - Implemented as 2-stage pipeline
  - All 16 L1 I/D caches simultaneously accessed
- **L2 Cache**
  - Acts as large victim cache (evicted blocks from L1)
  - Each reference processed in 2 cycles (not pipelined)

Prototype Specs

- **FPGA:** Single Xilinx V2P70 (BEE2 board)
- **Clock Frequency:** 100 MHz
- **Configuration**
  - 16 nodes
  - private 64KB 2-way split I&D L1 caches
  - 4MB 8-way shared L2
- **FPGA Utilization**
  - LUTs: 7602 (11%)
  - BlockRAMs: 136 (41%)

FACS Architecture



Methodology & Results

- **Collected traces from real workloads & fed to:**
  - TraceCMPFlex (SW) [Intel Xeon 5130 @ 2GHz (4MB L2) with 8GB RAM]
  - FACS (HW) [BEE2 board @ 100MHz]
- **Correctness**
  - Matched cache contents and statistics for HW and SW
- **Performance Results**

	TraceCMPFlex (SW)	**FACS$_{worst}$**	**FACS$_{best}$**
Throughput	455K refs/sec	50M refs/sec	100M refs/sec

- **Speedup:** 110x – 220x

Lessons Learned & Future Work

- **Lessons Learned**
  - Simultaneous access to large # of L1s is hard to route
  - L1 cache size should be chosen to match well with the FPGA BlockRAM dimensions
  - Hard to develop synthesizable parameterizable modules with Verilog
- **Future Work**
  - Experiment with larger cache configurations
  - Larger cache sizes through virtualization of tag arrays
  - Timing?

Thank you!