ATLAS
A Scalable Emulator for Transactional Parallel Systems

Christos Kozyrakis and Kunle Olukotun

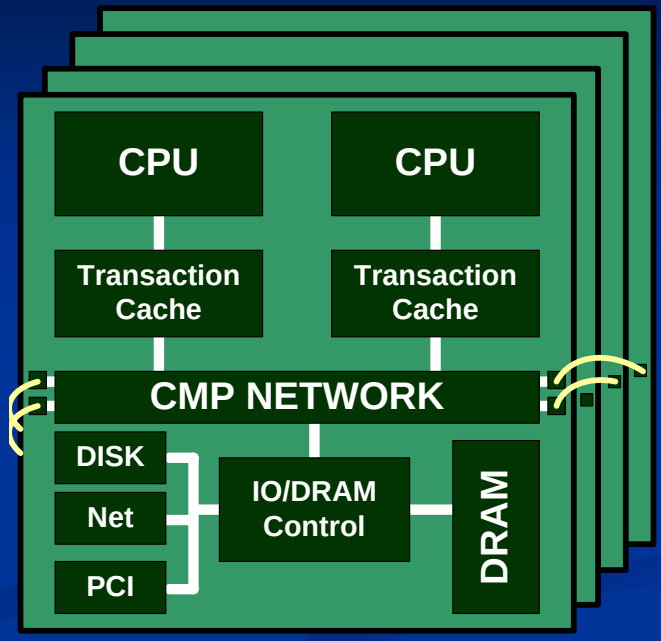
Computer Systems Laboratory
Stanford University
http://tcc.stanford.edu

Motivation

- CMPs are here, but how do we program them?
- Our proposal: transactional programming & execution
  - Programs written as sequences of transactions
  - CMP executes transactions in parallel with optimistic concurrency
  - More details at http://tcc.stanford.edu
- Challenges
  - Explore programming model with large applications & datasets
  - Interactions with operating systems and IO
  - Large-scale transactional architectures (>16 nodes)
- Need a fast, scalable emulator for system-level studies
  - Full-system simulation too slow for our purposes…

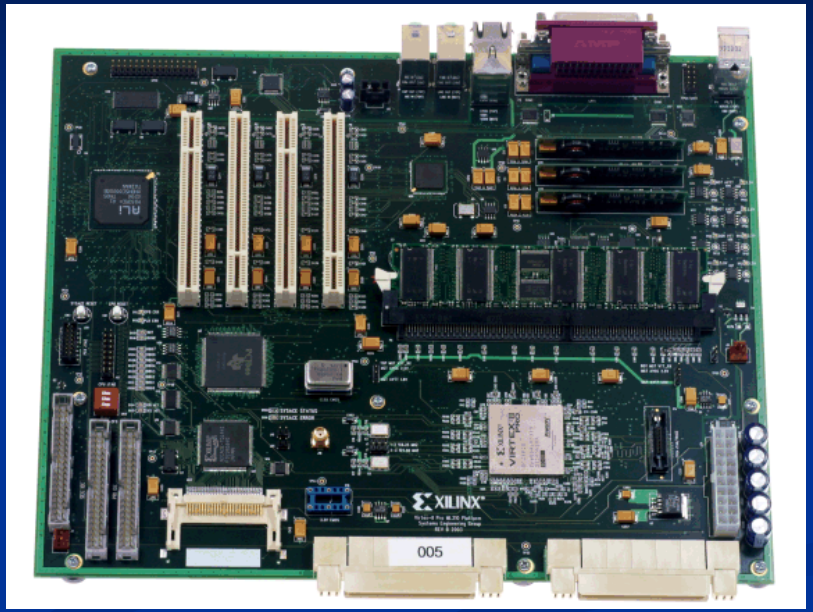
ATLAS Overview

- A multi-board emulator for transactional parallel systems
- Goals
  - 16 to 64 CPUs (8 to 32 boards)
  - 50 to 100MHz
  - Stand-alone full-feature system
    - OS, IDE disks, 100Mb Ethernet, …
- ATLAS architecture space
  - Small, medium, and large-scale CMPs and SMPs
  - UMA and NUMA
  - Flexible transactional memory hierarchy & protocol
  - Flexible network model
  - Flexible clocking, latency, and bandwidth settings

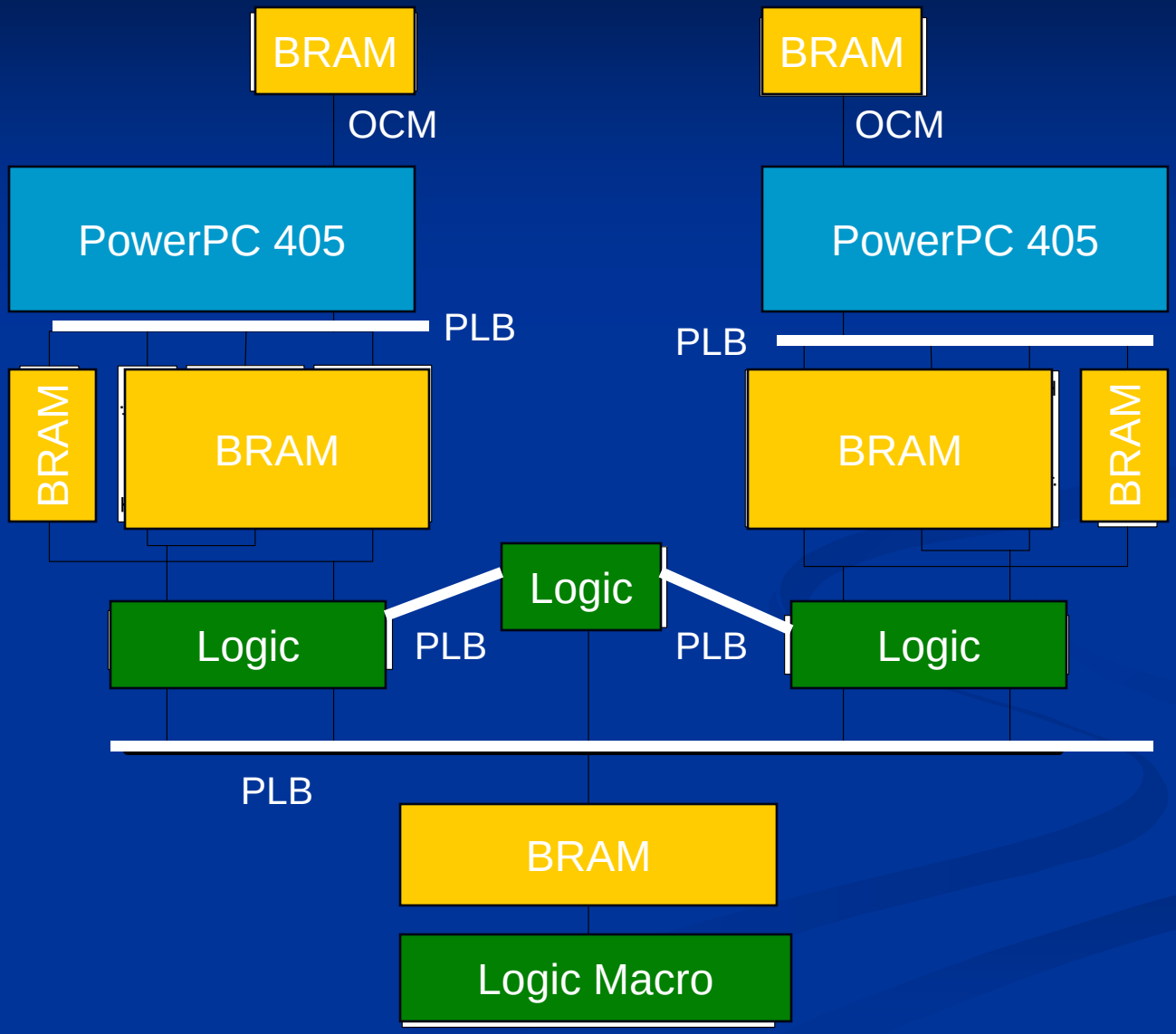


Building Block: Xilinx ML310 Board

- XC2VP30 FPGA features
  - 2 PowerPC 405 cores
  - 2.4Mb dual-ported SRAM
  - 30K logic cells
  - 8 RocketIO 3.125Gbps transceivers
- System features
  - 256MB DDR, 512MB CompactFlash
  - Ethernet, PCI, USB, IDE, …
- Design and development tools
  - Foundation ISE for design entry, synthesis, …
    - For the transactional memory hierarchy and network
  - Chipscope Pro logic analyzer for debugging
  - EDK for system simulation, system SW development, configuration, …
  - Montavista Linux 3.1 Pro



Example: 2-way bus-based transactional CMP



ATLAS Software Framework

- PowerPC and ML310 features provide rich SW framework
  - Linux OS
    - Port for Xilinx boards available from Montavista
    - Allows exploration of transactions with IO and scheduling
  - Gcc C/C++ software framework
    - TCC API for transactional programming
    - Allows experimentation with wide range of applications
  - Jikes-RVM Java framework
    - TCC API for transactional programming
    - Allows exploration of dynamic optimization techniques
- Allows us to focus on parallel programming quickly
  - No need to develop significant infrastructure from scratch
  - Gradual path to parallel application development
    - Sequential version of C/C++/Java apps runs immediately

Trade-offs & Scalability

- ATLAS trade-offs
  - – Sacrifice some hardware modeling flexibility
    - Simple CPU, SW or coprocessor FPU, bounded on-chip memory
  - \+ Fast hardware prototyping
    - Develop RTL for transactional memory + networking protocol
  - \+ Rich software framework
  - \+ Based on commercial hardware and software
    - Low cost, timely upgrades and improvements
- Scaling
  - Scalability by adding boards (size & performance)
    - Use RocketIO tranceivers and Xilinx Aurora protocol
  - Limitations
    - 32-bit cores can address up to 4GB of shared memory
    - 8 transceivers per chip $\Rightarrow$ must synthesize router for >16 CPU

Summary

- A scalable emulator for transactional parallel systems
  - Based on commercial FPGA chips, boards, and software
  - 32 to 64 CPUs at 50 to 100MHz
    - A 6.4 GIPS emulator at full scale
  - Low cost, fast, flexible
- ATLAS architecture space
  - Large-scale parallel systems with transactional memory support
- ATLAS software space
  - Transactional parallel programming and optimizations
  - Operating systems and IO research
  - Large-scale application development
    - Embedded, server, desktop