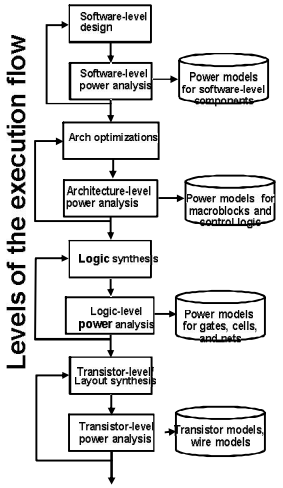
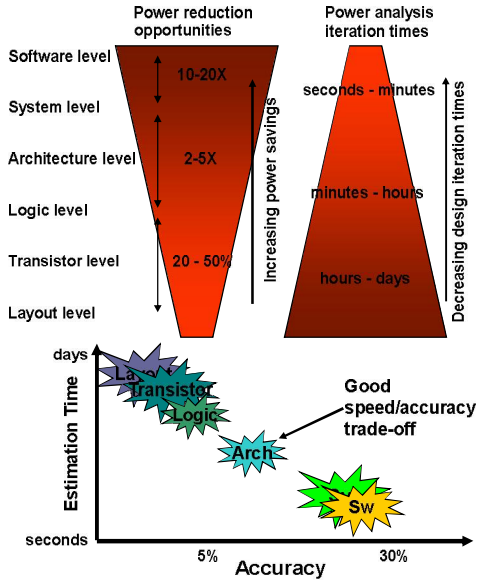
Power and performance modelling

Ramon Canal

How Power Estimation is Addressed





Agenda

- Building-up a model: Orion
- Power estimation through Hardware Counters -Based on PARAPET group work at Princeton
- Architecture Simulators
- Conclusions
- Related Work

Orion: Interconnect model/simulator

Ramon Canal

Orion

- Introduction
- Dynamic Network Simulator
- Power Modeling
- Case Studies
- Related Work
- Conclusions

Orion

- Introduction
- Dynamic Network Simulator
- Power Modeling
- Case Studies
- Related Work
- Conclusions

Introduction

- Interconnection networks are becoming an import part of micro-processing
- System power consumption is increasingly becoming at least equally as important as performance
- Interconnect networks are consuming an ever greater percentage of system power

Introduction (cont.)

- Interconnection networks are seen as only scalable solution to inter-processor comm by single-chip multiprocessors
- Soon, routers and links will be critical components of a microprocessor system
- InfiniBand switch is estimated to dissipate almost 37.5% of the blade's power budget in a Mellanox server blade

Introduction (cont.)

- Simulator is constructed within the Liberty Simulation Environment (LSE)
- Goal: provide a complete platform for exploring interconnected microprocessors, whether single-chip or spanning multiple chips, at the architectural level
- Power model for interconnects usable from other tools

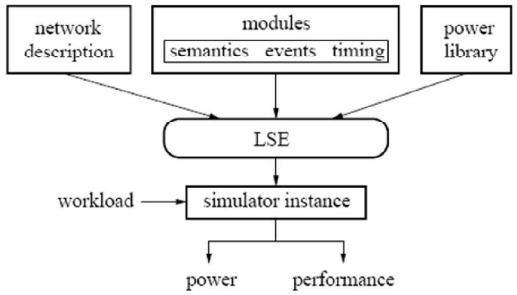
Orion

- Introduction
- Dynamic Network Simulator
- Power Modeling
- Case Studies
- Related Work
- Conclusions

Simulator Infrastructure

- Adopted LSE as basic simulator infrastructure
- LSE targets fast design space exploration for modern microprocessors
- LSE models physical hardware blocks as logical functional models which communicate through ports
- Integration of power models is based on the event subsystem of LSE that facilitates collection of execution stats
- Power models in power simulation are hooked to these events
- Specific power model calculates and accumulates the energy which is consumed

Process of building a simulator in LSE



Building blocks of interconnection network

- Interconnection networks can be decomposed into component modules
- Modules are parameterized so that they can be reused
- A relatively small library of modules is able to represent an extensive range of architecture choices

Orion

- Introduction
- Dynamic Network Simulator
- Power Modeling
- Case Studies
- Related Work
- Conclusions

Power Modeling

- Architectural-level parameterized power models are derived for several major building blocks (FIFO buffers, crossbars and arbiters)

Component Power Modeling

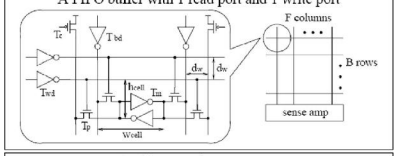
- For each component, first the canonical structure is described in terms of architectural and technological parameters
- The detailed analysis is performed to determine parameterized capacitance equations
- Capacitance equations and switch activity estimation are combined to determine energy consumption/component operation

Power Model for FIFO buffer

Table 1. Terminology

$C_g(T)$	gate capacitance of transistor/gate T
$C_d(T)$	diffusion capacitance of transistor/gate T
$C_a(T)$	$C_g(T) + C_d(T)$
$C_w(L)$	capacitance of metal wire of length L
E_x	$\frac{1}{2}C_x V_{dd}^2$ or $C_x V_{dd}^2$ depending on how to count switches, provided C_x is defined

Table 2. Model for FIFO buffers

Canonical structure	
A FIFO buffer with 1 read port and 1 write port	
	
Architectural parameters	
B	buffer size in flits
F	flit size in bits
P_r	number of buffer read ports
P_w	number of buffer write ports
Technological parameters	
h_{cell}	memory cell height
w_{cell}	memory cell width
d_w	wire spacing

Capacitance equations	
wordline length	$L_{wl} = F(w_{cell}+2(P_r+P_w)d_w)$
bitline length	$L_{bl} = B(h_{cell}+(P_r+P_w)d_w)$
wordline cap.	$C_{wl} = 2FC_g(T_p)+C_a(T_{wd})+C_w(L_{wl})^\star$
read bitline cap.	$C_{br} = BC_d(T_p) + C_d(T_c) + C_w(L_{bl})$
write bitline cap.	$C_{bw} = BC_d(T_p) + C_a(T_{bd}) + C_w(L_{bl})$
precharge cap.	$C_{chg} = C_g(T_c)$
memory cell cap.	$C_{cell} = 2(P_r + P_w)C_d(T_p) + 2C_a(T_m)$
sense amp energy	E_{amp} from empirical model [28]

Operation energy equations	
δ_{bw}	number of switching write bitlines
δ_{bc}	number of switching memory cells
read energy	$E_{read} = E_{wl} + F(E_{br} + 2E_{chg} + E_{amp})$
write energy	$E_{wrt} = E_{wl} + \delta_{bw}E_{bw} + \delta_{bc}E_{cell}$

($\star$) T_p is the pass transistor connecting bitlines and memory cells, T_{wd} is the wordline driver, T_{bd} is the write bitline driver, T_c is the read bitline precharge transistor, T_m is the memory cell inverter.

Discussion

- Power models are based on detailed estimates of gate and wire capacitance and switching activity
- In process of validating power models against measured power numbers of existing routers and against low-level power estimation tools

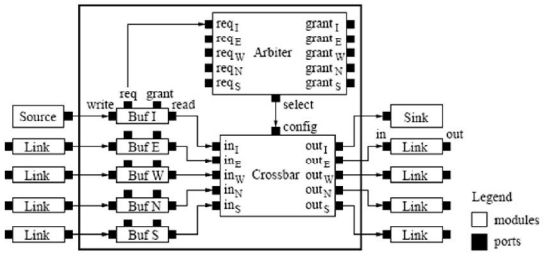
Discussion (cont.)

- Currently working with chip-to-chip and on-chip link designers to develop parameterized link models
- Power models (coded in C) as part of Orion's release

Walkthrough example: wormhole router

- Will move a head flit (smallest unit of flow control, and is a fixed-sized unit of a packet)
- Assumptions:
  - router has 5 input/output ports
  - 4 flit buffers per input port
  - each flit 32 bits wide
  - 5x5 crossbar
  - 4:1 arbiter per output port

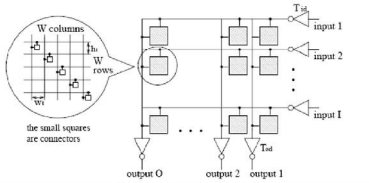
Model of simple wormhole router in Orion



$$E_{flit} = E_{wrt} + E_{arb} + E_{read} + E_{xb} + E_{link}$$

More Power Models

Table 3. Model for matrix crossbars

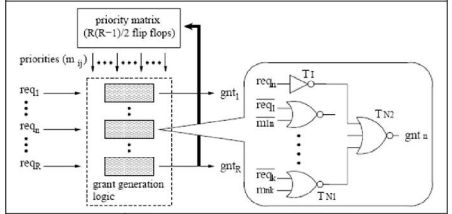
Canonical structure and notation	
	
connector	either a tri-state buffer or a transmission gate
C_{in_cnt}	input node cap. of a connector
C_{out_cnt}	output node cap. of a connector
C_{ctr_cnt}	control node cap. of a connector
Architectural parameters	
I	number of crossbar input ports
O	number of crossbar output ports
W	port width in bits
Technological parameters	
h_t	track height
w_t	track width

Capacitance equations	
input line length	$L_{in} = O \cdot W \cdot w_t$
output line length	$L_{out} = I \cdot W \cdot h_t$
input line cap.	$C_{xb\_in} = O \cdot C_{in\_cnt} + C_a(T_{id}) + C_w(L_{in})^*$
output line cap.	$C_{xb\_out} = I \cdot C_{out\_cnt} + C_a(T_{od}) + C_w(L_{out})$
control line cap.	$C_{xb\_ctr} = W \cdot C_{ctr\_cnt} + C_w(\frac{L_{in}}{2})$
Operation energy equations	
δ_{xi}	number of switching input bits
δ_{xo}	number of switching output bits
traversal energy	$E_{xb} = \delta_{xi} E_{xb\_in} + \delta_{xo} E_{xb\_out}$

(*) T_{id} is the input driver, T_{od} is the output driver.

More Power Models (cont.)

Table 4. Model for matrix arbiters

Canonical structure and notation	
	
C_{FF}	switch capacitance of a flip flop
C_{FC}	clock capacitance of a flip flop
Architectural parameters	
R	number of requesters

Capacitance equations	
request cap.	$C_{req} = C_a(T_I) + (R-1)C_g(T_{N1}) + C_g(T_{N2})^*$
grant cap.	$C_{gnt} = C_d(T_{N2})$
priority cap.	$C_{pri} = C_{FF} + 2C_g(T_{N1})$
internal cap.	$C_{int} = C_d(T_{N1}) + C_g(T_{N2})$
clock cap.	$C_{clk} = C_{FC}$
Operation energy equations	
δ_{ar}	number of switching request signals
δ_{ap}	number of switching priority bits
δ_{ai}	number of switching internal nodes
arbitration energy	$E_{arb} = \delta_{ar} E_{req} + \delta_{ap} E_{pri} + \delta_{ai} E_{int} + (E_{gnt} + E_{xb\_ctr})$

(*) T_{N1} is the first level NOR gate, T_{N2} is the second level NOR gate, T_I is the inverter.

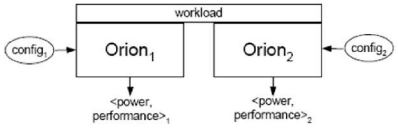
Orion

- Introduction
- Dynamic Network Simulator
- Power Modeling
- Case Studies
- Related Work
- Conclusions

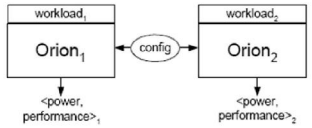
Case Studies

- Authors envision three primary uses for Orion
- They provide examples for each one of the envisioned uses
- The case studies are various design space explorations over the same system

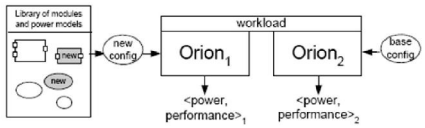
Three potential ways of utilizing Orion



(a) Exploring different configurations



(b) Exploring different workloads



(c) Exploring new microarchitectures

Experimental Setup

- 16-node network
- Each router has 5 physical bidirectional ports
- Router keeps count of available buffers
- No dropped packets
- Source dimension-ordered routing is used
- Simulator generates uniformly distributed traffic to random destinations unless otherwise mentioned

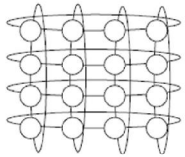
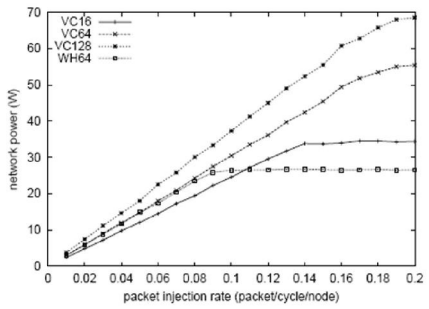


Figure 4. A 4-by-4 torus network.

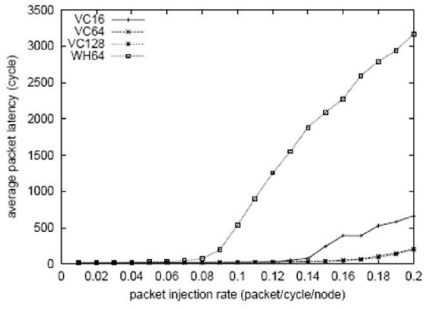
Exploring Configurations: wormhole vs. virtual-channel

- Four different router configurations are compared:
  - wormhole router with 64-flit input buffer per port (WH64).
  - virtual-channel (VC) router with 2 VCs per port and 8-flit input buffer per VC (VC16).
  - virtual-channel router with 8 VCs per port and 8-flit input buffer per VC (VC64).
  - virtual-channel router with 8 VCs per port and 16-flit input buffer per VC (VC128).

Results Case 1

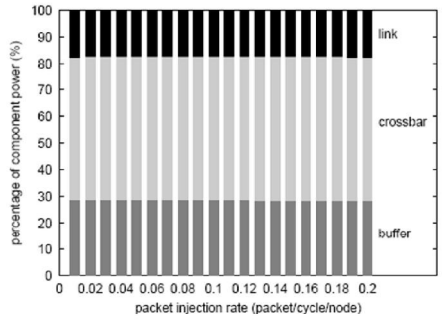


(b) total network power



(a) average packet latency

Results Case 1

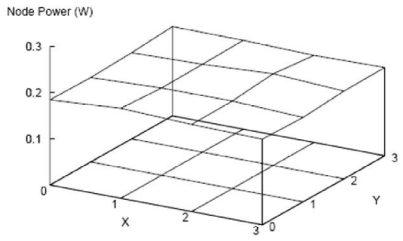


(c) VC64 average power breakdown, arbiter power is invisible at current scale.

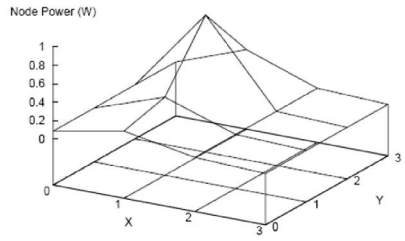
Exploring Workloads: broadcast vs. uniform traffic

- Compares two traffic patterns
  - uniform random traffic, *i.e.* each node injects packets to randomly distributed destinations other than itself in the network.
  - broadcast traffic, *i.e.* one node injects packets to all the other nodes in the network.

Results Case 2



(a) uniform random traffic

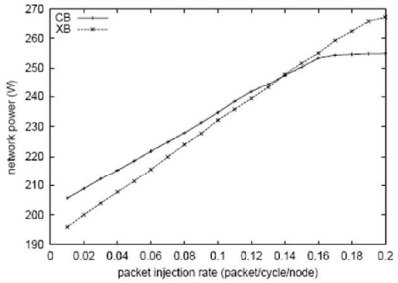


(b) broadcast traffic from node (1,2)

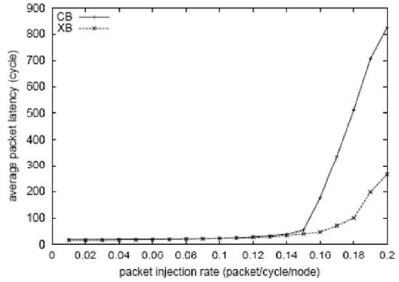
Exploring new micro-arch: central buffered routers

- Two different router architectures are explored:
  - Central-buffered router with a 4-bank central buffer, each 1 flit wide, 2560 chunks (2560 rows, each row 4-flit wide), 2 read ports, 2 write ports, and a 64-flit input buffer at each port (CB).
  - Input-buffered crossbar-based router with 16 virtual channels, 268-flit input buffer per VC and a 5x5 crossbar(XB).

Results Case 3 (random traffic)

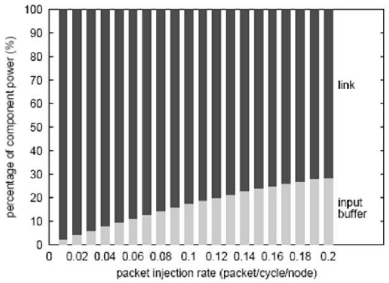


(b) total network power (random traffic)



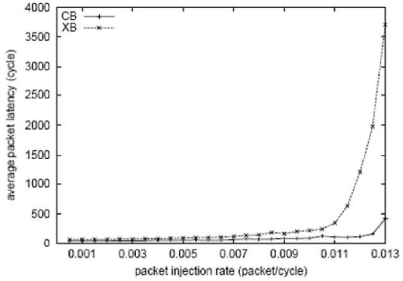
(a) average packet latency (random traffic)

Results Case 3 (random traffic)

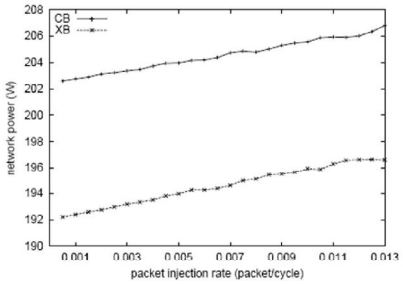


(c) average power breakdown of XB router (random traffic), arbiter power and crossbar power are invisible at current scale.

Results Case 3 (broadcast traffic)

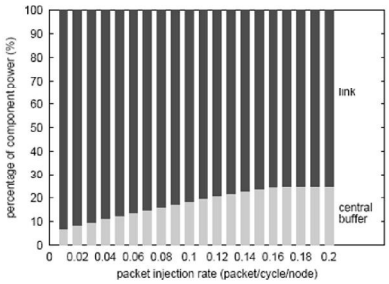


(d) average packet latency (broadcast traffic)



(e) total network power (broadcast traffic)

Results Case 3 (broadcast traffic)



(f) average power breakdown of CB router (random traffic), arbiter power and input buffer power are invisible at current scale.

Round-up

- Orion is a useful tool/model for rapidly exploring power-performance tradeoffs in network micro-architecture design
- Its design makes it relatively easy to port to other simulator infrastructures

Agenda

- Building-up a model: Orion
- Power estimation through Hardware Counters -Based on PARAPET group work at Princeton
- Architecture Simulators
- Conclusions
- Related Work

Hardware Counters

- Introduction
- Using Hardware Counters
  - Based on PARAPET group work (Princeton)
- Architecture Simulation
- Statistical sampling
- Conclusions

Advantages of Hdw Counters

- Power Models reflecting modern processors
  - Clock gating, power
  - Voltage regulation, di/dt
- Need for Fast-Realtime Modeling and Measurement to observe long time periods
  - Thermal time constants: O(s)
  - Not feasible even with architecural simulators
    - i.e.: 1s of real run ⇔~5 x IPC hrs of WATTCH simulation
- Need live, run-time power/thermal measures
  - Dynamic Thermal Management
  - Power-Aware OS & Systems control

Where all this is useful?

- Measurement/Modeling for microarchitectural details
- Compiler level power
  - SW power profiling
- Power Aware OS
  - Dynamic power/thermal/March. Configuration
    - Dynamic memory allocate, Process cruise control, etc.
- Demonstrates modern processor power
- Need for speed! Long Timescales, thermal constants
- Identify program phases w/o knowledge of code, no basic block info whatsoever
- Program signatures for detailed simulation, say: "power points rather than simpoints"

INTRODUCTION

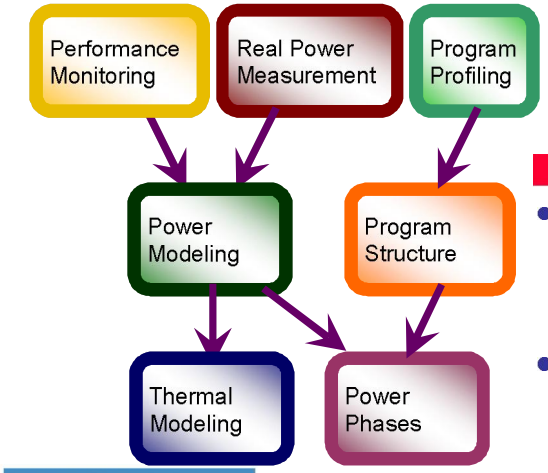
- Runtime processor power
  - Measurement with HW
  - Estimation with Performance counters
    - CPU Unit Power Breakdowns
    - Runtime verification

Detailed

- Processor thermal modeling
- Power Phase Behavior of programs
- Mapping between power behavior and program structure

Touch

THE BIG PICTURE

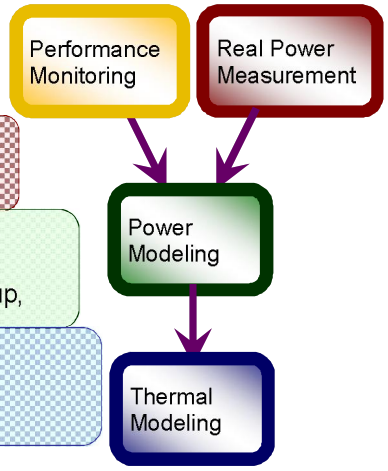


Bottom line…

- *To Estimate component power & temperature breakdowns for P4 at runtime…*
- *To analyze how power phase behavior relates to program structure*

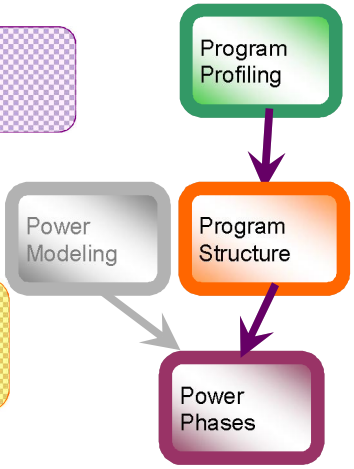
Agenda

- Performance Monitoring
  - P4 Performance Counters
  - Performance Reader LKM
- Real Power Measurement
  - P4 Power Measurement Setup
  - Examples
- Power Modeling
  - P4 Power Model
  - Model + Measurement Sync Setup, Verification
- Thermal Modeling
  - Brief Thermal Model Intro
  - Ppro Thermal Model Results



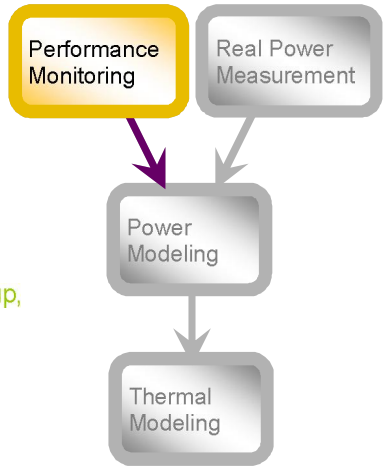
Bonus Material

- Power Phase Behavior
  - Similarity Based on Power Vectors
  - Identifying similar program regions
- Profiling Execution Flow
  - Sampling process' execution
  - "PCsampler" LKM
- Program Structure
  - Execution vs. Code space
  - Power Phases ⇔ Exec. Phases
    - <OR VICE VERSA>

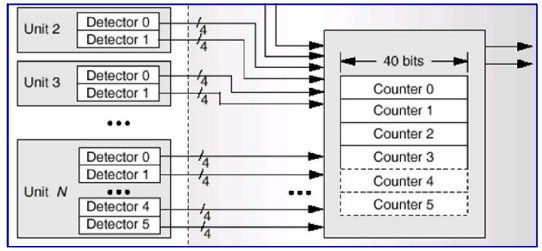


Performance Monitoring

- Related Work
- Performance Monitoring
  - P4 Performance Counters
  - Performance Reader LKM
- Real Power Measurement
  - P4 Power Measurement Setup
  - Examples
- Power Modeling
  - P4 Power Model
  - Model + Measurement Sync Setup, Verification
- Thermal Modeling
  - Refined Thermal Model
  - Ex: Ppro Thermal Model



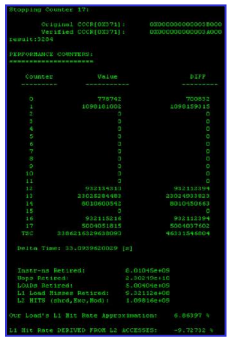
Live CPU Performance Monitoring with Hardware Counters



- Most CPUs have hardware performance counters
- P4 Performance Monitoring HW:
  - 18 Event Counters
  - 18 Counter Configuration Control Registers
    - Configure how to count
  - 45 Event Selection Control Registers
    - Configure what to count
  - Additional Control Registers

Our Event-Counter: Performance Reader

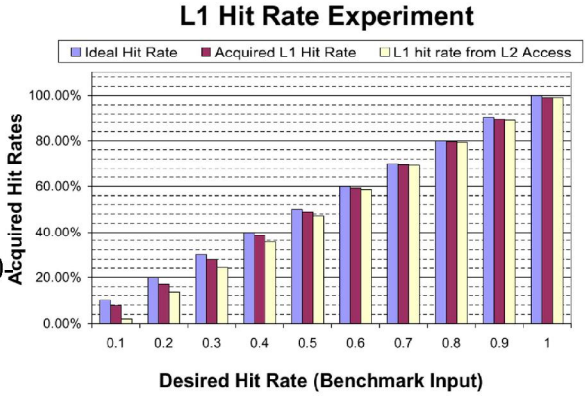
- Performance Reader implemented as Linux Loadable Kernel Module
  - Implements 6 syscalls:
    - *select_events()*
    - *reset_event_counter()*
    - *start_event_counter()*
    - *stop_event_counter()*
    - *get_event_counts()*
    - *set_replay_MSRs()*
- User Level Interface:
  - Defines the events → Starts counters
  - Stops counters → Reads counters & TSC



Event Types:
- 59 event classes
- 100s of events to count

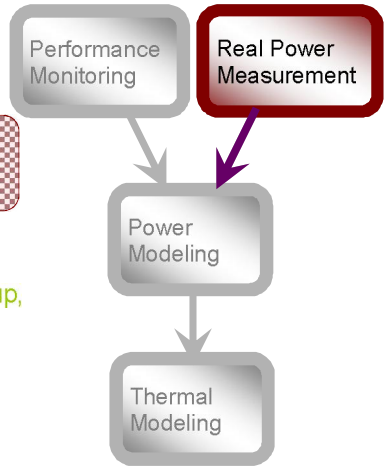
Performance Reader: Example Validation

- L1_Dcache benchmark
- Controls cache hit behavior
- Validated against measured cache events
- Vary hit rate from 0-100%

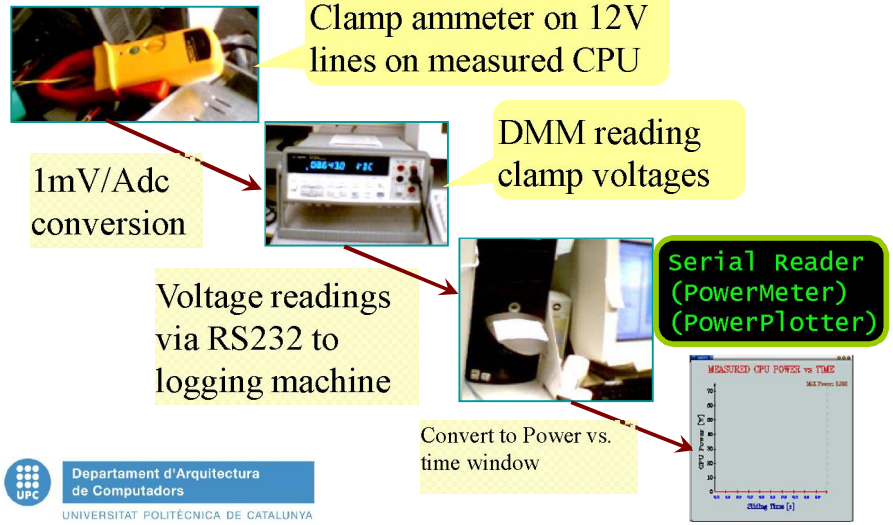


Processor Power Measurement

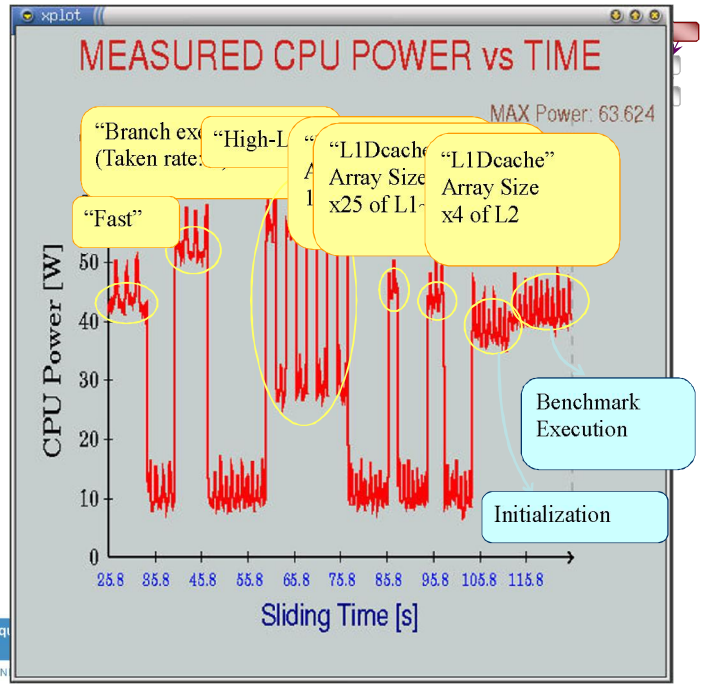
- Related Work
- Performance Monitoring
  - P4 Performance Counters
  - Performance Reader LKM
- [redacted]
- Power Modeling
  - P4 Power Model
  - Model + Measurement Sync Setup, Verification
- Thermal Modeling
  - Refined Thermal Model
  - Ex: Ppro Thermal Model



P4 Power Measurement Setup

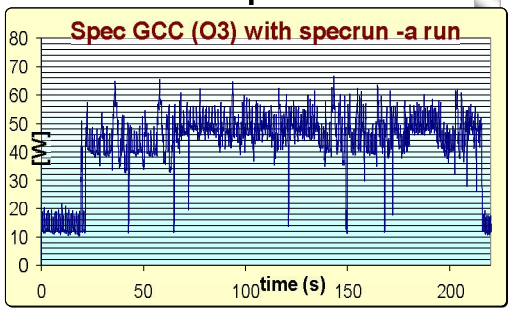


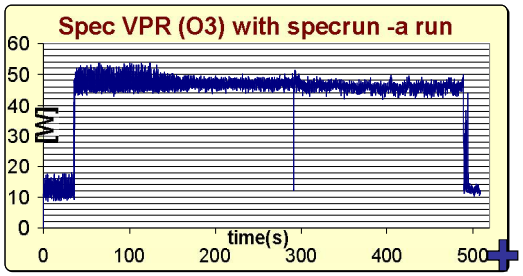
PowerPlotter: Example



SPEC Power Examples

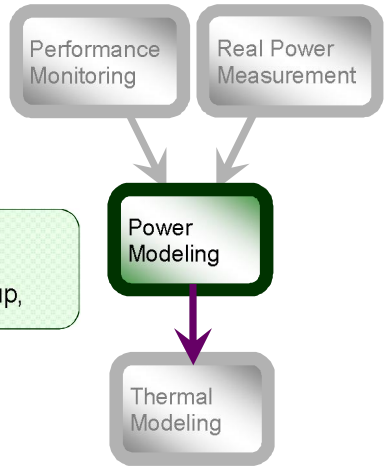
- Different programs show very different power characteristics
- Timescale of interest can be huge => inaccessible via simulation





Processor Power Modeling

- Related Work
- Performance Monitoring
  - P4 Performance Counters
  - Performance Reader LKM
- Real Power Measurement
  - P4 Power Measurement Setup
  - Examples
- Power Modeling
  - P4 Power Model
  - Model + Measurement Sync Setup, Verification
- Thermal Modeling
  - Refined Thermal Model
  - Ex: Ppro Thermal Model



P4 POWER MODEL

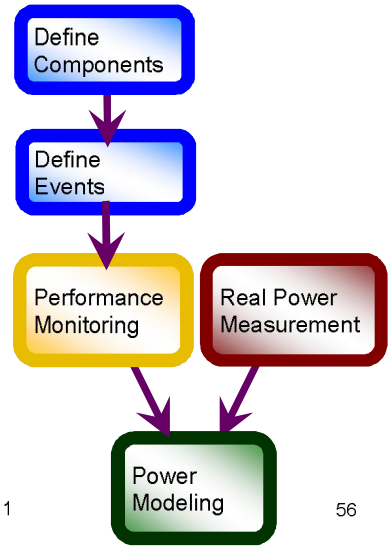
Define components (I.e. L1 cache, BPU, Regs, etc.), whose powers we'll model:
- from annotated layout

Determine combination of P4 events that represent component accesses best

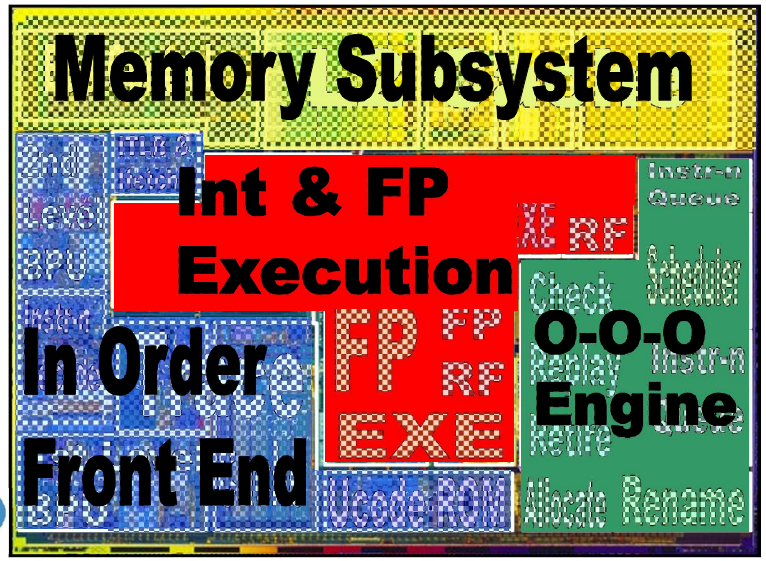
Gather counter info with minimal power overhead and program interruption

Convert counter info into component power breakdowns

Verify total power against measured processor power



Defining Components



Defining Events → Access Rates

- We determined 24 events to approximate access rates for 22 components
- Used Several Heuristics to represent each access rate
- **Examples:**

	Access Heuristics
Bus Control	$\frac{IOQ\ Allocation}{\Delta Cycles_1} + \frac{Bus\ Ratio \cdot FSB\ Data\ Activity}{\Delta Cycles_2}$
Front End BPU	$\frac{8 \cdot ITLB\ Reference}{\Delta Cycles_1} + \frac{Branch\ Retired}{\Delta Cycles_2}$
L1 Cache	$\frac{Ld\ Port\ Replay + St\ Port\ Replay}{\Delta Cycles_1} + \frac{Front\ End\ Event}{\Delta Cycles_2}$
Trace Cache	$\frac{Uop\ Queue\ Writes}{\Delta Cycles_1}$
Integer Execution	$2 \cdot (\frac{Uop\ Queue\ Writes}{\Delta Cycles_1} - FP\ Exe.\ Access\ Rate) - L1\ Cache\ Access\ Rate - \frac{Branch\ Retired}{\Delta Cycles_2}$

- Need to rotate counters 4 times to collect all event data
  - Used 15 counters & 4 rotations to collect all event data

Access Rates → Component Powers

- "Performance Counter based Access Rate estimations are used as proxy for max component power weighting together with microarchitectural details in order to estimate processor sub-unit powers"

$$Power(C_i) = AccessRate(C_i) \cdot ArchitecturalScaling(C_i) \cdot MaxPower(C_i) + NonGatedClockPower(C_i)$$

  - EX: Trace cache delivers 3 uops/cycle in deliver mode and 1 uop/cycle in build mode:
    - Power(TC)=[Access-Rate(TC)/3 + Access-Rate(ID)] x MaxPower(TC) + Non-gated TC CLK power
- Total power is computed as the sum of all 22 component powers + measured idle power (8W):

$$Total\ Power = \sum_{i=1}^{22} Power(C_i) + Idle\ Power$$

Experiment Setup – Recall:

Clamp ammeter on 12V lines on measured CPU

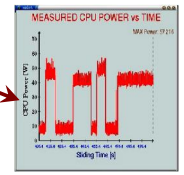
1mV/Adc conversion

DMM reading clamp voltages

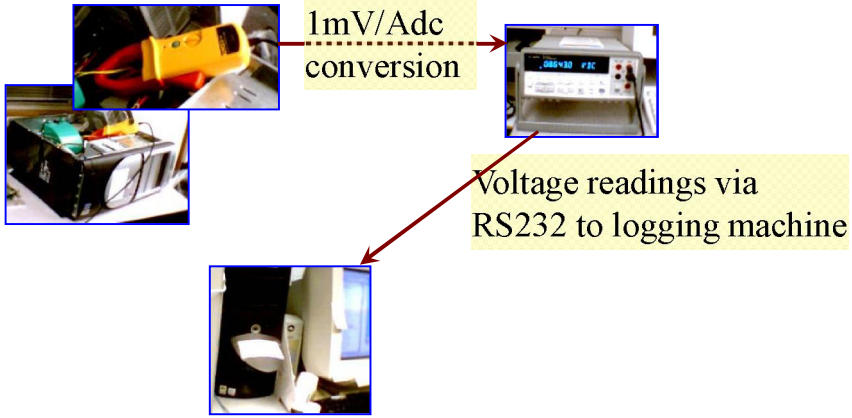
Voltage readings via RS232 to logging machine

Serial Reader (PowerMeter) (PowerPlotter)

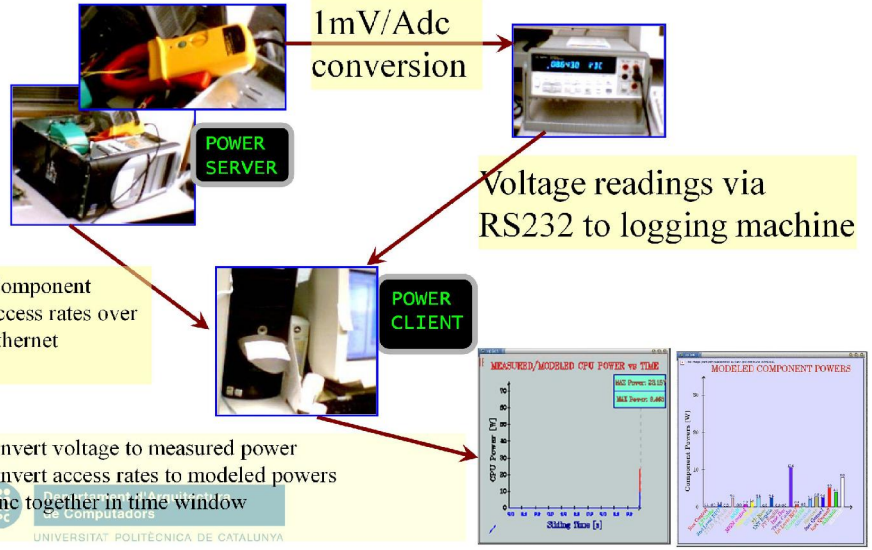
Convert to Power vs. time window



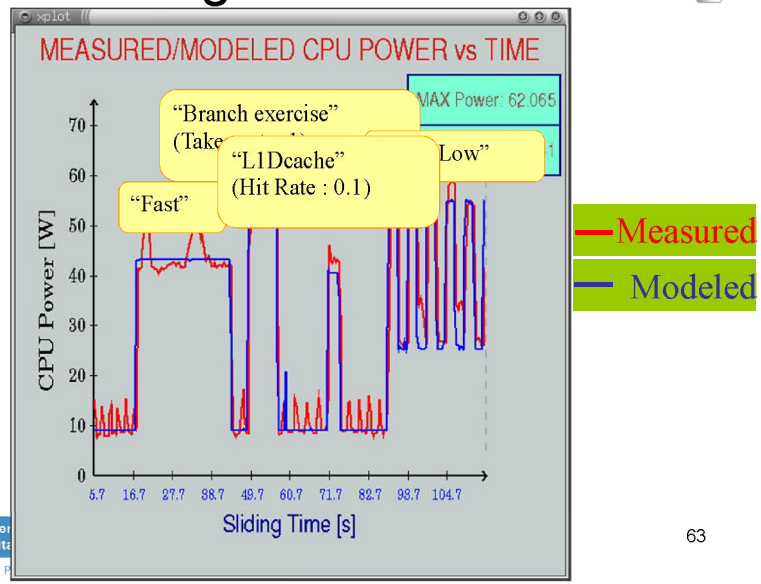
Experiment Setup



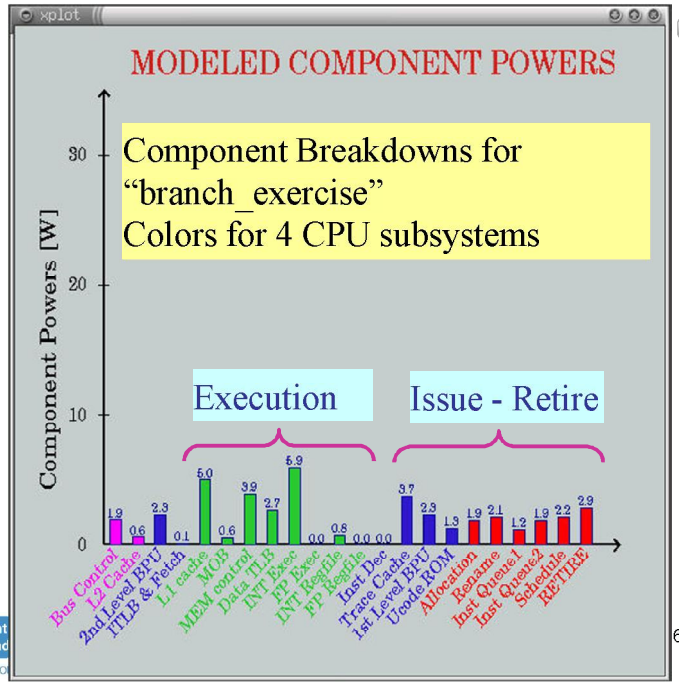
Experiment Setup



Tuning Benchmarks



Component Breakdowns



Benchmark Power Breakdowns

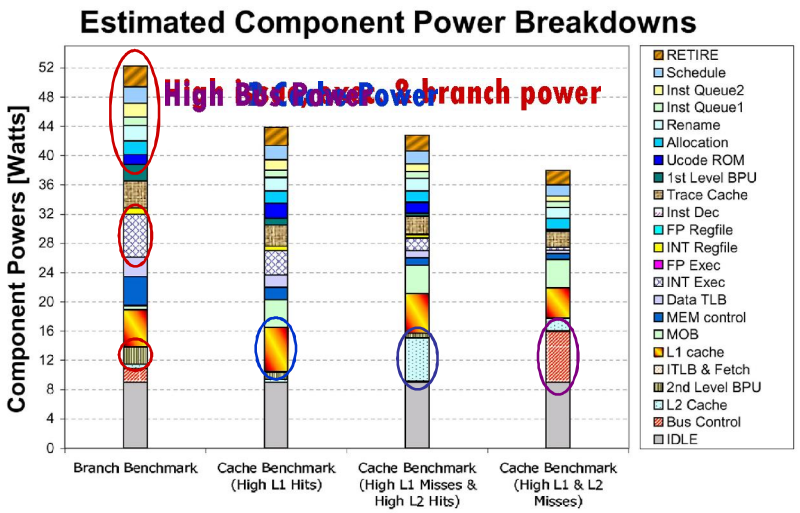
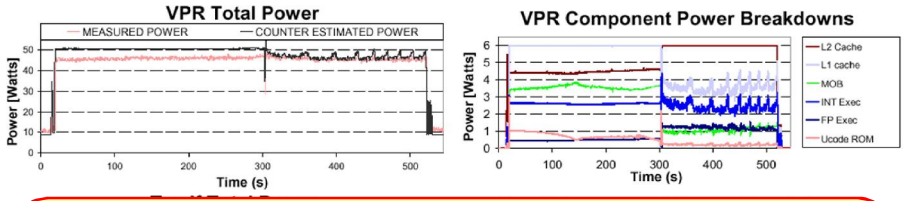


Figure 8. Power breakdowns for branch and cache benchmarks.

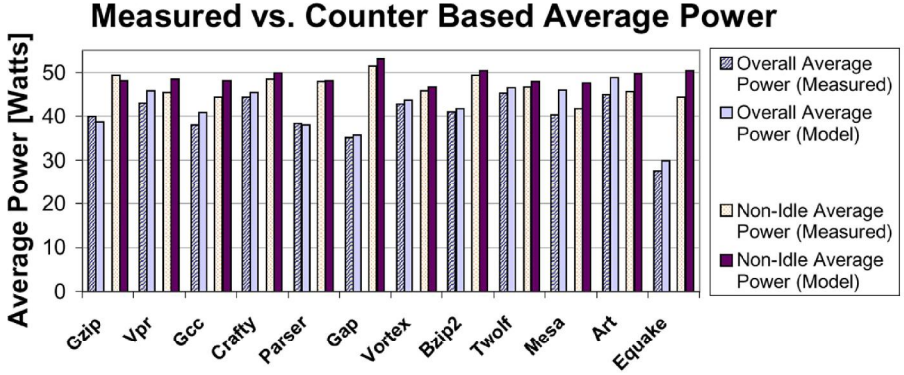
SPEC2000 Results



Equake Elaboration: (FP benchmark)
Initialization and computation phases
FP intensive mesh computation phase
Initialization with high complex IA32 instructions

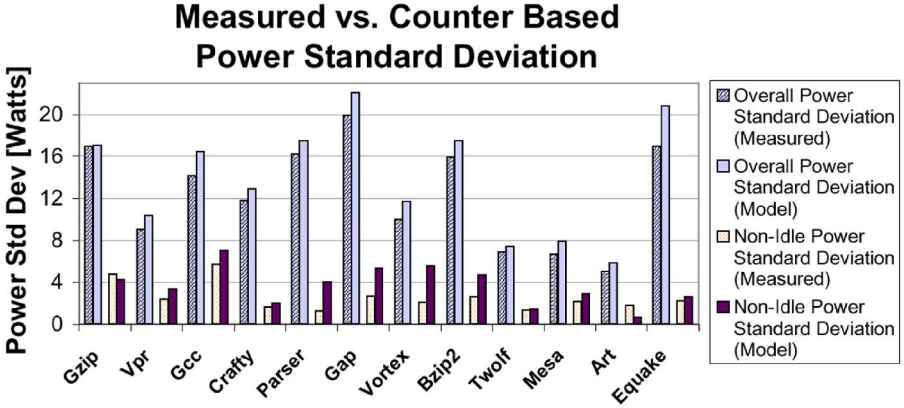
Twolf Elaboration:(Integer benchmark)
Several loop computations traversing memory
<High Memory Power>
Although ~const. Total power, component powers have slight gradients

Average SPEC Total Powers



- 1st set: Overall, 2nd set: Non-idle power
- Average difference between measurement and estimation: 3W
- Worst case: Equake (5.8W)

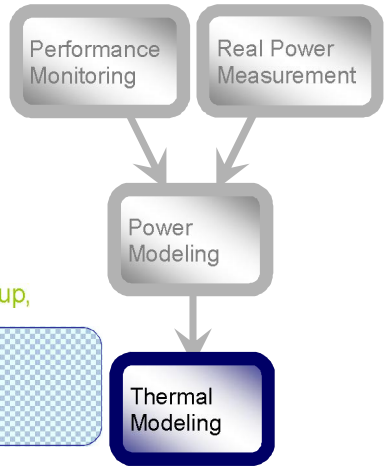
Stdev of SPEC Total Powers



- 1st set: Overall, 2nd set: Non-idle power
- Average difference: 2W
- Worst case: Vortex (3.5W)

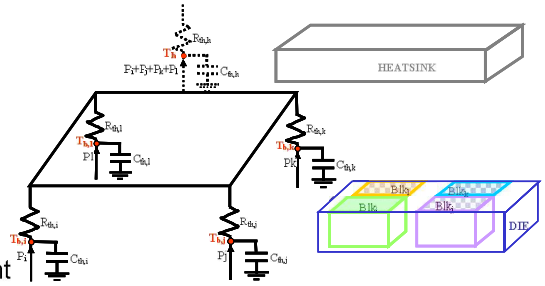
Thermal Model

- Related Work
- Performance Monitoring
  - P4 Performance Counters
  - Performance Reader LKM
- Real Power Measurement
  - P4 Power Measurement Setup
  - Examples
- Power Modeling
  - P4 Power Model
  - Model + Measurement Sync Setup, Verification
- Thermal Modeling
  - Brief Thermal Model Intro
  - Ppro Thermal Model Results

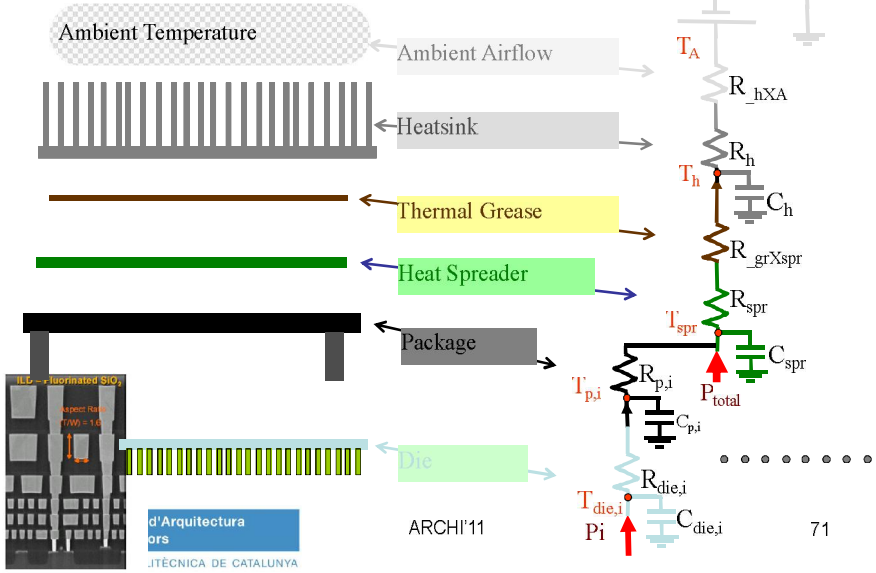


THERMAL MODELING: A Basic Model

- Based on lumped R-C model from packaging
- Built upon power modeling
  - Sampled Component Powers
  - Respective component areas
  - Physical processor Parameters
    - Packaging
    - Heat Transfer

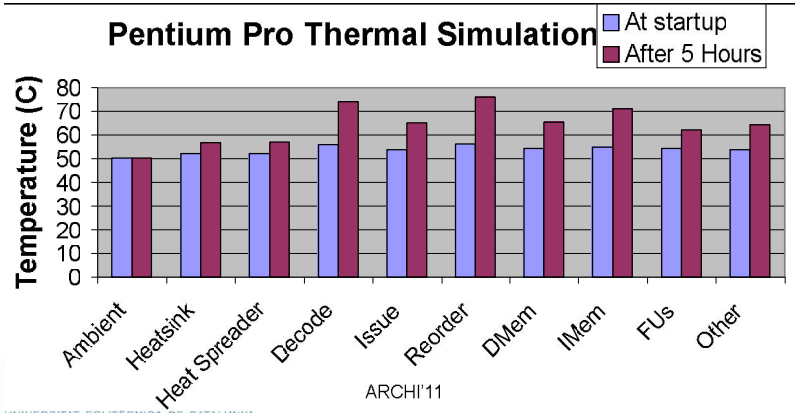


Physical Structure vs. Thermal Model



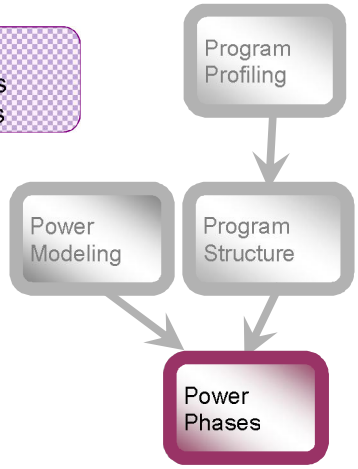
Simulation Outputs

- Thermal nodes updated every Δt~20ms
  - Component Temperatures Build up to ~350K in ~5hrs
  - $T_{heatsink}$ moves very slowly as expected



Power Phase Behavior

- Power Phase Behavior
  - Similarity Based on Power Vectors
  - Identifying similar program regions
- Profiling Execution Flow
  - Sampling process' execution
  - "PCsampler" LKM
- Program Structure
  - Execution vs. Code space
  - Power Phases ⇔ Exec. Phases
    - <OR VICE VERSA>



Power Vectors for Similarity

- Similar to basic block vectors
- Use component power vector samples to represent program phases
- Consider Manhattan distance between 2 vectors as the 'measure of dissimilarity' between the corresponding execution points
- Construct a similarity matrix to represent similarity among all pairs of execution points
  - Each entry in the similarity matrix:

$$Similarity\ Matrix(r,c) = \sum_{i=1}^{22} |Power_r(C_i) - Power_c(C_i)|$$

Gcc & Gzip Similarity Matrices

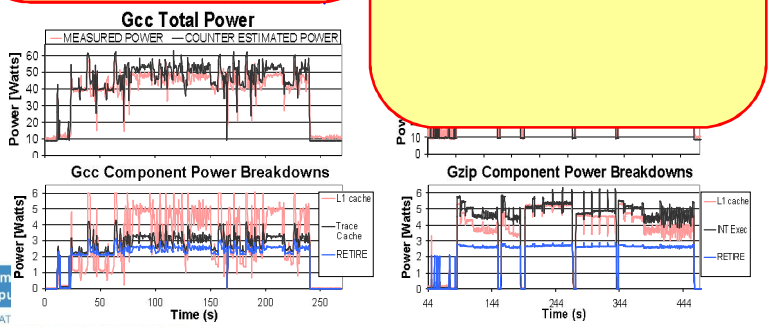
Gzip Elaboration:
Much regular power behavior

Spurious similarities such as 100-150s and 200-280 are distinguished by the similarity analysis

Gcc Elaboration:
Very variant power

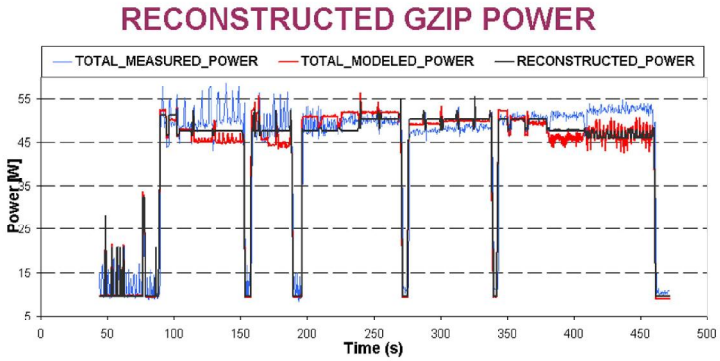
Almost identical power behavior at 30, 50, 180s.

Although 88s, 110s, 140s, 210s and 230s show similar total power; 88, 210 and 230 share higher similarity.



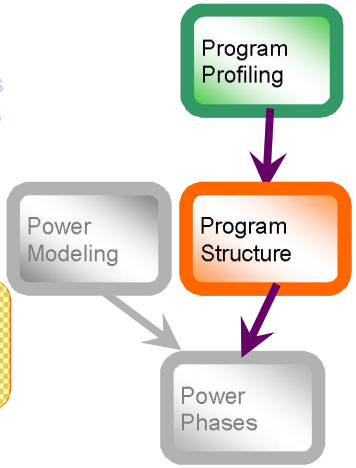
Generating representative vectors

- Gzip has ~1000 power vectors
- Cluster vectors based on similarity
- Could we represent power behavior with reasonable accuracy, with a small number of 'signature' vectors?
- **Ex: 26 representative vectors with "Thresholding Algorithm":**



Program Execution Profile

- Power Phase Behavior
  - Similarity Based on Power Vectors
  - Identifying similar program regions
- Profiling Execution Flow
  - Sampling process' execution
  - "PCsampler" LKM
- Program Structure
  - Execution vs. Code space
  - Power Phases ⇔ Exec. Phases
    - <OR VICE VERSA>



Program Execution Profile

- Sample program flow simultaneously with power
  - LKM implementation: "PCsampler"
- Generate code space similarity in parallel with power space similarity
- Relative comparisons for:
  - Complexity
  - Accuracy
  - Applicability, etc.

Round-up

- Hw counter give a precise measure of the CPU performance
- Good heuristics are needed to turn these event counters to activities
- Fast and accurate performance/power characterization
- Limited to that specific CPU. No design-space exploration possible.

Agenda

- Building-up a model: Orion
- Power estimation through Hardware Counters -Based on PARAPET group work at Princeton
- Architecture Simulators
- Conclusions
- Related Work

Wattch, Hotspot, Hotleakage

http://www.eecs.harvard.edu/~dbrooks/wattch-form.html
http://lava.cs.virginia.edu/HotSpot
http://lava.cs.virginia.edu/HotLeakage

Ramon Canal

Wattch, architecture power modelling

Idea: Perform cycle by cycle tracking of power dissipation
By estimating unit capacitances and activity factors.

Methodology:

Parameterized power models of common structures present in modern processors.

Basic mathematical equation:

Dynamic power consumption in CMOS processors

$$P_d = \alpha C V_{dd}^2 f$$

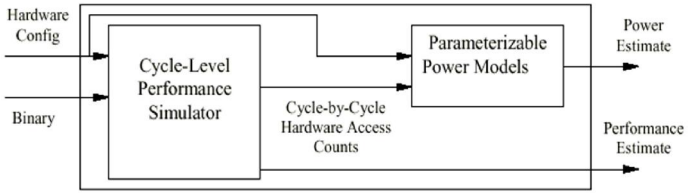
C : Load capacitance (farads)

V_{dd}: Supply voltage (V)

f : Clock frequency (Hz)

α : A number between 0 and 1 indicating how often clock ticks lead to activity on an average.

Power Modeling Methodology



Overall Structure of the Power Simulator

Classification of the processor units:

1. Array structures:
   Data and instruction caches, cache tag arrays, all register files, register alias tables etc.

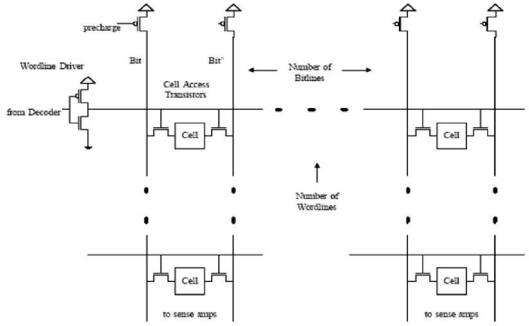


Figure 3: Schematic of wordlines and bitlines in array structure.

Classification of the processor units:

2. Fully associative Content-addressable memories:
   TLBs, Instruction window wake-up logic etc…

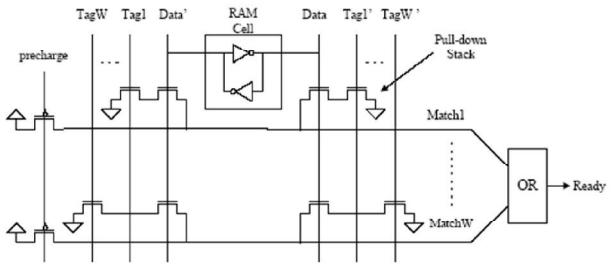


Figure 4: Core cell of wakeup logic modeled as a CAM.

Classification of the processor units:

3. Combinational Logic and Wires:
   Instruction selection logic, Functional units, result buses etc…
   - Result buses: Find their lengths and multiply by their capacitance per unit length.
   - Functional units: based on their own model and further scaling for process technology and frequency

4. Clocking Model:
   Clock buffers, clock wires and capacitive loads.
   - Global clock metal lines : A modified H-tree network in which the global clock signal is routed to all portions of the chip

SimpleScalar Interface

- The power models are interfaced with SimpleScalar
  - *sim-outorder*
    - modern out-of-order processors with 5-stage pipelines : fetch, decode, issue, writeback and commit
    - speculative execution
  - Conditional Clocking Styles
    - All or Nothing Clock Gating
      - full power for use and no power for idle
    - Linear Clock Gating with 10%
      - power is scaled linearly but with minimum background power
    - Linear Clock Gating
      - power is scaled according to the portion of a unit's ports accessed

SimpleScalar Interface

- The power models are interfaced with SimpleScalar
  - Simulation Speed
    - performance simulator + power model
    - 105K instructions per second → 80K instructions per second
    - PowerMill takes one hour to run 64-bit adder for 100 test vectors.
      → Wattch can simulate a full CPU running 280M instructions
    - quite tolerable overhead
  - Simulation Method
    - Compute the base power dissipation for each unit at program startup.
    - The access counts per unit are obtained from SimpleScalar simulator.
    - The base power costs are scaled with per-unit access counts.

How accurate is it?

Makers claim accuracy within 10% of estimates from industry's leading tools.

✓ Seen as a complement to low level tools but is 1000 times faster.

Hardware Structure	Intel Data	Model
Instruction Fetch	22.2%	21.0%
Register Alias Table	6.3%	4.9%
Reservation Stations	7.9%	8.9%
Reorder Buffer	11.1%	11.9%
Integer Exec. Unit	14.3%	14.6%
Data Cache Unit	11.1%	11.5%
Memory Order Buffer	6.3%	4.7%
Floating Point Exec. Unit	7.9%	8.0%
Global Clock	7.9%	10.5%
Branch Target Buffer	4.7%	3.8%

Table 4: Comparison between Modeled and Reported Power Breakdowns for the Pentium Pro®.

Hardware Structure	Alpha 21264	Model
Caches	16.1%	15.3%
Out-of-Order Issue Logic	19.3%	20.6%
Memory Management Unit	8.6%	11.7%
Floating Point Exec. Unit	10.8%	11.0%
Integer Exec. Unit	10.8%	11.0%
Total Clock Power	34.4%	30.4%

Table 5: Comparison between Modeled and Reported Power Breakdowns for the Alpha 21264.

Shortcomings

- ❑ Fails to model interconnect capacitances
  - ▪ Difficult to model capacitances of polysilicon wires as they vary with the physical layout.
  - ▪ There is no way to estimate the length of interconnect. Capacitance estimation errors gives us an overall error of 6-11%
  - ▪ Physical implementation may be different from the models assumed in Wattch. A critical thing to note!
  - ▪ Models only one decoder for an array structure.

Hotspot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Desgin

Kevin Skadron, Mircea R. Stan, et al.
University of Virginia

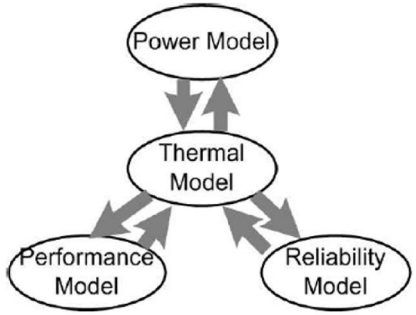
Overview

- Introduction to Thermal Models
- Hotspot thermal modeling in detail
- Thermal Model for Interconnects
- Validation of the Model
- Hotspot Applications

Impact of high temperatures on Very Large Scale Integrated Systems

- Degradation in carrier mobility
  - A single inverter is 35% slower at 110 C than at 60 C.
- Exponential increase in sub-threshold leakage.
- Increase in interconnect resistivity.
- Decrease in device lifetimes.
- Temperature plays an important role in the early and accurate estimation of power, performance and reliability.

Interactions among thermal model and power, performance and reliability models



The Need for Architecture level Thermal Modeling

Ability of architecture domain

- To use runtime knowledge of application behavior and
- Current thermal status of different units of the chip to adjust execution, distribute the workload to control thermal behaviour and exploit instruction-level parallelism.[2]

Hotspot - A microarchitecture level compact thermal model

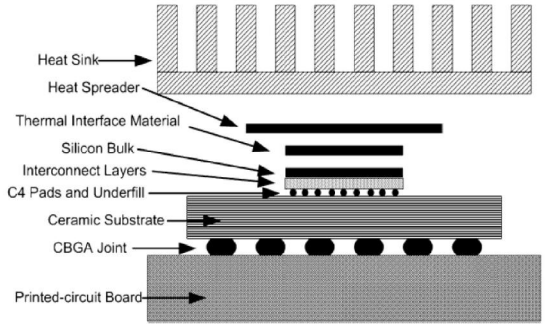
- Modeling methodology based on Compact Thermal Models (CTM) and stacked layer packaging configurations.
- Analytical investigation of the relationship between the number of nodes in the CTM and accuracy of the model.
- High-level model for on-chip interconnect self-heating power and temperature.

Compact Thermal Model in Hotspot

- The model is based on the duality between heat transfer and electrical phenomena.
- Rationale behind this duality – Heat flow and electrical current can be described by similar set of differential equations.

Thermal quantity	unit	Electrical quantity	unit
P, Heat flow, power	W	I, Current	A
T, Temperature difference	K	V, Voltage	V
R_{th}, Thermal resistance	K/W	R, Electrical resistance	Ω
C_{th}, Thermal capacitance	J/K	C, Electrical capacitance	F

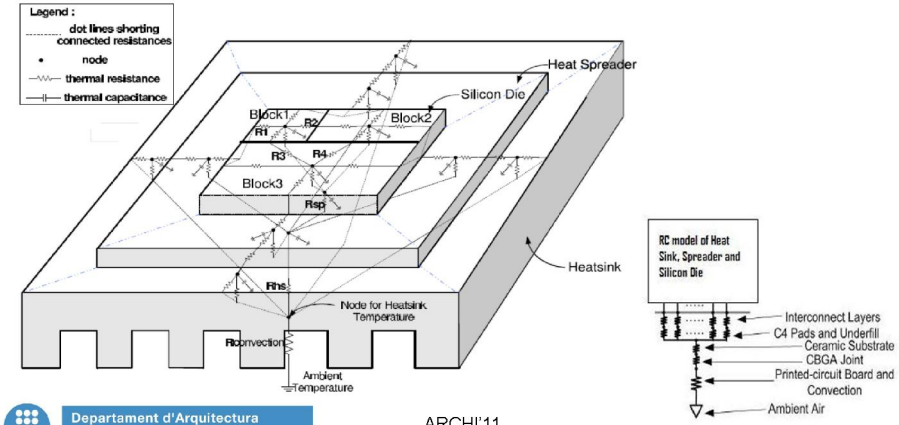
A typical Ceramic Ball Grid Array Package



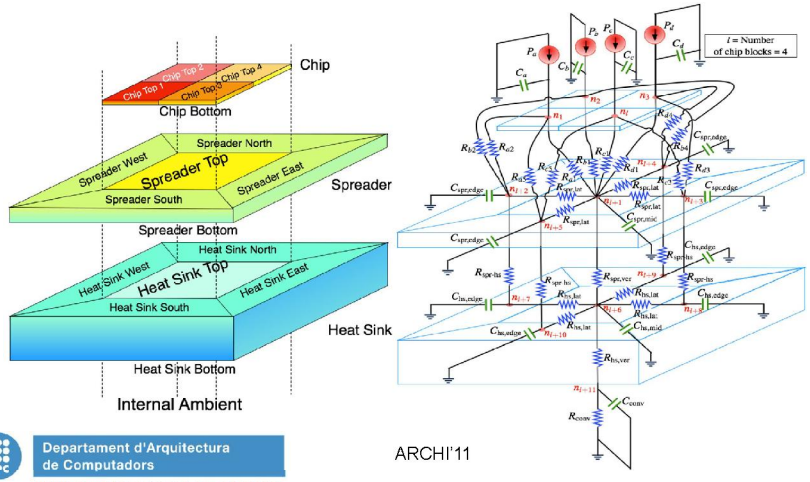
Model Overview

- The CTM proposed is essentially a thermal RC circuit.
- Each node in the circuit corresponds to a block at the desired level of granularity.
- Heat dissipation is modeled using a current source connected to each node.
- Solving the thermal RC circuit gives the temperature at each node.

Example Hotspot RC model



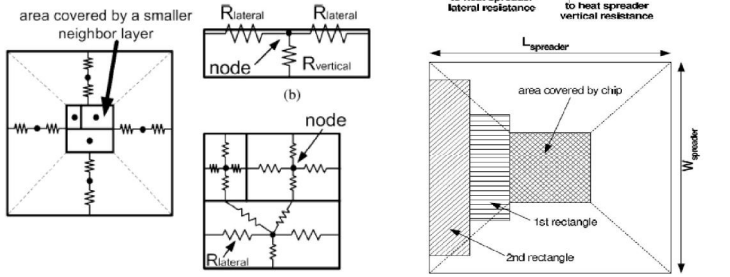
A closer view of the RC circuit



Rs and Cs in detail

- Thermal Resistance and Capacitances

$$R = \frac{t}{k \cdot A} \qquad C = c \cdot t \cdot A$$

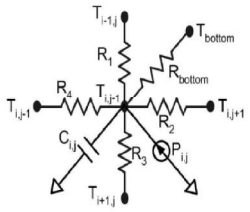


Solving the thermal RC circuit

$$P_{i,j} = \frac{T_{i,j} - T_{i-1,j}}{R_1} + \frac{T_{i,j} - T_{i,j+1}}{R_2} + \frac{T_{i,j} - T_{i+1,j}}{R_3} + \frac{T_{i,j} - T_{i,j-1}}{R_4} + \frac{T_{i,j} - T_{bottom}}{R_{bottom}} + \frac{C_{i,j}\Delta T_{i,j}}{\Delta t}$$

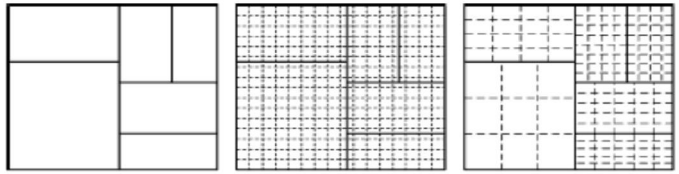
$$\Delta T_{i,j} = \frac{P_{i,j}\Delta t}{C_{i,j}} + \frac{\Delta t}{C_{i,j}}\left(\frac{T_{i-1,j}}{R_1} + \frac{T_{i,j+1}}{R_2} + \frac{T_{i+1,j}}{R_3} + \frac{T_{i,j-1}}{R_4} + \frac{T_{bottom}}{R_{bottom}}\right) - \frac{\Delta t}{C_{i,j}} G_{T_{i,j}}$$

$$G_{T_{i,j}} = \frac{1}{R_1} + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_{bottom}}$$



Modeling at different levels of Granularity

Temperature can be modeled at different granularity levels by dividing the die into a number of grid cells.



Derivation for the minimum grid cell size

Condition :

Temperature diff across one grid cell ≤

p % of the max. temperature difference across the die.

$$\Delta T_{\text{grid}} \leq T\left(\frac{-w}{2}\right) - T(0).$$

$$T\left(\frac{-w}{2}\right) - T(0) = \frac{T_{\max 0}}{2}\left(1 - e^{\frac{-w}{2t}}\right) = p\% \cdot T_{\max 0}$$

$$w = 2 \cdot t \cdot \ln \frac{1}{1 - 2 \cdot p\%}.$$

Interconnect Self-Heating Power and Thermal Modeling

1. The average self-heating power of interconnects in each metal layer,
2. The equivalent thermal resistance for metal wires. (including vias)

1. Interconnect self-heating power model

- Self-heating power of a metal wire is given by

$$P_{\text{self}} = I^2 \cdot R = I^2 \cdot \rho_m \cdot l / A_m$$

- Metal resistivity is temperature-dependent.
- The model needs to predict wire temperature before physical layout is available.
- Hence, it has to be able to predict the average wire length and self-heating current.

a) Average interconnect length in each metal layer

- Different methods of prediction for Signal Interconnects and Power Distribution Network because of the difference in their routing schemes.
- The wire-length distribution model presented by Davis et al.[3] (involving the concept of Rent's rule) is used in determining the average interconnect length in each layer.

b) Average Interconnect rms self-heating current in each metal layer

- The average current flow through the interconnect is solved from the equation,

$$I_{\mathrm{RMS}}^2(R_{\mathrm{tr}} + R_{\mathrm{wire}})t_d = \frac{1}{2}\alpha C_L V_{\mathrm{dd}}^2$$

I_{RMS} – Self-heating current per wire in each metal layer,
R_{tr} – On-resistance of the transistor,
R_{wire} – Wire resistance,
t_d – Delay of the switching event.

c) Interconnect rms self-heating current for Power supply sections

- Method 1:
  Modeled and solved using an RC circuit similar to that of the silicon die layer.
- Method 2:
  By dividing the total current delivered to a metal layer by the number of power grid sections.

d) Total Interconnect self-heating power in each metal layer

- Self-heating power of a metal layer *i* is given by

$$P_{\mathrm{self\_}i} = P_{\mathrm{wire\_sig\_}i} \cdot n_{\mathrm{sig\_}i} + P_{\mathrm{wire\_pwr\_}i} \cdot n_{\mathrm{pwr\_}i}$$

$P_{wire\_sig\_i}$ and $P_{wire\_pwr\_i}$ are the self-heating power of each individual signal interconnect and power supply wire for metal layer I,

n_{sig_i} and n_{pwr_i} are the number of signal interconnects and power supply sections in metal layer i.

Accuracy concerns about the interconnect power and thermal model

- Usefulness of the Interconnect model
- Accuracy concern of Rent's Rule
- Concern about current loading accuracy.

Computation Speed of Hotspot CTMs

- In the order of milliseconds to minutes. (AMD MP 1.5 GHz Dual-Processor)

simulated time interval	CPU time
0.1 ms	20 ms
20 ms	100 ms
2 sec	7.9 sec
20 sec	78.5 sec
steady-state	14.9 sec

- The small overhead is due to the manageable number of nodes in the lumped RC circuit and
- Because of the use of first-order differential equations to iteratively solve the RC network.

Validation

- An FPGA based system with 6 functional blocks and temperature sensors is implemented.
- The errors between Hotspot model and the thermal sensor measurements is found to be within 10%.

Unit	Power(mW)	Sensor Temperature	HotSpot Temperature
blank1	0.1	3.4	3.37
left_ppc	75	3.5	3.69
bott_ppc	75	3.4	3.67
ppc	45	3.5	3.66
mb	313	4.1	3.96
blank2	0.1	3.4	3.38

Hotspot Applications

- Temperature estimations are used for temperature-aware design.
- The CTMs have been used to leakage power calculations.
- Used to explore different Dynamic Thermal Management Techniques (DTM).
- For an accurate interconnect lifetime predictions.

Floorplan

- Use available floorplans in the tool
- Manually adapt the floorplan
- Use a graphic tool such as QUILT
  - http://www.ece.rochester.edu/research/acal/quilt/

Case study 1:

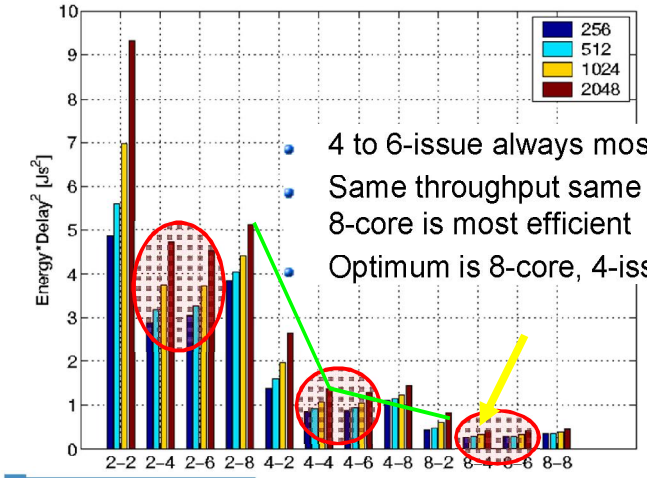
Design Space Exploration for Multicore Architectures: A Power/Performance/Thermal View

Main Goals

- Exploration of private-L2 shared memory Chip Multiprocessors
- Understand interactions among
  - Power consumption
  - Temperature
  - Chip floorplan

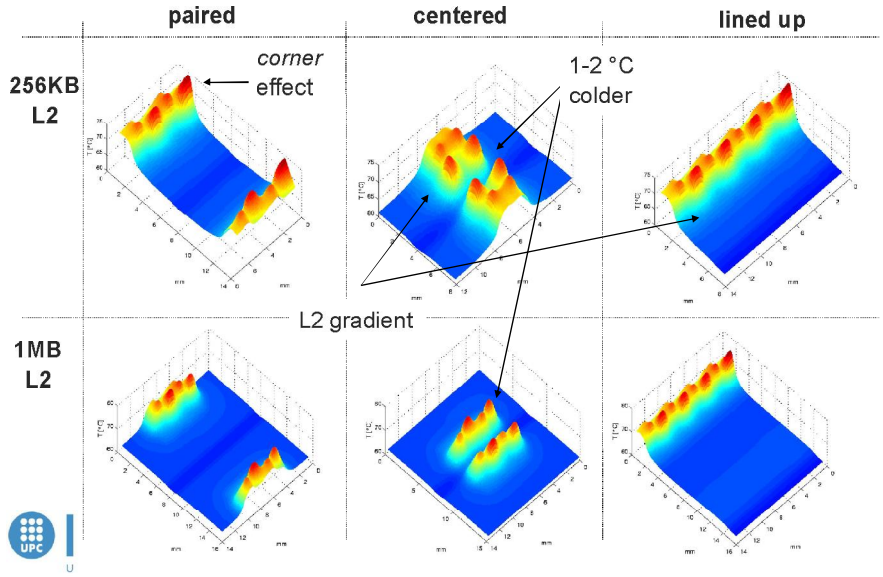
2-8 cores, 2-8 wide issue, 256KB-2M L2 caches
(Benchmarks: Splash-2, AlpBench)

Energy-Delay^2 Trade-offs

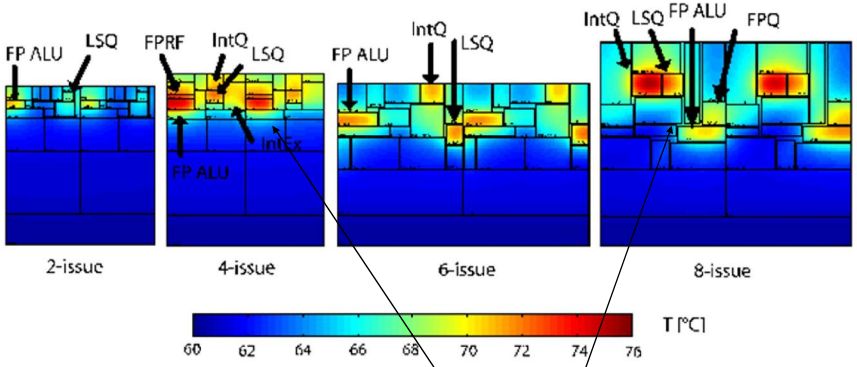


- 4 to 6-issue always most efficient
- Same throughput same memory, then 8-core is most efficient
- Optimum is 8-core, 4-issue, 256KB-L2

Floorplan Evaluation



Processor Complexity



- 2-core, 256KB-L2 and variable issue
- Hotspots:
  - FP ALU
  - LSQ & Int Queue for wide issue
- Inter-proc thermal coupling
- Intra-porc thermal coupling

Case study 2:

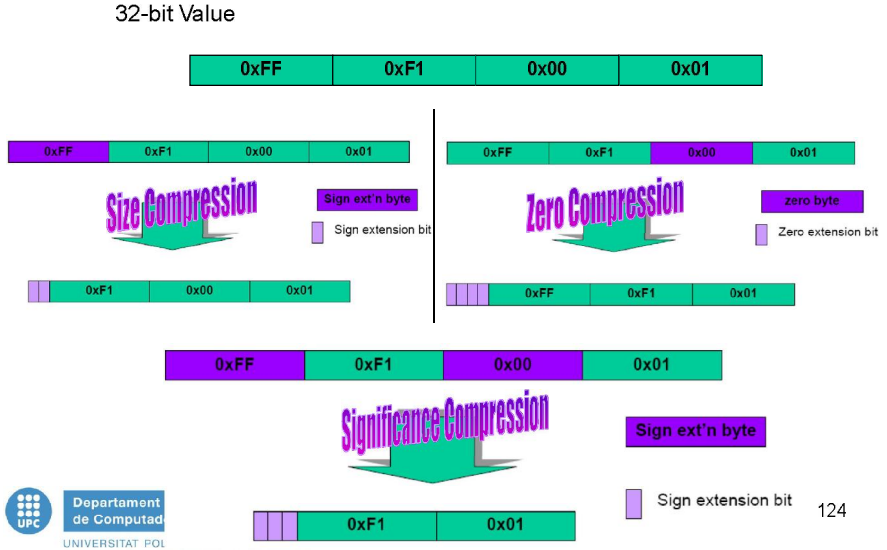
Value Compression for Hot Spot reduction

Main Goals

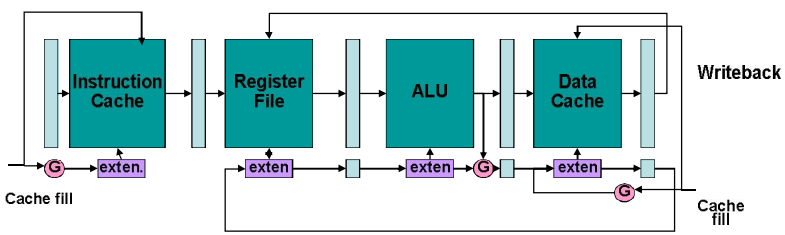
- Exploration of the power/thermal behaviour of architectures with value compression
- Compress values in all microarchitectural blocks
  - Size compression
  - Significance compression
  - Zero compression (Villa et al., MICRO 2000)

Single core, 4 wide issue, 64KB L1 cache, 2MB L2 cache
(Benchmark: Crafty -SpecInt2K-)

Value compression styles



Hw Value Compression

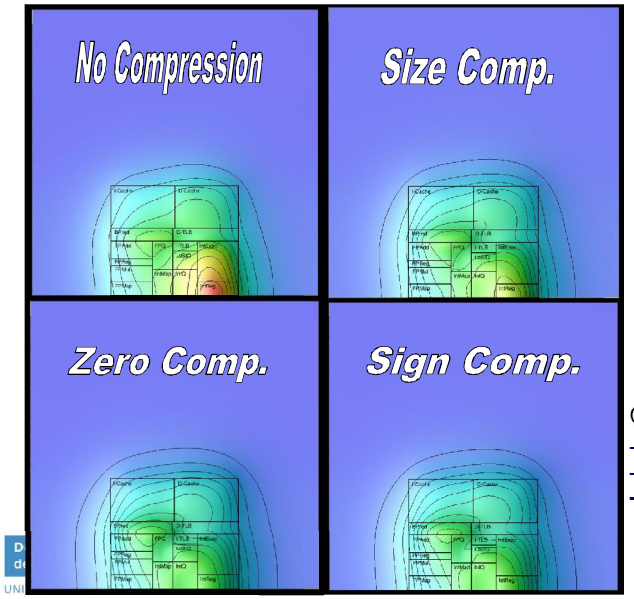


32-bit embedded processor pipeline with value compression

- Dynamically compress values flowing through the pipeline, with or without compiler help.
- Good for embedded and high performance processors!!

"Very Low Power Pipelines using Significance Compression", MICRO-33
"Software-Controlled Operand-Gating", CGO 2004
"Value Compression for Efficient Computation", EuroPar 2005

Thermal behaviour



Configurations used:
- Size 8-16-32-64
- Zero 8-16-24-32-40-48-56-64
- Sign 8-16-24-32-40-48-56-64

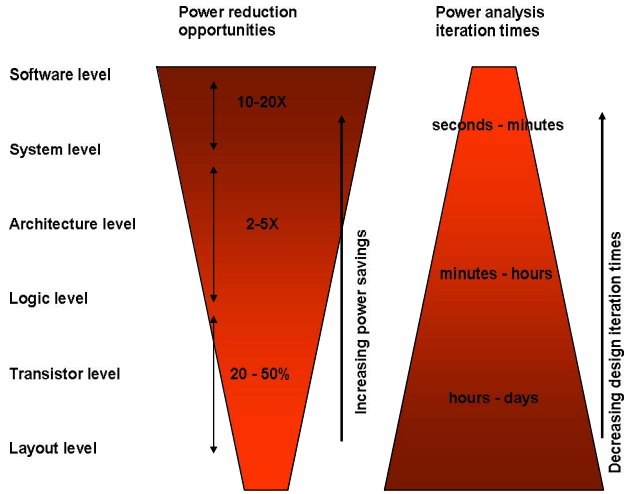
Conclusions

- Framework for power/temperature/leakage evaluation of microarchitectures:
  - Wattch + HotSpot + Leakage model
- But… beware of:
  - Integration: All the tools should use the same technology parameters
  - Floorplan: needs to be updated if the structures change
  - Accuracy is limited (but accepted in the area)

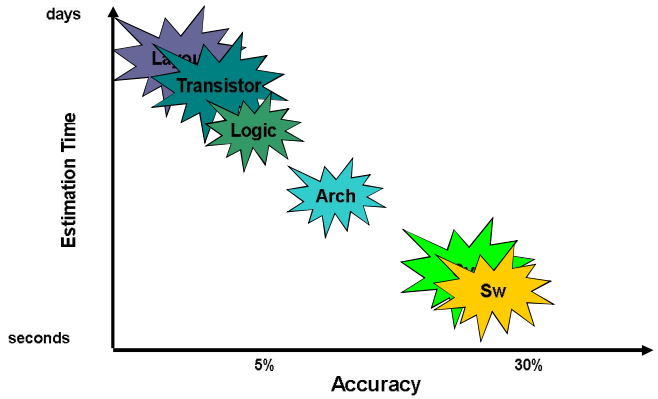
Agenda

- Building-up a model: Orion
- Power estimation through Hardware Counters -Based on PARAPET group work at Princeton
- Architecture Simulators
- Conclusions
- Related Work

Conclusions



Conclusions



RELATED WORK

- **Implementing counter readers:**
  - PCL [Berrendorf 1998], Intel VTune, Brink & Abyss [Sprunt 2002]
- **Using counters for Performance:**
  - HPC [Crummey 2001], CPU profilers
- **Using counters for Power:**
  - CASTLE [Joseph 2001], power profilers
  - event driven OS/cruise control [Bellosa 2000,2002]
- **Real Power Measurement:**
  - Compiler Optimizations [Seng 2003]
  - Cycle-accurate measurement with switch caps [Chang 2002]

RELATED WORK

- **Power Management and Modeling Support:**
  - Instruction level energy [Tiwari 1994]
  - PowerScope: Procedure level energy [Flinn 1999]
  - Event counter driven energy coprocessor [Haid 2003]
  - Power-breakdown driven energy reduction [Huang 2001]
  - Virtual Energy Counters for Mem. [Kadayif 2001]
  - ECOsystem: OS energy accounting [Ellis 2002]
- **Thermal Management and Modeling Support:**
  - PID based DTM [Skadron 2002]
  - Architectural Thermal Model [Skadron 2003]
  - Evaluating DTM techniques [Brooks 2001]

RELATED WORK – performance monitoring

- **implementing counter readers:**
- PCL Performance Counter Library, by Rudolf Berrendorf (University of Applied Sciences Bonn-Rhein-Sieg), Heinz Ziegler, and Bernd Mohr at the Central Institute for Applied Mathematics (ZAM) at the Research Centre Juelich , Germany
  - uniform interface for several architectures (intel Pentium,MMX, Pro, III, 4/linux; IBM Power3, Power3-II/AIX; etc.)
  - Software library with C, C++, Java & Fortran Bindings
  - Kernel patch (Mikael Pettersson)→ recompile
- PAPI Performance Application Programming Interface Project, by Jack Dongarra, Kevin London, Shirley Moore, Philip Mucci, etc., at Innovative Computing Lab, CS dept., University of Tennessee
  - Standard Simple high level API and low level programmable interface
  - Supports Pentium, MMX, Pro, III/Linux, Windows; Power 3,4/AIX; etc.
  - PerfCtr kernel patch (Mikael Pettersson) → recompile

RELATED WORK – performance monitoring

- **implementing counter readers:**
- Perfmon Performance Monitoring Tool by Richard Enbody, Associate Professor Department of Computer Science and Engineering, Michigan State University.
  - For SUN Ultra-Sparc & Ppro
  - Device Driver (LKM)
- Rabbit Performance Counters Library by Don Heller, Scalable Computing Laboratory, Iowa State University
  - for Intel Pentium MMX, Pro, II, III/Linux; AMD/Linux
  - functions to access from within C
    - Cleanest of all, but still ~30 files & ~50instructions
  - LKM
- Intel's VTune Performance analyzer
  - Windows & Linux <New>
- IBM's HPM toolkit
  - Power 3,4/AIX
- Brink and Abyss Pentium 4 Performance Counter Tools For Linux, by Brinkley Sprunt, Electrical Engineering, Bucknell University
  - brink: high level perl script to read experiment/config files
  - abyss: c program to access counters
  - abyss_dev: device driver for counter access
  - EBS kernel patches: to handle PMIs

RELATED WORK – performance monitoring

- **using counter readers:**
- CASTLE Project by Margaret Martonosi and Russ Joseph, Princeton University
  - acquire Ppro counter data to model component power breakdowns
- Frank Bellosa, "Benefits of Event Driven energy Accounting in Power Sensitive Systems", 9th SIGOPS European workshop, 2000
  - Counters to show power ~ k x instr-ns/cycle (PII)
  - OS power optimizations:
    - Throttle down CPU/extend thread time for cache hit/slow down CPU core if main memory is accessed
- Andreas Weissel, Frank Bellosa, "Process Cruise Control: Event driven clock scaling for dynamic power management", CASES 2002
  - Use event counters info to scale individual thread frequencies
  - Intel Xscale / Modified Linux kernel

RELATED WORK – performance monitoring

- **using counter readers:**
- HPC Toolkit, by John Mellor-Crummey, Rob Fowler, CS Dept. Rice University
  - Uses perf counter data for profiling
  - converts raw profiling information into platform independent XML formats and produces performance metric correlations from multiple sources
  - Used in compiler optimizations
- Jennifer Anderson, et al, "Continuous Profiling: Where Have All the Cycles Gone?", ACM Transactions on Computer Systems, Vol. 15, No. 4, November 1997, pp. 357 - 390.
  - Performance analysis example – from DEC
  - Data collection by counter sampling, performance info from program level to individual instructions

RELATED WORK – real power

- CASTLE Project by Margaret Martonosi and Russ Joseph, Princeton University
  - Shunt R over Ppro power lines to measure total processor power
- John Seng, Dean Tullsen, "Effect of compiler optimizations on Pentium 4 Power consumption", 7th Annual Workshop on Interaction between Compilers and Computer Architectures, February, 2003
  - Shunt R between VRM and CPU
- Marc A. Viredaz, Deborah A. Wallach, "Power Evaluation of Itsy Version 2.3", tech. note TN-57, WRL, Compaq Computer Corp., 2000
  - similar series R to estimate battery life of itsy pocket computer

RELATED WORK – real power

- Frank Bellosa, "Benefits of Event Driven energy Accounting in Power Sensitive Systems", 9th SIGOPS European workshop, 2000
  - Crude Current measurement with DMM for Pentium II to help define per instruction powers
- Andreas Weissel, Frank Bellosa, "Process Cruise Control: Event driven clock scaling for dynamic power management", CASES 2002
  - series sense resistor added to Intel IQ 80310 evaluation platform power supply, to measure energy effect of frequency scaling
- Naehyuck Chang, Kwanho Kim, and Hyun Gyu Lee, "Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI" ISLPED 2000 & IEEE Transactions on VLSI Systems, Vol. 10, pp. 146 - 154, Apr., 2002.
  - cycle accurate energy consumption measurement based on charge transfer
  - Inserts switch caps between power supply and Processor that switch with the same clock frequency!!

RELATED WORK – power model

- **Simulation Tools:**
- WATTCH, by David Brooks and Margaret Martonosi, Princeton University, ISCA 2000
  - Architectural power simulator
  - Power Models intergrated upon SimpleScalar
- SimplePower by W. Ye, N. Vijaykrishnan, M. Kandemir, Penn-State University, and M. Irwin "The Design and Use of SimplePower: A cycle-accurate energy estimation tool", DAC, June 2000
  - Execution driven, Cycle accurate, RTL power estimation
  - Emulates 5 stage pipe with SimpleScalar's Integer ISA

RELATED WORK – power model

- **Power Modeling:**
- R. Joseph and M. Martonosi. "Run-Time Power Estimation in High Performance Microprocessors", International Symposium on Low Power Electronics and Design, 2001
  - complete CASTLE Project: Collects Ppro counter data and models component power breakdowns verifying against measured total power
  - Also Wattch simulation vs. counter approximation for SimpleScalar architecture
- Russ Joseph, David Brooks, and Margaret Martonosi, "Live, Runtime Power Measurements as a Foundation for Evaluating Power/Performance Tradeoffs" Workshop on Complexity Effectice Design (WCED, held in conjunction with ISCA-28), 2001
  - Evaluate power vs. performance by measuring total power and acquiring performance data from counters – i.e. Cache hit rate, branch prediction, bitline activity

RELATED WORK – power model

- H. Zeng, X. Fan, C. Ellis, A. Lebeck, and A. Vahdat, "ECOSystem: Managing Energy as a First Class Operating System Resource", Proceedings of ASPLOS X, Oct. 2002
  - Uses Currentcy Model (Fixed Power & Time budget for a task) for OS level energy management for battery life
  - ECOsystem is the Linux OS implementation <No counters>
  - Considers CPU ON/OFF → could do better with Power model
- H. Zeng, C. Ellis, A. Lebeck, A. Vahdat , "Currentcy: Unifying Policies for Resource Management", USENIX 2003 Annual Technical Conference
  - Detailed description of currency (OS scheduling, etc.)
- Flinn J., Satyanarayanan, M., "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications", Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications February, 1999
  - Maps Energy ⇔ Program structure (Power Profiling – Energy efficient SW design)
  - DMM gets energy for machine
  - kernel modification (system monitor) gets PIDs for processes and identifies procedures for profiling offline

RELATED WORK – power model

- V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization", International Conference on Computer-Aided Design & IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1994
  - PIONEER WORK in Power Measurement/Modeling
  - Measure current drawn by an Intel 486DX2 Processor and DRAM
  - Generate Energy cost table for instructions
  - Identify inter-instructions effects: circuit state overhead, resource constraint effect, cache miss effects
  - there are 1 million like this: modeling SW energy, I won't put here
- Lee, A. Ermedahl, and S. Min. "An accurate instruction-level energy consumption model for embedded risc processors" ACM SIGPLAN Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES'01), Jun 2001
  - Derives energy consumption for instructions rather than functional units for RISC ARM7TDMI processor
  - Uses their cycle-accurate power measurement scheme
  - Black box approach (similar to F. Bellosa) with linear regression

RELATED WORK – power model

- J. Russell and M.F. Jacome, "Software Power Estimation and Optimization for High Performance, 32-bit Embedded Processors," Proc. of ICCD '98
  - Estimates SW energy for i960 family 32 bit embedded RISC processors
  - Uses digitizing oscilloscope/series Resistor over processor power lines for measurement
  - Uses const Pest for processor power and estimates energy based on runtime (won't work with clock gating!)
- J. Haid, G. Kafer, et al, "Run-Time Energy Estimation in System-On-a-Chip Designs", ASP-DAC 2003
  - Proposes a coprocessor for runtime energy estimation for SoC
  - Defines similar event counters in coprocessor and uses power macro-models
- M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno, and A. Sangiovanni-Vincentelli. "Efficient power estimation techniques for hw/sw systems", IEEE Proc. VOLTA'99 International Workshop on Low Power Design, pages 191--199, March 1999.
  - Power estimation for HW/SW SoC designs
  - RTL HW simulator and Instruction Set simulator using instruction level power models

RELATED WORK – power model

- M. Huang, J. Renau, and J. Torrellas. "Profile-based energy reduction in high-performance processors", In 4th Workshop on Feedback-Directed and Dynamic Optimization, December 2001
  - Use profiling to determine when to activate/deactivate low power methods –i.e. DVS, clock gating, etc.
  - Use energy statistics (power breakdowns) from performance counters for profiling (SIM)
- I. Kadayif , T. Chinoda , M. Kandemir , N. Vijaykirsnan , M. J. Irwin , A. Sivasubramaniam, "vEC: virtual energy counters", Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, 2001
  - Uses Perfmon library for UltraSPARC to read SPARC HW perf counters related to memory
  - Converts readings to power using analytical memory energy model
  - estimates memory system energy consumption

RELATED WORK – power model

- Luca Benini et al
  - "System-level power estimation and optimization", Proceedings 1998 international symposium on Low power electronics and design
  - "System-level power optimization: techniques and tools", Proceedings of international symposium on Low power electronics and design, 1999
    - Tutorial on power conscious system level design
    - Memory optimizations, Hardware software partitioning, instruction level power optimizations, DVS, DPM (allow components to sleep)
  - "Supporting system-level power exploration for DSP applications", Proceedings of the 10th Great Lakes Symposium on VLSI, 2000
    - Modified ARM simulator for instruction level power estimation

RELATED WORK – thermal model

- K. Skadron, T. Abdelzaher, and M. R. Stan. "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management", In Proc. HPCA-8, pages 17--28, Feb. 2002.
  - Single degree component based thermal R-C model for MIPS R10000 scaled to 0.18Um
  - Only die → heatsink thermal conduction, with const. heatsink and Si properties only
  - Power/Thermal Simulation using Wattch for verification of DTM with PID controller
- Sabry, M.-N.; Bontemps, A.; Aubert, V.; Vahrmann, R, "Realistic and efficient simulation of electro-thermal effects in VLSI circuits", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , Volume: 5 Issue: 3 , Sep 1997
  - Transistor level with interdevice thermal resistances
- Szekely, V.; Poppe, A.; Pahi, A.; Csendes, A.; Hajas, G.; Rencz, M, "Electro-thermal and logi-thermal simulation of VLSI designs", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , Volume: 5 Issue: 3 , Sep 1997
  - LOGITHERM simulator module for gate level thermal simulation, by thermal characterization of logic gates

RELATED WORK – thermal model

- COSMOS/FloWorks by NIKA
  - fluid flow and thermal analysis program
  - Heat flow computation based on mesh analysis
- A. Dhodapkar, C. H. Lim, G. Cai, and W. R. Daasch. "TEMPEST: A thermal enabled multi-model power / performance estimator", Proceedings of Workshop on Power-Aware Computer Systems, Nov. 2000.
  - Thermally enabled architectural simulator based on SimpleScalar
  - Single R,C for the whole processor → packaging oriented
- D. Brooks and M. Martonosi. *Dynamic thermal management for high-performance microprocessors*. In Proceedings of the Seventh International Symposium on High-Performance Computer Architecture, pages 171--82, Jan. 2001.
  - Discusses Microarchitectural and scaling DTM mechanisms
  - Uses moving average of power for ~100K cycles of Wattch simulation as a proxy for temperature to detect thermal emergencies for DTM triggering

RELATED WORK – thermal model

- Thermal Monitoring, "Intel Architecture SW developer's Manual vol. 3"
  - Catastrophic shutdown detector
    - thermal diode resets stop clock duty cycle
  - Automatic Thermal monitor
    - Internally modulate stop clock duty cycle
  - Software controlled clock modulation
    - SW modulates stop clock duty cycle
- Kevin Skadron et al, "Temperature aware Microarchitecture", 30th ISCA, 2003
  - HotSpot: architecture level thermal simulator built upon Wattch
  - Uses multiple degree thermal R-C model for die, packaging, heatsink and convection to ambient
  - More realistic area estimates based on Alpha 21364