Manifold: A Parallel Simulation Framework for Multicore Systems

Jun Wang, Jesse Beu, Rishiraj Bheda, Tom Conte, Zhenjiang Dong, Chad Kersey, Mitchelle Rasquinha, George Riley, William Song, He Xiao, Peng Xu, and Sudhakar Yalamanchili

School of Electrical and Computer Engineering and School of Computer Science
Georgia Institute of Technology
Atlanta, GA. 30332

ISPASS 2014

Sponsors

NSF · Sandia National Laboratories · LABS hp · Oracle Labs · SRC Semiconductor Research Corporation

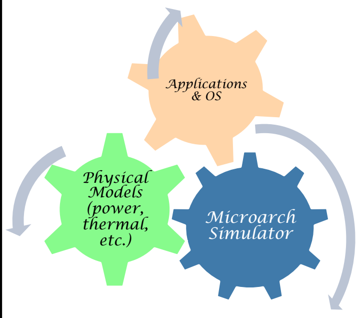
Motivation

“Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.”

Box, G. E. P., and Draper, N. R., (1987), *Empirical Model Building and Response Surfaces*, John Wiley & Sons, New York, NY.

George E. P. Box, 2011

Simulation Infrastructure Challenges



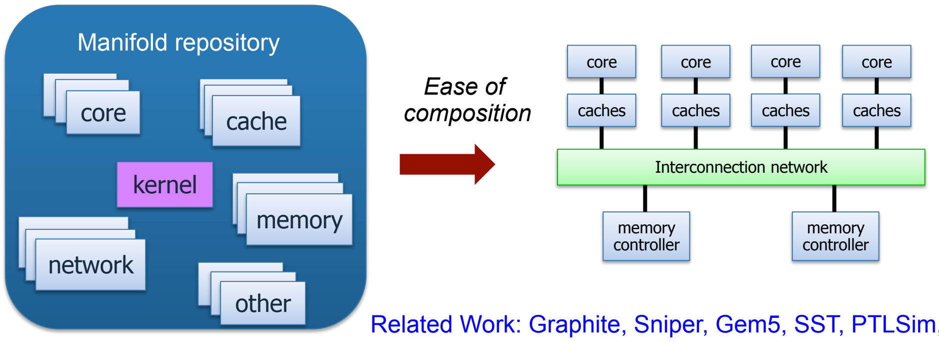
- Scalability
  - Processors are parallel and tools are not → not sustainable
- Multi-disciplinary
  - Functional + Timing + Physical models
- Need to model complete systems
  - Cores, networks, memories, software at scale
- Islands of expertise
  - Ability to integrate point tools → best of breed models
- Composability
  - Easily construct the simulator you need

Overview

- Execution Model ⬅
- Multicore Emulator Front-End & Component Based Timing Model Back-End
- Physical Modeling
- Parallel Simulation
- Some Example Simulators

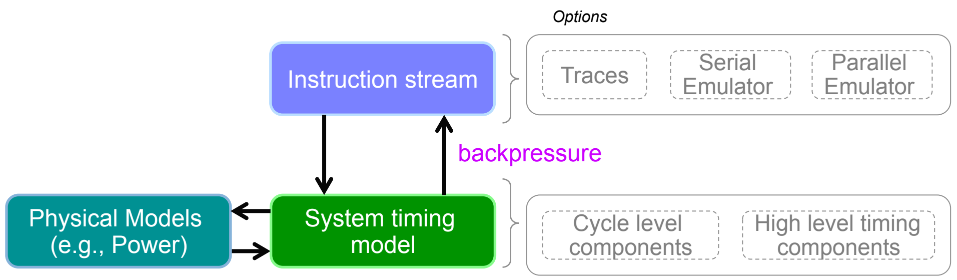
Manifold Overview

- A parallel simulation framework for multicore architectures
- Consists of:
  - A parallel simulation kernel
  - A (growing) set of architectural components
  - Integration of physical models for energy, thermal, power, and so on
- Goal: easy construction of parallel simulators of multicore architectures



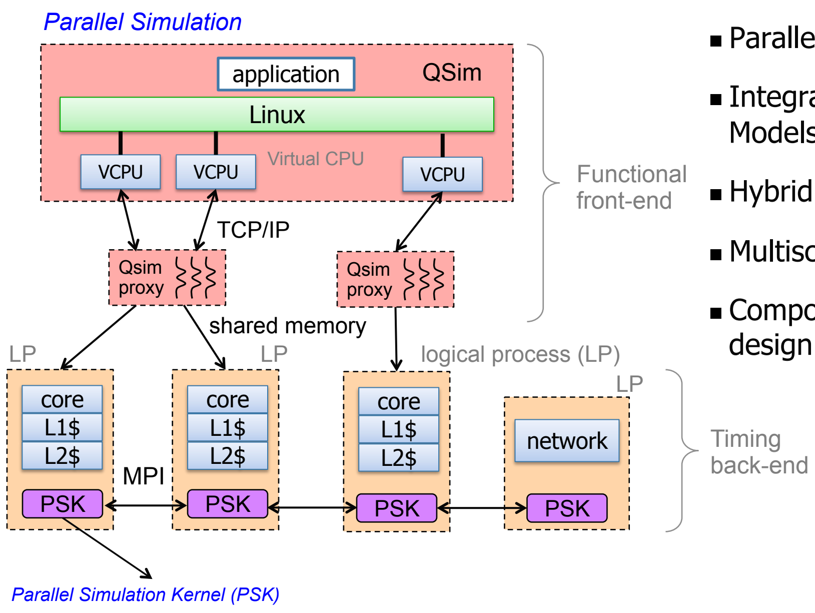
Related Work: Graphite, Sniper, Gem5, SST, PTLSim, etc.

Execution Model: Overview



- Instruction stream
  - Generated by i) trace files, ii) Qsim server, iii) Qsim Lib
- System timing model
  - Multicore model built with Manifold components
  - Components assigned to multiple logical processes (LPs)
    - Each LP assigned to one MPI task; LPs run in parallel

Execution Model (Socket/Blade)

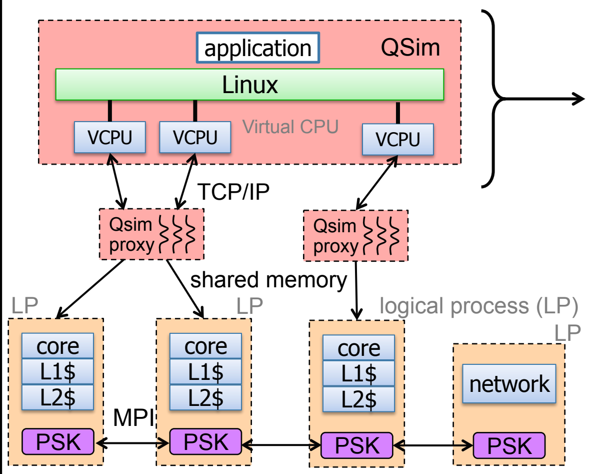


- Full-system simulation
- Parallel Simulation
- Integrated Physical Models
- Hybrid timing model
- Multiscale
- Component-based design

Overview

- Execution Model
- Multicore Emulator Front-End & Component Based Timing Model Back-End ⬅
- Physical Modeling
- Parallel Simulation
- Some Example Simulators

QSim Multicore Emulator



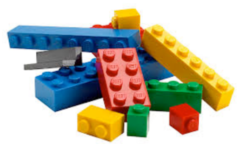
- Runs unmodified x86 (32-bit) binaries on lightly-modified Linux kernel.
- Provides callback interface for execution events
- Callbacks generated for all instructions, including OS
- Filtering of instruction stream
- Can be extended to support other ISAs, e.g., ARM, PPC, via QEMU support

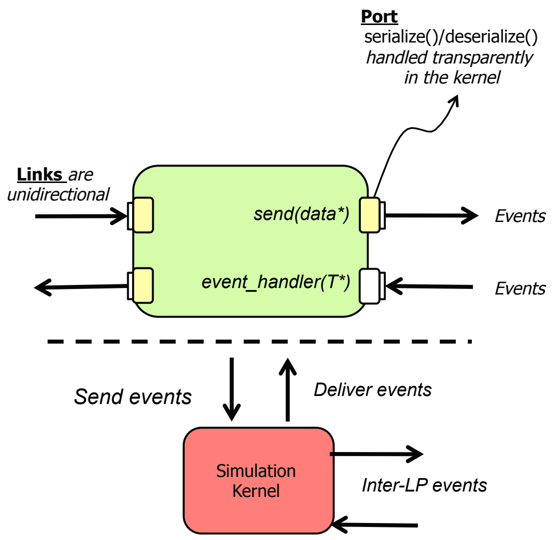
- Library for instantiating multicore emulators
- Based on the core translation engine from QEMU

QSim Features

- Fine-grained Instruction-level execution control
  - Single instruction level
- Two ways to use:
  - QsimLib: for creating multithreaded emulators
  - QSim Server: for serving parallel/distributed simulations
- State files for fast startup
  - State file contains memory state after Linux bootup
- Fast forward and region of interest support
- Booted up to 512 virtual cores

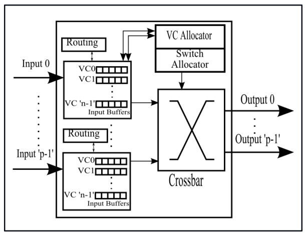
Timing Model Components





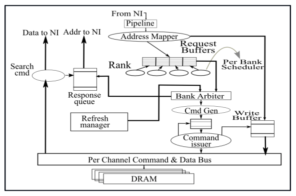
- Component may be time-stepped, discrete event, or both
  - Named handlers for each case
  - Clock subscription
  - Can mix time stepped and DES
- Kernels enforce correct ordering across and within LPs

Manifold Component Models



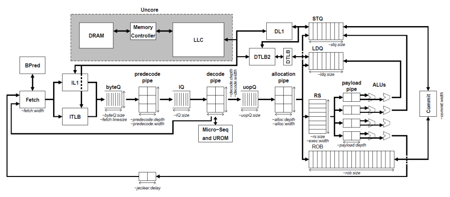
IRIS: Flit level Network Simulator

- *Multiple Models*
- *Multiple level of Abstraction*



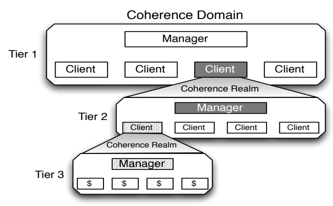
CaffDRAM

Multiple core models: in-order, out-of-order, abstract, etc.



G. Loh et. al Zesto: A Cycle-Level Simulator for Highly Detailed Microarchitecture Exploration, ISPASS 2009

Coherent cache hierarchy



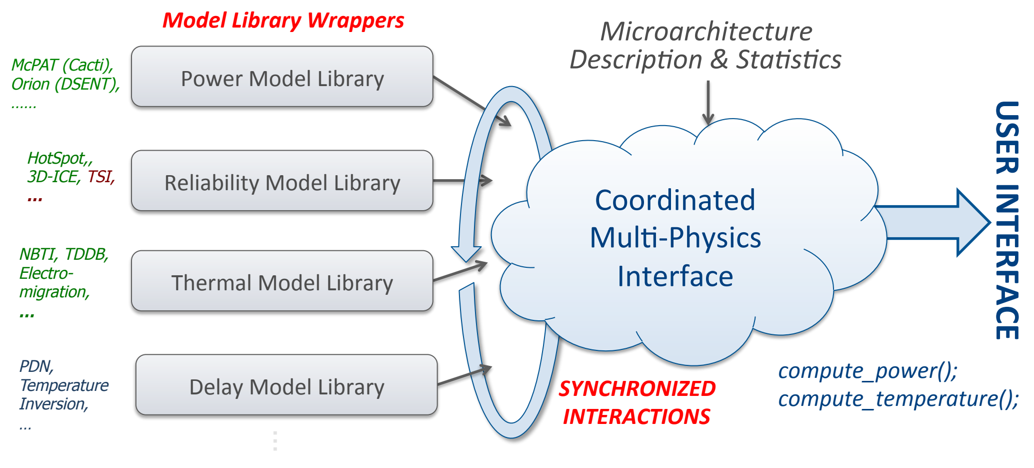
*J. G. Beu,, "Manager-Client Pairing: A Framework for Implementing Coherence Hierarchies," MICRO-44, Dec., 2011.

Overview

- Execution Model
- Multicore Emulator Front-End & Component Based Timing Model Back-End
- Physical Modeling
- Parallel Simulation
- Some Example Simulators

Modeling Physical Phenomena

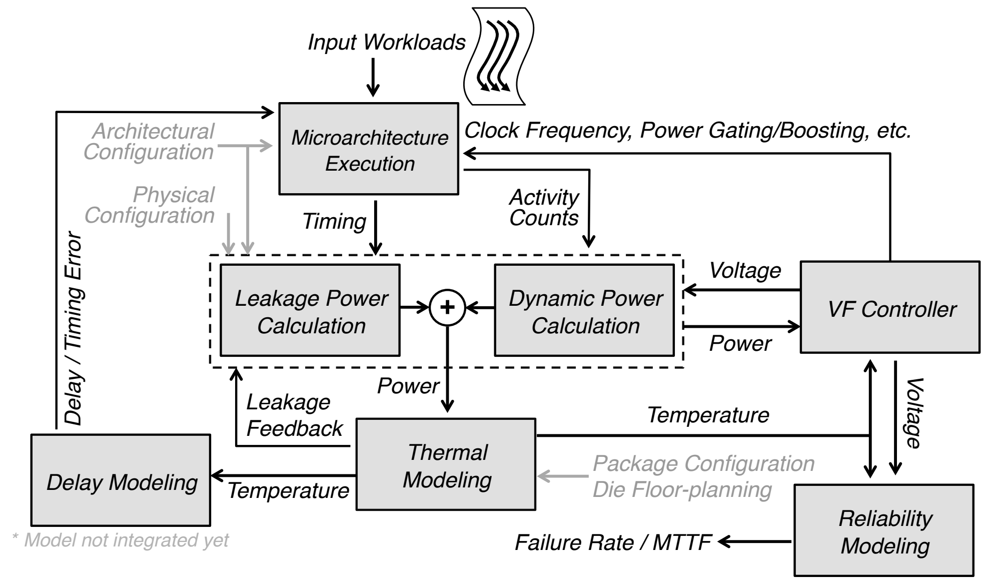
- *Energy Introspector* (**EI**) is a modeling library to facilitate the (selective) uses of different models and capture the interactions among microprocessor physics models.


Model Library Wrappers
McPAT (Cacti), Orion (DSENT), ...
Power Model Library
HotSpot,, 3D-ICE, TSI, ...
Reliability Model Library
NBTI, TDDB, Electro-migration, ...
Thermal Model Library
PDN, Temperature Inversion, ...
Delay Model Library
Microarchitecture Description & Statistics
Coordinated Multi-Physics Interface
SYNCHRONIZED INTERACTIONS
USER INTERFACE
compute_power();
compute_temperature();

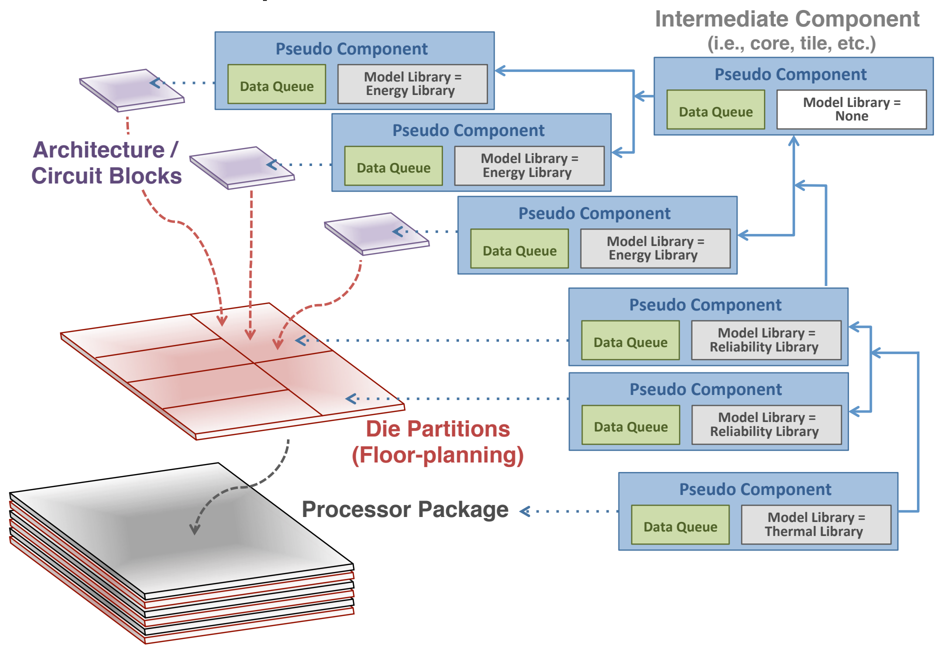
Available at *www.manifold.gatech.edu*

Architecture-Level Physical Modeling

Abstract Representation of Architecture-Physics Interactions



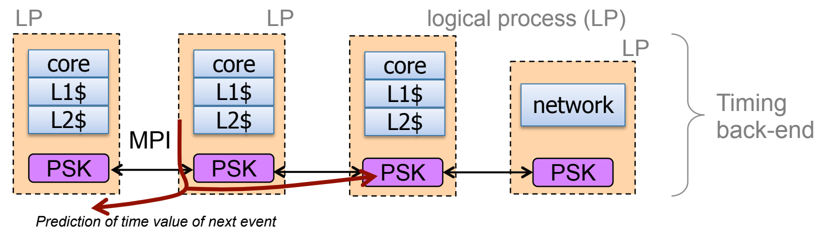
Processor Representation



Overview

- Execution Model
- Multicore Emulator Front-End & Component Based Timing Model Back-End
- Physical Modeling
- Parallel Simulation
- Some Example Simulators

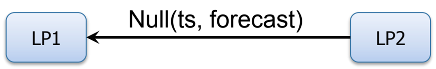
Parallel Simulation Kernel: Synchronization Algorithms



- Problem: Synchronized advance of simulation time
- State of the Practice solutions
  - Conservative algorithms, e.g., lower bound time stamp (PBTS)
  - Optimistic algorithms, e.g., Time Warp
  - Included in Manifold
- New: Forecast Null Message (FNM)
  - Use domain specific information to predict time of future events
  - For example, consequence of Last Level Cache access.

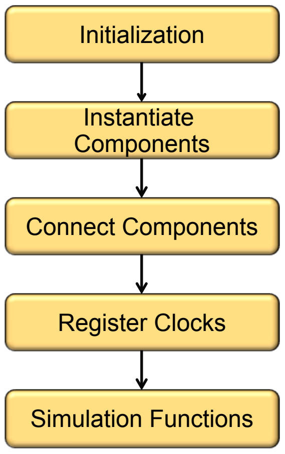
Forecast Null Message (FNM) Algorithm

1. Forecast determined by runtime state
   - E.g., at cycle t, cache receives a request; with latency L, the earliest time when it sends out a message is (t + L). This is its forecast.
2. Because components send credits after receiving events, time-stamp for Null-message must consider neighbors' forecast.



3. Null-message time-stamp set to `min(out_forecast, in_forecast)`, where `in_forecast` is forecast carried on incoming Null-messages.

Building and Running Parallel Simulations



Configuration parameters

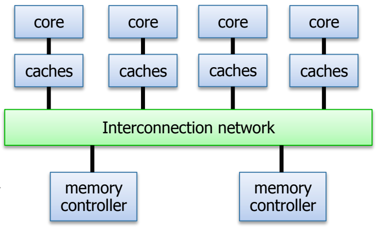
- From Manifold Library
- Inputs (trace, QSIM, etc.)

Instantiate Links

- Set Timing Behavior
- Time stepped vs. discrete event

Set Duration, Cleanup, etc.

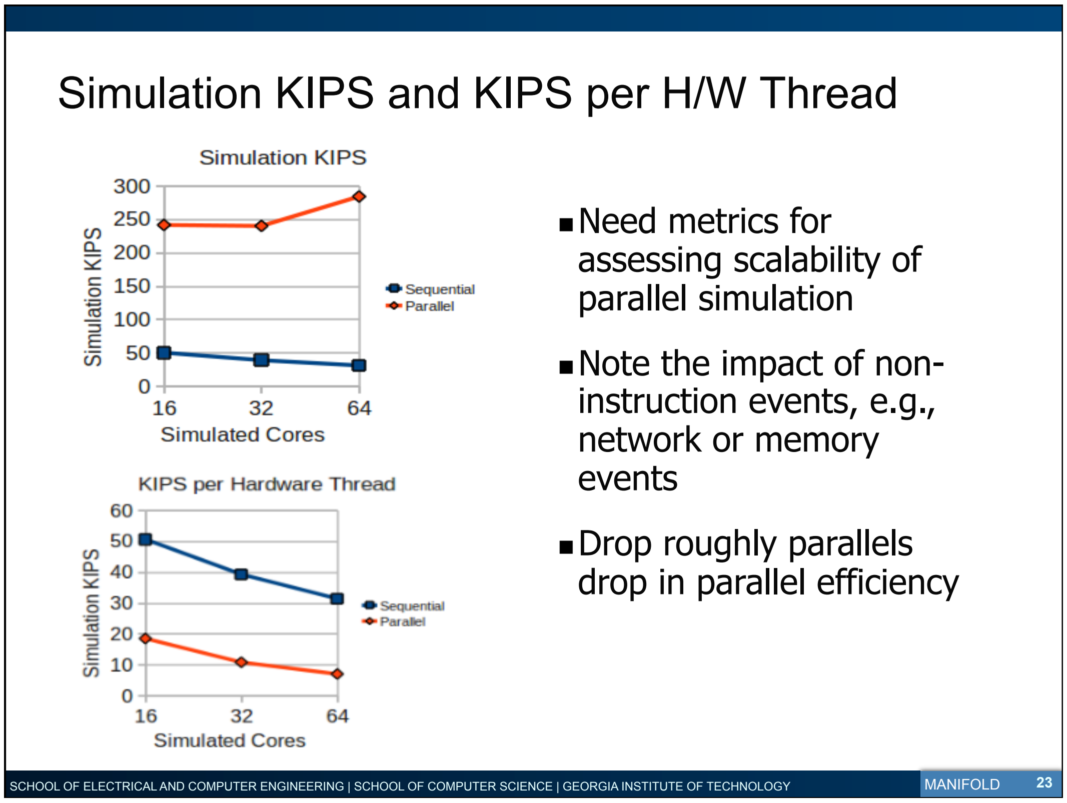
CMP (16-, 32-, 64-core)



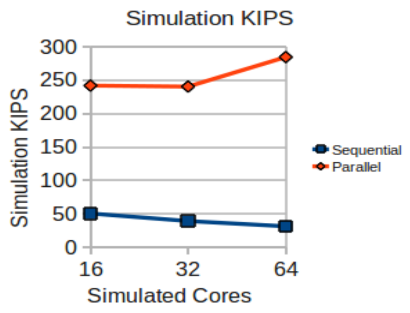
- 16, 32, 64-core CMP models
- 2, 4, 8 memory controllers, respectively
- 5x4, 6x6, 9x8 torus, respectively
- Host: Linux cluster; each node has 2 Intel Xeon X5670 6-core CPUs with 24 h/w threads
- 13, 22, 40 h/w threads used by the simulator on 1, 2, 3 nodes, respectively
- 200 Million simulated cycles in region of interest (ROI)
  - Saved boot state and fast forward to ROI

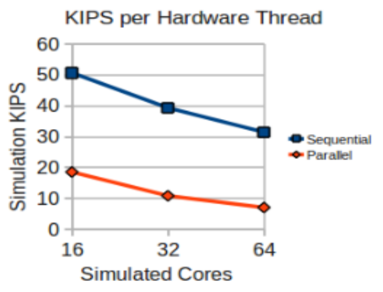
Sample Results: Simulation Time in Minutes

	16-core		32-core		64-core	
	Seq.	Para.	Seq.	Para.	Seq.	Para.
dedup	1095.7	251.4 (4.4X)	2134.8	301.3 (7.1X)	2322.9	345.3 (6.7X)
facesim	1259.3	234.9 (5.4X)	2614.2	303.6 (8.6X)	3170.2	342.3 (9.3X)
ferret	1124.8	227.8 (4.9X)	1777.9	255.6 (7.0X)	2534.3	331.3 (7.6X)
freqmine	1203.3	218.0 (5.5X)	1635.6	245.6 (6.7X)	2718.9	337.3 (8.1X)
stream	1183.8	222.7 (5.3X)	1710.6	244.3 (7.0X)	4796.4	396.2 (12.1X)
vips	1167.0	227.3 (5.1X)	1716.3	257.2 (6.7X)	2564.6	337.9 (7.6X)
barnes	1039.9	224.3 (4.6X)	1693.0	283.3 (6.0X)	3791.8	341.4 (11.1X)
cholesky	1182.4	227.2 (5.2X)	1600.3	245.7 (6.5X)	4278.3	402.1 (10.6X)
fmm	1146.3	229.6 (5.0X)	1689.8	253.6 (6.7X)	5037.2	416.1 (12.1X)
lu	871.2	156.4 (5.6X)	1475.8	204.6 (7.2X)	4540.3	402.7 (11.3X)
radiosity	1022.3	228.8 (4.5X)	1567.5	250.4 (6.3X)	2813.5	350.3 (8.0X)
water	671.5	158.4 (4.2X)	1397.3	236.7 (5.9X)	2560.1	356.3 (7.2X)

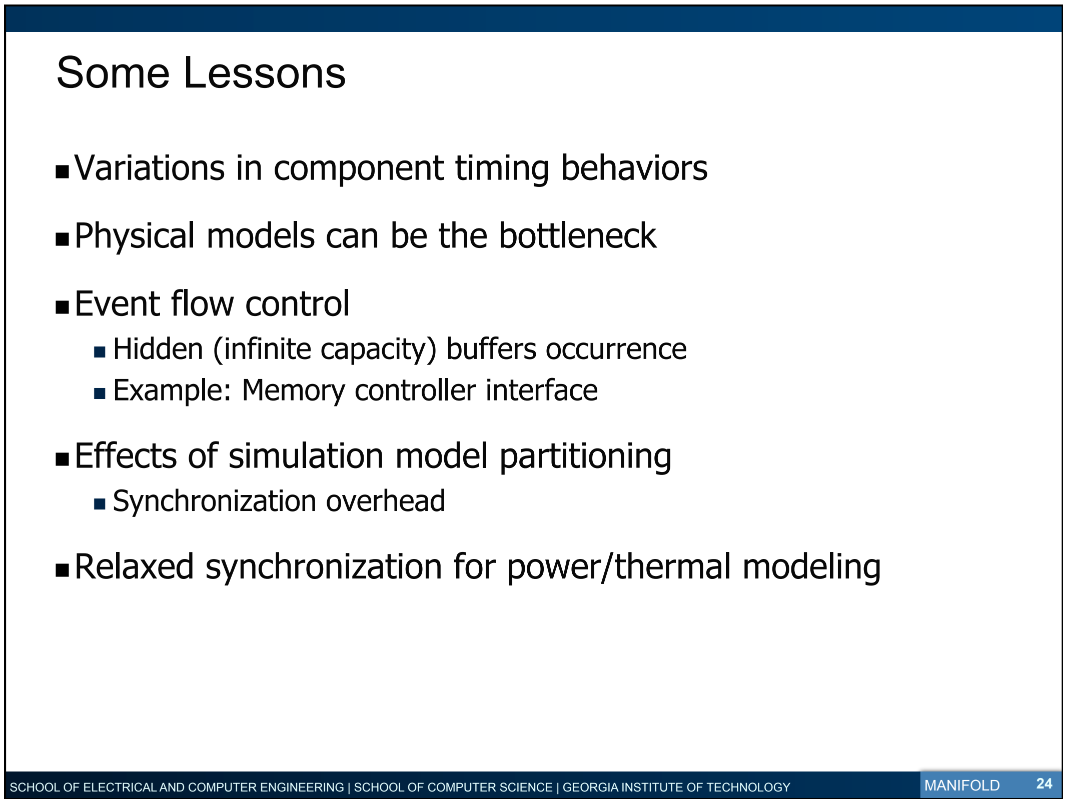


Simulation KIPS and KIPS per H/W Thread





- Need metrics for assessing scalability of parallel simulation
- Note the impact of non-instruction events, e.g., network or memory events
- Drop roughly parallels drop in parallel efficiency



Some Lessons

- Variations in component timing behaviors
- Physical models can be the bottleneck
- Event flow control
  - Hidden (infinite capacity) buffers occurrence
  - Example: Memory controller interface
- Effects of simulation model partitioning
  - Synchronization overhead
- Relaxed synchronization for power/thermal modeling

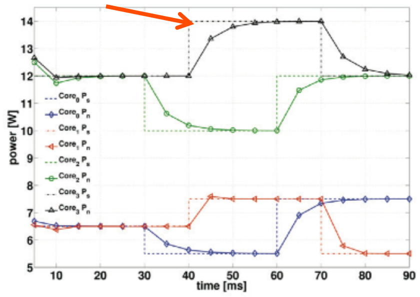
Overview

- ▪Execution Model
- ▪Multicore Emulator Front-End & Component Based Timing Model Back-End
- ▪Physical Modeling
- ▪Parallel Simulation
- ▪Some Example Simulators

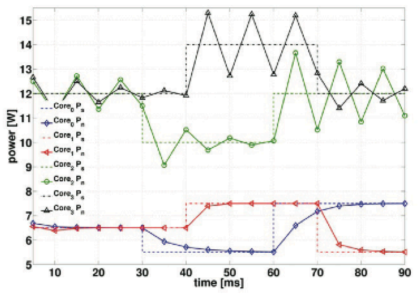
Power Capping Controller

New set point

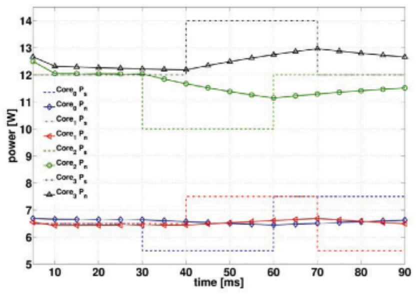
Two in-order and two OOO cores



(a) Adaptive gain controller



(b) High fixed gain controller (K = $500e^6$)

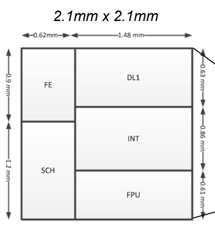


(c) Low fixed gain controller (K = $25e^6$)

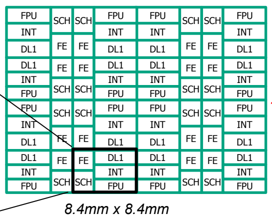
- Dynamic Voltage Frequency Scaling
- Regulating asymmetric processors

N. Almoosa, W. Song, Y. Wardi, and S. Yalamanchili, "A Power Capping Controller for Multicore Processors," American Control Conf., June 2012.

Microfluidic Cooling in Die Stacks

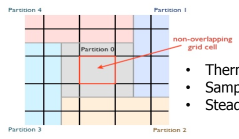


Nehalem-like, OoO cores;
3GHz, 1.0V, max temp 100°C
DL1: 128KB, 4096 sets, 64B
IL1: 32KB, 256 sets, 32B, 4 cycles;

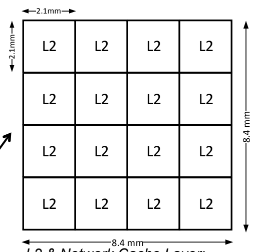


8.4mm x 8.4mm

16 symmetric cores

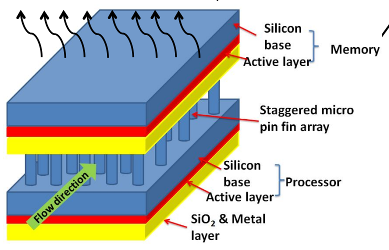


- Thermal Grids: 50x50
- Sampling Period: 1us
- Steady-State Analysis



L2 & Network Cache Layer:
L2 (per core): 2MB, 4096 sets, 128B, 35 cycles;
DRAM: 1GB, 50ns access time (for performance model)

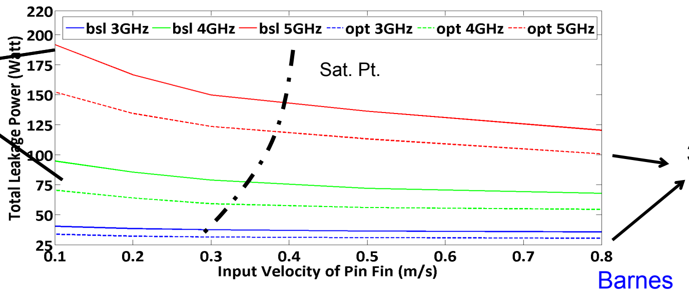
Ambient:
Temperature: 300K



Executing SPLASH and PARSEC Benchmarks

H. Xiao, Z. Min, S. Yalamanchili and Y. Joshi, "Leakage Power Characterization and Minimization over 3D Stacked Multi-core Chip with Microfluidic Cooling," *IEEE Symposium on Thermal Measurement, Modeling, and Management (SEMITHERM)*, March 2014

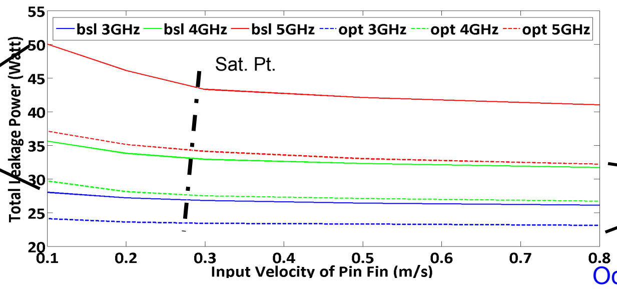
Impact of Cooling Co-design on Leakage Power



4.75X

3.31X

Barnes



1.79X

1.39X

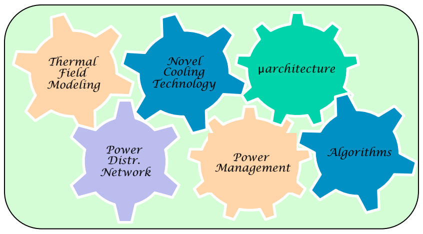
Ocean-c

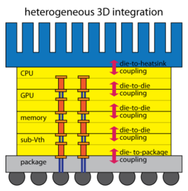
H. Xiao, Z. Min, S. Yalamanchili and Y. Joshi, "Leakage Power Characterization and Minimization over 3D Stacked Multi-core Chip with Microfluidic Cooling," *IEEE Symposium on Thermal Measurement, Modeling, and Management (SEMITHERM)*, March 2014

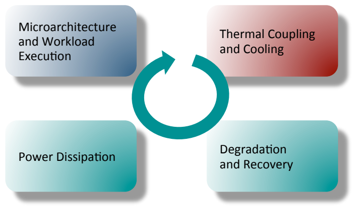
Summary

www.manifold.gatech.edu

- Composable simulation infrastructure for constructing multicore simulators
  - Parallel execution
  - Integrated physical models
- Provide base library of components to build useful simulators
- Distribute some stock simulators
- Need: Validation Techniques such as Uncertainty Quantification






Microarchitecture and Workload Execution
Thermal Coupling and Cooling
Power Dissipation
Degradation and Recovery