

Modeling and Analyzing CPU Power and Performance: Metrics, Methods, and Abstractions

Pradip Bose
Margaret Martonosi
David Brooks



Moore's Law & Power Dissipation...



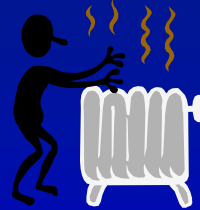
Moore's Law:

- The Good News: 2X Transistor counts every 18 months
- The Bad News: To get the performance improvements we're accustomed to, CPU Power consumption will increase exponentially too...

(Graphs courtesy of Fred Pollack, Intel)

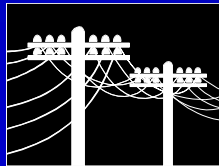
Why worry about power dissipation?

Battery life



Thermal issues: affect cooling, packaging, reliability, timing

Environment

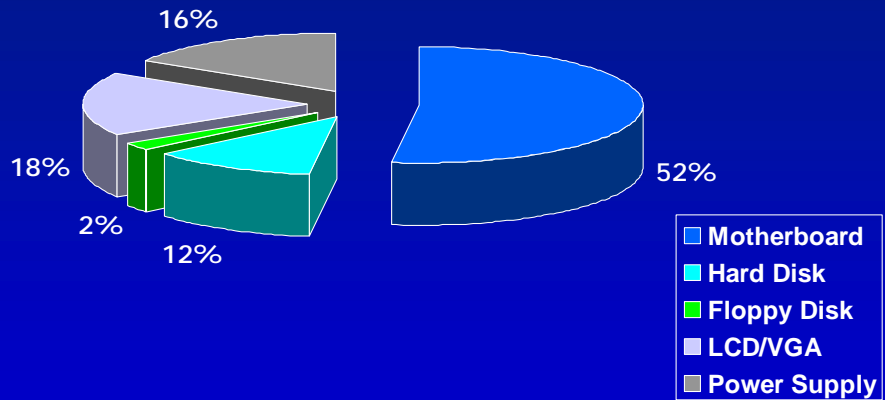


Hitting the wall...

- **Battery technology**
 - Linear improvements, nowhere near the exponential power increases we've seen
- **Cooling techniques**
 - Air-cooled is reaching limits
 - Fans often undesirable (noise, weight, expense)
 - \$1 per chip per Watt when operating in the >40W realm
 - Water-cooled !!?
- **Environment**
 - US EPA: 10% of current electricity usage in US is directly due to desktop computers
 - Increasing fast. And doesn't count embedded systems, Printers, UPS backup?
- **Past:**
 - Power important for laptops, cell phones
- **Present:**
 - Power a Critical, Universal design constraint even for very high-end chips
 - Circuits and process scaling can no longer solve all power problems.
 - *SYSTEMS* must also be power-aware
 - Architecture, OS, compilers



Notebook Power Usage Stats

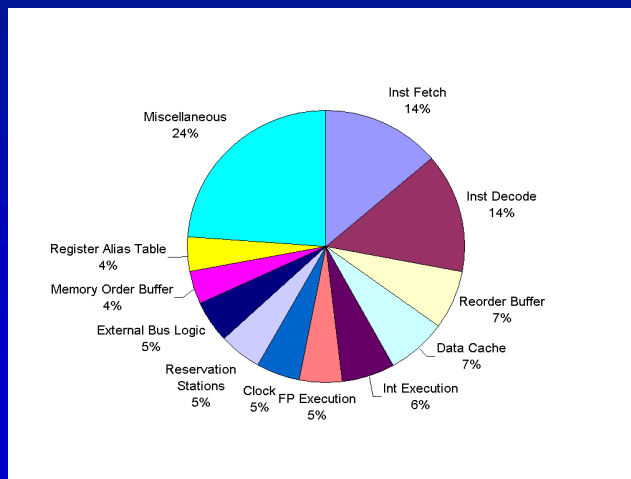


1995 5V Notebook PC

From Roy, 1997

Processor Power Breakdown

- Pentium Pro Breakdown of Power by unit



Outline

- Power basics
- Power and energy metrics
- Modeling abstractions
- Measuring power in real machines
- Validation
- Trends and conclusions

Power: The Basics

- Dynamic power vs. Static power vs. short-circuit power
 - “switching” power
 - “leakage” power
 - Dynamic power dominates, but static power increasing in importance
 - Trends in each
- Static power: steady, per-cycle energy cost
- Dynamic power: power dissipation due to capacitance charging at transitions from 0->1 and 1->0
- Short-circuit power: power due to brief short-circuit current during transitions.
- Mostly focus on dynamic, but recent work on others

Dynamic CMOS Power dissipation

Capacitance:
Function of wire
length, transistor size

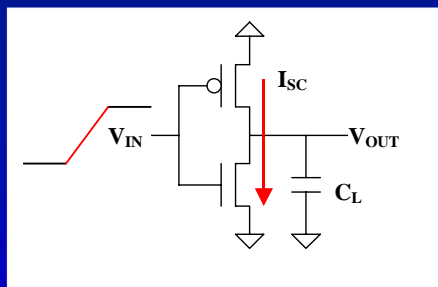
Supply Voltage:
Has been dropping
with successive fab
generations

$$\text{Power} \sim \frac{1}{2} CV^2Af$$

Activity factor:
How often, on average,
do wires switch?

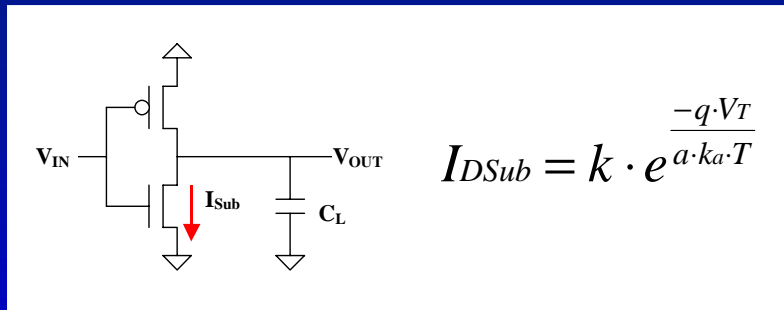
Clock frequency:
Increasing...

Short-Circuit Power Dissipation



- Short-Circuit Current caused by finite-slope input signals
- Direct Current Path between VDD and GND when both NMOS and PMOS transistors are conducting

Leakage Power



- Subthreshold currents grow exponentially with increases in temperature, decreases in threshold voltage

Power and Energy Metrics

Metrics Overview: A Microarchitect's View

- Performance metrics:
 - delay (execution time) per instruction; MIPS
 - CPI (cycles per instr): abstracts out the MHz
 - SPEC (int or fp); TPM: factors in benchmark, MHz
- energy and power metrics:
 - joules (J) and watts (W)
- joint metric possibilities (perf and power)
 - watts (W): for ultra LP processors; also, thermal issues
 - MIPS/W or SPEC/W ~ energy per instruction
 - $\text{CPI} * \text{W}$: equivalent inverse metric
 - MIPS²/W or SPEC²/W ~ energy*delay (EDP)
 - MIPS³/W or SPEC³/W ~ energy*(delay)² (ED²P)

Energy vs. Power

- Energy metrics (like SPEC/W):
 - compare battery life expectations; *given workload*
 - compare energy efficiencies: processors that use constant voltage, frequency or capacitance scaling to reduce power
- Power metrics (like W):
 - max power => package design, cost, reliability
 - average power => avg electric bill, battery life
- ED²P metrics (like SPEC³/W or CPI³ * W):
 - compare pwr-perf efficiencies: processors that use voltage scaling as the primary method of power reduction/control

E vs. EDP vs. ED²P

- Power $\sim C.V^2.f \sim f$ (fixed voltage, design)
 $\sim C$ (fixed voltage, freq)
- Perf $\sim f$ (fixed voltage and design)
 $\sim IPC$ (fixed voltage, freq)

*So, across processors that use either frequency scaling or capacitance scaling, e.g. via clock gating or adaptive microarch techniques, multiple clocks, etc., MIPS/W or SPEC/W is the right metric to compare energy efficiencies. (Also, CPI * W)*

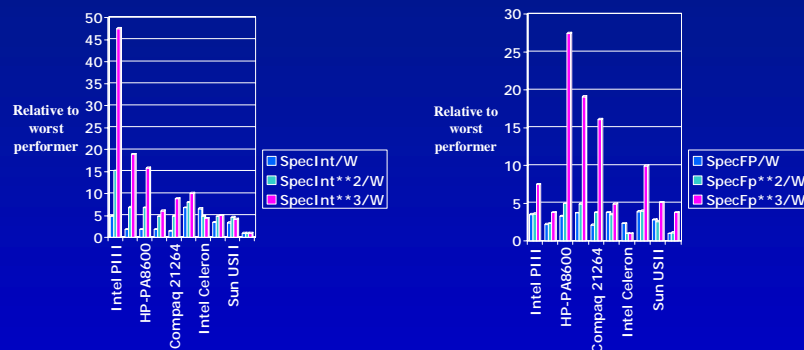
E vs. EDP vs. ED²P

- Power $\sim CV^2.f \sim V^3$ (fixed microarch/design)
- Performance $\sim f \sim V$ (fixed microarch/design)
- (For the 1-3 volt range, f varies approx. linearly with V)
- So, across processors that use voltage scaling as the primary method of power control (e.g. Transmeta),
■ (perf)³ / power, or MIPS³ / W or SPEC³ /W is a fair metric to compare energy efficiencies.
- This is an ED² P metric. We could also use: (CPI)³ * W for a given application

E vs. EDP vs. ED2P

- EDP metrics like MIPS2/W or SPEC2/W cannot be applied across an arbitrary set of processors to yield fair comparisons of efficiency; although, EDP could still be a meaningful optimization vehicle for a given processor or family of processors.
- Our view: use either E or ED2P type metrics, depending on the class of processors being compared (i.e. fixed voltage, variable cap/freq - E metrics; and, variable voltage/freq designs - ED2P metrics)
 - caveat: leakage power control techniques in future processors, that use lots of low-Vt transistors may require some rethinking of metrics

Metrics Comparison

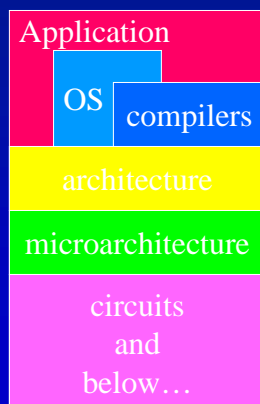


(Brooks et al., IEEE Micro, Nov/Dec 2000)

- Note:
 - > at the low end, E metrics like SpecInt/W appear to be fair
 - > at the highest end, ED2P metrics like $(\text{SpecInt})^3/W$ seem to do the job
 - > perhaps at the midrange, EDP metrics like $(\text{SpecInt})^2/W$ are appropriate?

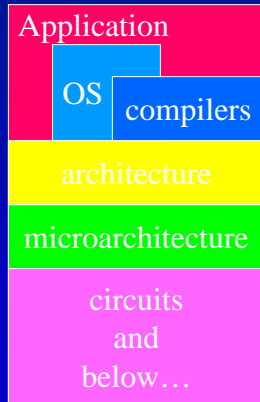
Modeling & Abstractions

What can architects & systems people do to help?



- Micro-Architecture & Architecture
 - Shrink structures
 - Shorten wires
 - Reduce activity factors
 - Improve instruction-level control
- Compilers
 - Reduce wasted work: "standard" operations
 - More aggressive register allocation and cache optimization
 - Trade off parallelism against clock frequency
- Operating Systems
 - Natural, since OS is traditional resource manager
 - Equal energy scheduling
 - Battery-aware or thermally-aware adaptation

What do architects & systems people need to have, in order to help?



- Better observability and control of power characteristics
 - Ability to see current power, thermal status
 - Temperature sensors on-chip
 - Battery meters
 - Ability to control power dissipation
 - Turn units on/off
 - Techniques to impact leakage
- Abstractions for efficient modeling/estimation of power consumption

Analysis Abstraction Levels

Abstraction Level	Analysis Capacity	Analysis Accuracy	Analysis Speed	Analysis Resources	Energy Savings
Application	Most	Worst	Fastest	Least	Most
Behavioral	↑	↓	↑	↓	↑
Architectural (RTL)					
Logic (Gate)					
Transistor (Circuit)	Least	Best	Slowest	Most	Least

Power/Performance abstractions at different levels of this hierarchy...

- Low-level:
 - Hspice
 - PowerMill
- Medium-Level:
 - RTL Models
- Architecture-level:
 - PennState SimplePower
 - Intel Tempest
 - Princeton Wattch
 - IBM PowerTimer

Low-level models: Hspice

- Extracted netlists from circuit/layout descriptions
 - Diffusion, gate, and wiring capacitance is modeled
- Analog simulation performed
 - Detailed device models used
 - Large systems of equations are solved
 - Can estimate dynamic and leakage power dissipation within a few percent
 - Slow, only practical for 10-100K transistors
- PowerMill (Synopsys) is similar but about 10x faster

Medium-level models: RTL

- Logic simulation obtains switching events for every signal
- Structural VHDL or verilog with zero or unit-delay timing models
- Capacitance estimates performed
 - Device Capacitance
 - Gate sizing estimates performed, similar to synthesis
 - Wiring Capacitance
 - Wire load estimates performed, similar to placement and routing
- Switching event and capacitance estimates provide dynamic power estimates

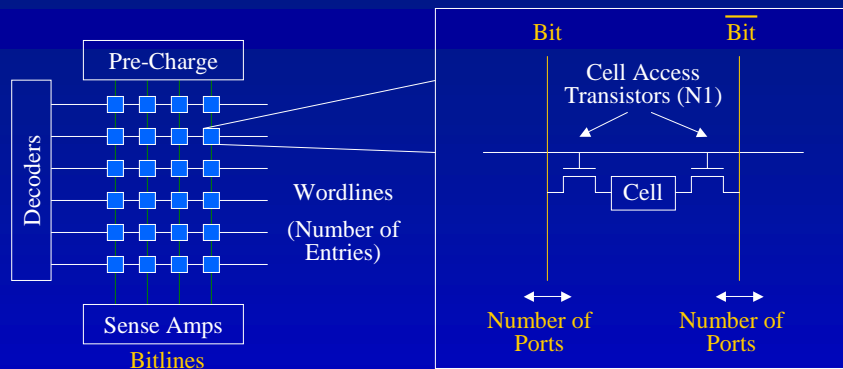
Architecture level models

- Examples:
 - SimplePower, Tempest, Wattch, PowerTimer...
- Components of a "good" Arch. Level power model
 - Capacitance model
 - Circuit design styles
 - Clock gating styles & Unit usage statistics
 - Signal transition statistics

Modeling Capacitance

- Requires modeling wire length and estimating transistor sizes
- Related to RC Delay analysis for speed along critical path
 - But capacitance estimates require summing up all wire lengths, rather than only an accurate estimate of the longest one.

Register File: Example of Capacitance Analysis



$$C_{wordline} = C_{diffcapWordlineDriver} + NumberBitlines * C_{gatecapN1} +$$

$$Wordlinelength * C_{metal}$$

$$C_{bitline} = C_{diffcapPchg} + NumberWordlines * C_{diffcapN1}$$

$$+ Bitlinelength * C_{metal}$$

Register File Model: Validation

<i>Error Rates</i>	<i>Gate</i>	<i>Diff</i>	<i>InterConn.</i>	<i>Total</i>
Wordline(r)	1.11	0.79	15.06	8.02
Wordline(w)	-6.37	0.79	-10.68	-7.99
Bitline(r)	2.82	-10.58	-19.59	-10.91
Bitline(w)	-10.96	-10.60	7.98	-5.96

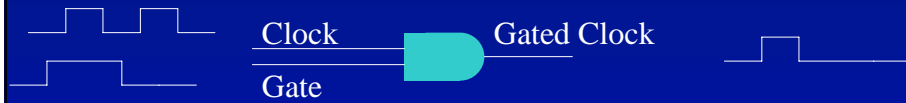
(Numbers in Percent)

- Validated against a register file schematic used in Intel's Merced design
- Compared capacitance values with estimates from a layout-level Intel tool
- Interconnect capacitance had largest errors
 - Model currently neglects poly connections
 - Differences in wire lengths -- difficult to tell wire distances of schematic nodes

Accounting for Different Circuit Design Styles

- RTL and Architectural level power estimation requires the tool/user to perform circuit design style assumptions
 - Static vs. Dynamic logic
 - Single vs. Double-ended bitlines in register files/caches
 - Sense Amp designs
 - Transistor and buffer sizings
- Generic solutions are difficult because many styles are popular
- Within individual companies, circuit design styles may be fixed

Clock Gating: What, why, when?



- Dynamic Power is dissipated on clock transitions
- Gating off clock lines when they are unneeded reduces activity factor
- But putting extra gate delays into clock lines increases clock skew
- End results:
 - Clock gating complicates design analysis but saves power. Used in cases where power is crucial.

Signal Transition Statistics

- Dynamic power is proportional to switching
- How to collect signal transition statistics in architectural-level simulation?
 - Many signals are available, but do we want to use all of them?
 - One solution (register file):
 - Collect statistics on the important ones (bitlines)
 - Infer where possible (wordlines)
 - Assign probabilities for less important ones (decoders)
 - Use Controllability and Observability notions from testing community?

Power Modeling at Architecture Level

- Previous academic research has either:
 - Calculated power within individual units: ie cache
 - Calculated abstract metrics instead of power
 - eg “needless speculative work saved per pipe stage”
- What is needed now?
 - A single common power metric for comparing different techniques
 - Reasonable accuracy
 - Flexible/modular enough to explore a design space
 - Fast enough to simulate real benchmarks
 - Facilitate early experiments: before HDL or circuits...

SimplePower

- Vijaykrishnan, et al. ISCA 2000
- Models datapath energy in 5-stage pipelined RISC datapath
- Table-lookup based power models for memory and functional units
- Transition sensitive: table lookups are done based on input bits and output bits for unit being considered
- Change size of units => supply a new lookup table

TEM²P²EST

- Thermal Enabled Multi-Model Power/Performance Estimator: Dhodapkar, Lim, Cai, and Daasch
- Empirical Mode
 - Used for synthesizable logic blocks
 - Used for Clock distribution/interconnection
- Analytical Mode
 - Used for regular structures
 - Allows time-delay model extensions
- Temperature Model
 - Simple model links power to temperature

Wattch: An Overview



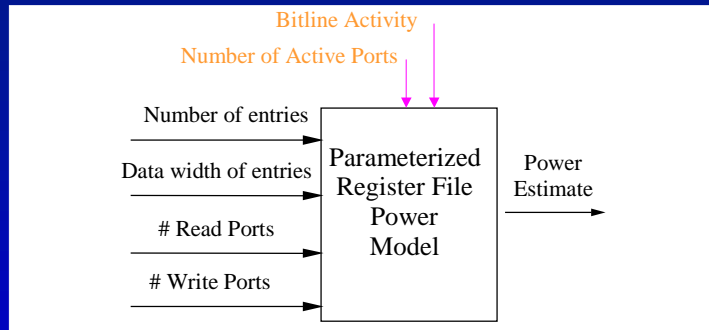
Wattch's Design Goals

- Flexibility
- Planning-stage info
- Speed
- Modularity
- Reasonable accuracy

Overview of Features

- Parameterized models for different CPU units
 - Can vary size or design style as needed
- Abstract signal transition models for speed
 - Can select different conditional clocking and input transition models as needed
- Based on SimpleScalar
- Modular: Can add new models for new units studied

Modeling Units at Architectural Level



Modeling Capacitance

- Models depend on structure, bitwidth, design style, etc.
- E.g., may model capacitance of a register file with bitwidth & number of ports as input parameters

Modeling Activity Factor

- Use cycle-level simulator to determine number and type of accesses
 - reads, writes, how many ports
- Abstract model of bitline activity

One Cycle in Wattch

	Fetch	Dispatch	Issue/Execute	Writeback/Commit
Power (Units Accessed)	<ul style="list-style-type: none"> I-cache Bpred 	<ul style="list-style-type: none"> Rename Table Inst. Window Reg. File 	<ul style="list-style-type: none"> Inst. Window Reg File ALU D-Cache Load/St Q 	<ul style="list-style-type: none"> Result Bus Reg File Bpred
Performance	<ul style="list-style-type: none"> Cache Hit? Bpred Lookup? 	<ul style="list-style-type: none"> Inst. Window Full? 	<ul style="list-style-type: none"> Dependencies Satisfied? Resources? 	<ul style="list-style-type: none"> Commit Bandwidth?

- On each cycle:
 - determine which units are accessed
 - model execution time issues
 - model per-unit energy/power based on which units used and how many ports.

R10K die size 16.6mm x 17.9mm

Units Modeled by Wattch

- **Array Structures**
 - Caches, Reg Files, Map/Bpred tables
- **Content-Addressable Memories (CAMs)**
 - TLBs, Issue Queue, Reorder Buffer
- **Complex combinational blocks**
 - ALUs, Dependency Check
- **Clocking network**
 - Global Clock Drivers, Local Buffers

Wattch accuracy

Relative Wattch estimates track well even in cases where absolute accuracy falls short.

Hardware Structure	Intel Data	Wattch
Instruction Fetch	22%	21%
Register Alias Table	6%	5%
Reservation Stations	8%	9%
Reorder Buffer	11%	12%
Integer Exec. Unit	15%	15%
Data Cache Unit	11%	11%
Memory Order Buffer	6%	5%
Floating Point Exec. Unit	8%	8%
Global Clock	8%	10%
Branch Target Buffer	5%	4%

Typically 10-15% relative accuracy as compared to low-level industry data.

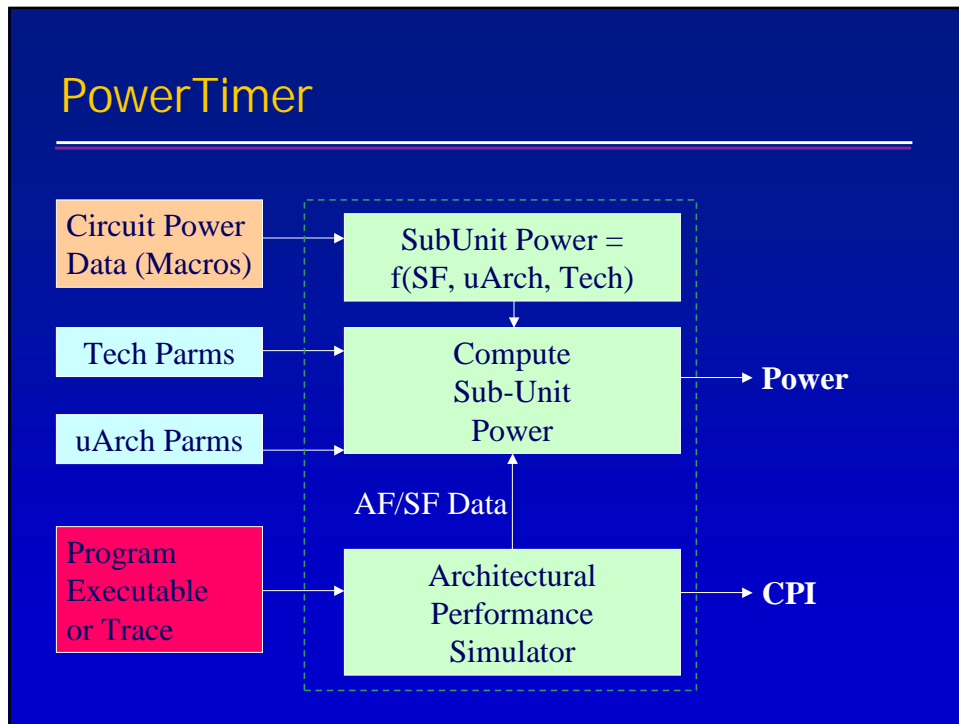
Wattch Simulation Speed

- Roughly 80K instructions per second (PII-450 host)
- ~30% overhead compared to performance simulation alone
 - Could be decreased if power estimates are not computed every cycle
- Many orders of magnitude faster than lower-level approaches
 - For example, PowerMill takes ~1hour to simulate 100 test vectors on a 64-bit adder

Wattch: Summary

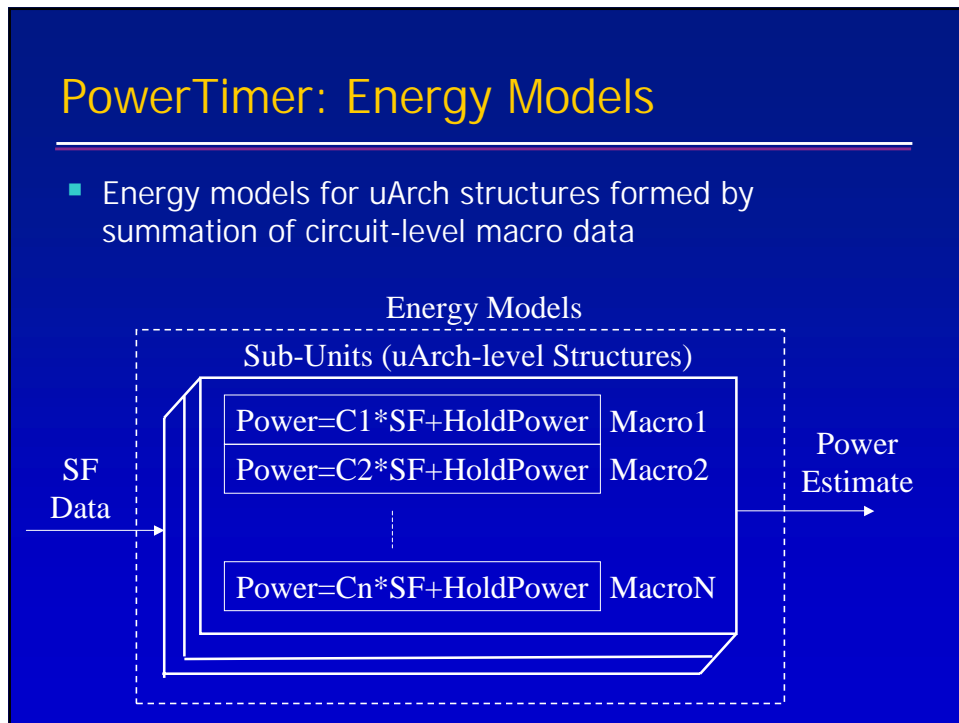
- A preliminary but useful step towards providing modular, flexible architecture-level models with reasonable accuracy
- Future Work:
 - User selectable circuit styles (high-performance, low-power, etc)
 - Update models as technologies change

PowerTimer

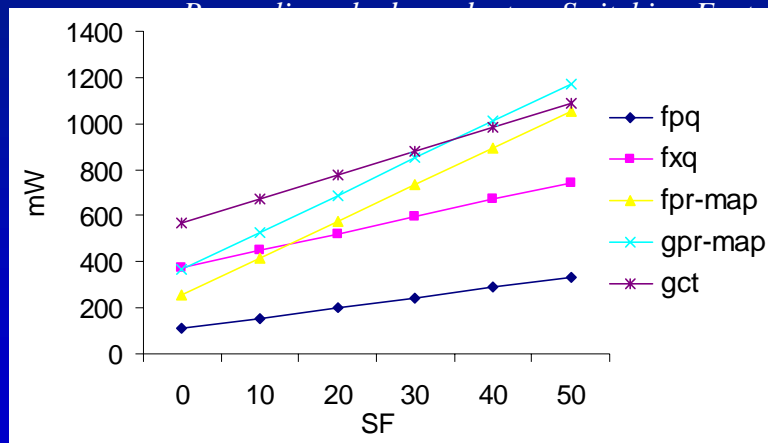


PowerTimer: Energy Models

- Energy models for uArch structures formed by summation of circuit-level macro data



PowerTimer: Power models $f(SF)$



At 0% SF, Power = Clock Power (significant without clock gating)

Comparing Arch. Level power models: Flexibility

- Flexibility necessary for certain studies
 - Resource tradeoff analysis
 - Modeling different architectures
- Wattch provides fully-parameterizable power models
 - Within this methodology, circuit design styles could also be studied
- PowerTimer scales power models in a user-defined manner for individual sub-units
 - Constrained to structures and circuit-styles currently in the library
- SimplePower provides parameterizable cache structures

Comparing Arch. Level power models: Speed

- Performance simulation is slow enough!
- Wattch's per-cycle power estimates: roughly 30% overhead
 - Post-processing (per-program power estimates) would be much faster (minimal overhead)
- PowerTimer has no overhead (currently all post-processed based on already existing stats)
- SimplePower may have more performance overhead because of table-lookups

Comparing Arch. Level power models: Accuracy

- Wattch provides excellent *relative* accuracy
 - Underestimates full chip power (some units not modeled, etc)
- PowerTimer models based on circuit-level power analysis
 - Inaccuracy is introduced in SF/AF and scaling assumptions
- SimplePower should provide highest accuracy
 - Static core (only caches are parameterized)
 - Table lookups track all transitions

Measuring power in real machines...

Measuring power (vs. modeling it)

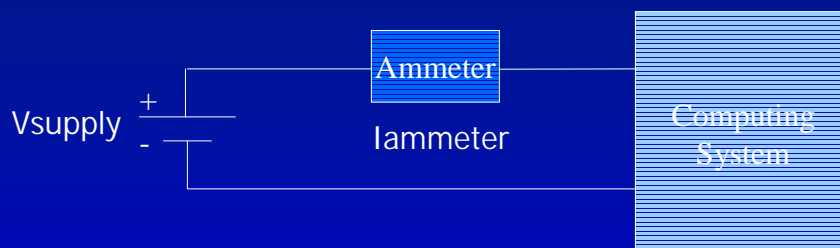
- First part of talk discussed power modeling.
- What about power measurement?
- Challenges:
 - Difficult to get enough motherboard information to measure the power you want to.
 - Even harder (ie impossible) to break down on-chip power into a pie chart of different contributors
 - Difficult to ascribe power peaks and valleys to particular software behavior or program constructs.

A few typical meter-based setups #1: Voltage-drops with transceivers ...



- $\text{Power} = V_{\text{supply}} * V_{\text{sense}}/R_{\text{sense}}$
- Itsy Study:
 - 0.02Ω R_{sense}
 - 5000 Samples/sec
 - Estimated Error: ± 0.005 Watts (~ 1 W measured)

Typical setups #2: Ammeter on incoming power supply lines

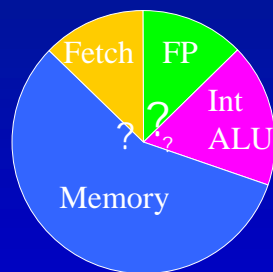


- $\text{Power} = V_{\text{supply}} * I_{\text{ammeter}}$
- Our equipment, typical ranges...

Limitations to meter-based Approaches

- Can only measure what actually exists
- Difficult to ascribe power to particular parts of the code
- Difficult to get very fine-grained readings due to power-supply capacitors etc.
- Difficult to “pie chart” power into which units are dissipating it

Monitoring power on existing CPUs: Counter-Based



- Say you wish to measure power consumption for a program running on an existing CPU?
 - Surprisingly difficult to do
 - Ammeter on power cord is difficult to synchronize with application runtimes
- Say you want to produce a **pie chart** of measured power dissipation per-unit for this program running an existing CPU?
 - Nearly impossible to do directly

CASTLE: Measuring Power Data from Event Counts

Basic idea:

- Most (all?) high-end CPUs have a bank of hardware performance counters
- Performance counters track a variety of program events
 - Cache misses
 - Branch mispredicts...
- If these events are also the main power dissipators, then we can hope these counters can also help power measurement
- Estimate power by weighting and superimposing event counts appropriately

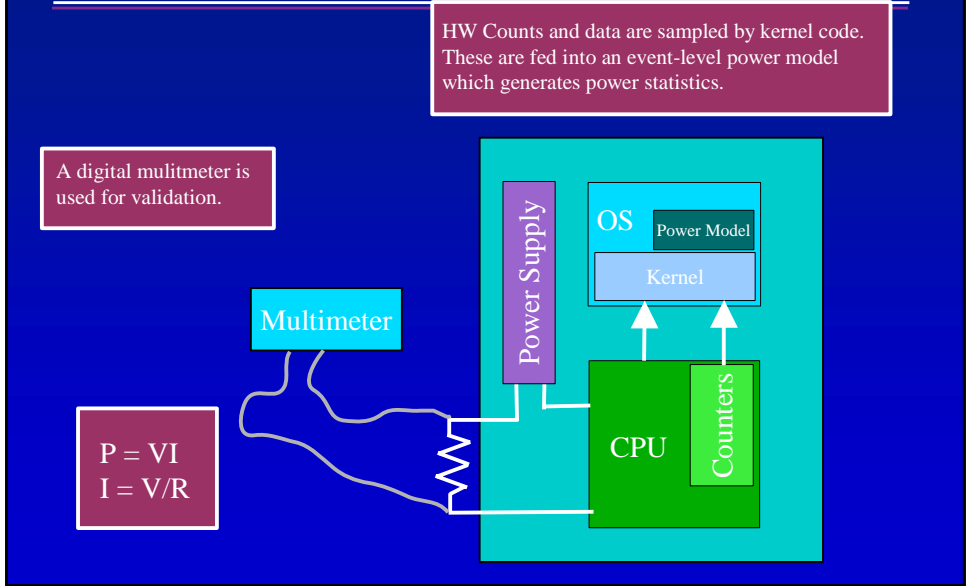
CASTLE: Details & Methodology

$$\begin{array}{c} \text{Appls} \\ \left[\begin{array}{ccc} C_{11} & \dots & C_{1M} \\ \vdots & & \vdots \\ C_{1N} & \dots & C_{NM} \end{array} \right] \left[\begin{array}{c} W_1 \\ \vdots \\ W_M \end{array} \right] = \left[\begin{array}{c} P_1 \\ \vdots \\ P_M \end{array} \right] \end{array}$$

Counter Values

- Gather M counts for N training applications
- Compute weights using least-squares error
- Use these weights (W_1 - W_M) to estimate power on other apps
- Consider accuracy of power estimates compared to other power measurements

General Approach and Validation (Not To Scale)



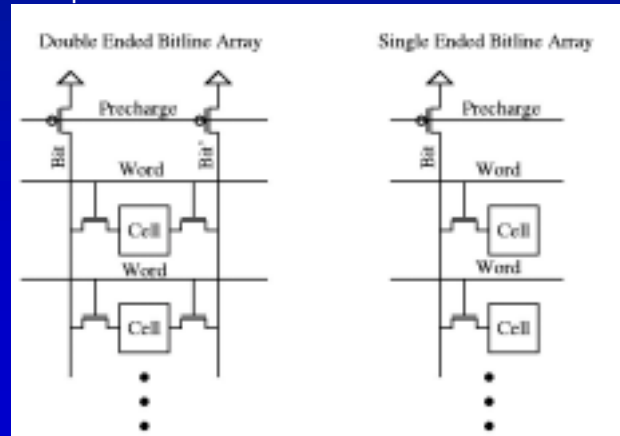
Power Model

$$\text{power} = \overset{\text{usage}}{\downarrow} \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{bmatrix} + \overset{\text{values}}{\downarrow} \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_k \end{bmatrix}$$

<p>Value Insensitive Power</p> <p>Constants:</p> <p>C_i maximum power of value insensitive component i</p> <p>Variables:</p> <p>u_i average utilization of value insensitive component i</p>	<p>Value Sensitive Power</p> <p>Constants:</p> <p>B_j maximum power of value sensitive component j</p> <p>Variables:</p> <p>v_j average activity of value sensitive component j</p>
--	---

Assumptions and Caveats

- Leakage power is not handled currently
- Only single-ended bitline power is value dependent



Performance Counters

- Are debugging aids found in most high performance processors

Processor	Available Events	Simultaneous Events
PowerPC 604	44	2
PowerPC 603e	116	4
POWER3	238	8
Pentium Pro	77	2

- Track performance relative events, including:
 - cache miss
 - instruction retirement
 - branch misprediction
- Give insight into utilization (u_1, u_2, \dots, u_n)
- Are not useful in calculating bitline values (v_1, v_2, \dots, v_k)

Estimation Problems and Solutions

Problem One: Abundance of performance relevant counts.
Lack of power relevant counts.

Solution: Approximate missing counts with available ones.

Problem Two: Insufficient counters to observe all resources.

Solution: Sample events and time-divide counters.

Problem Three: Cannot directly monitor bitline signals.

Solution: Assume register file typifies bitline data. Sample it.

Validations

- Used Wattch [Brooks-ISCA 2000], an architectural level simulator to model a four issue 660 MHz Alpha processor.
- Relied on Wattch models to compute maximum component and bitline power.
- Assumed a 10 millisecond OS time quanta for sampling.
- Assumed 2 hardware counters and used a total of 10 events:
instructions decoded, instructions executed, instructions retired
floating point operations executed, branches retired, branches
decoded
L1 instruction cache accesses, L1 data cache accesses, L2 unified
cache accesses, main memory requests

Problem One - Counter Relevance

Since counter events don't directly observe all power relevant events, we apply platform specific heuristics.

Example 1

register rename lookups = 2 x instructions decoded

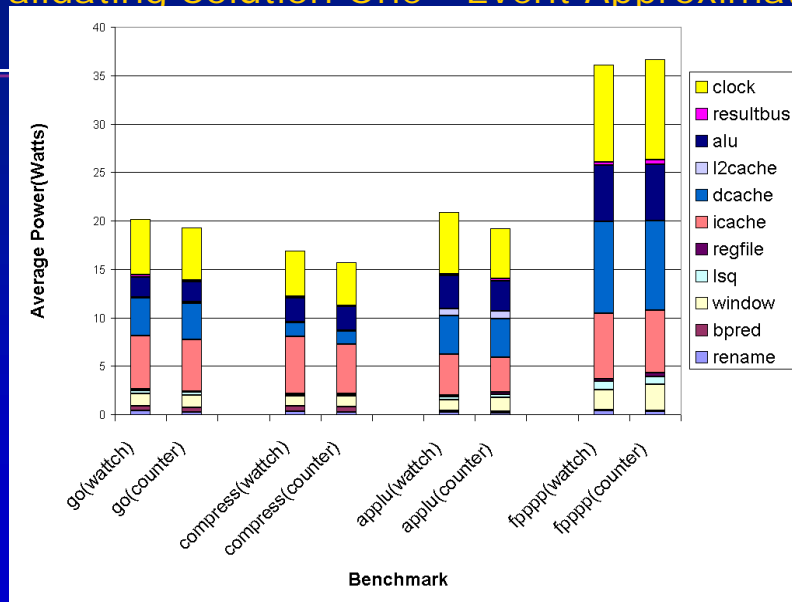
Example 2

integer alu accesses = instructions executed - floating point operations

Example 3

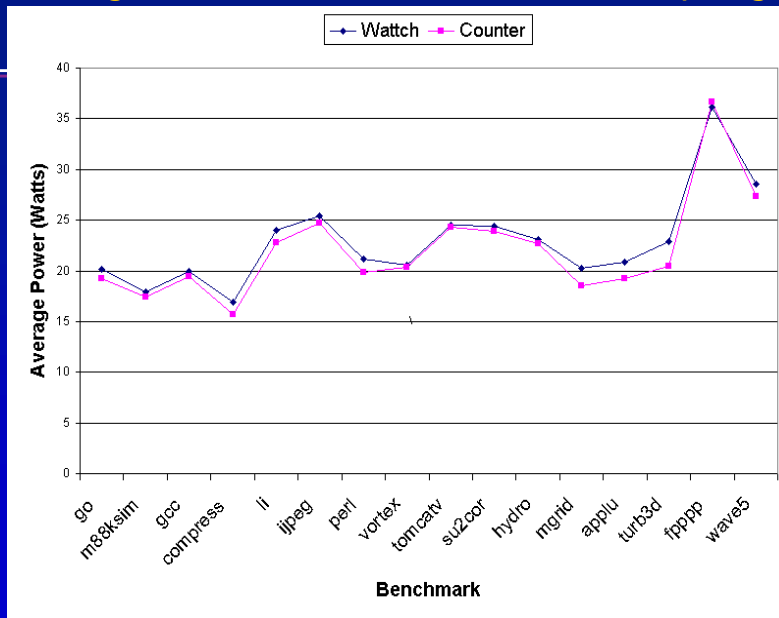
physical register accesses = 5 x instructions executed

Validating Solution One - Event Approximation



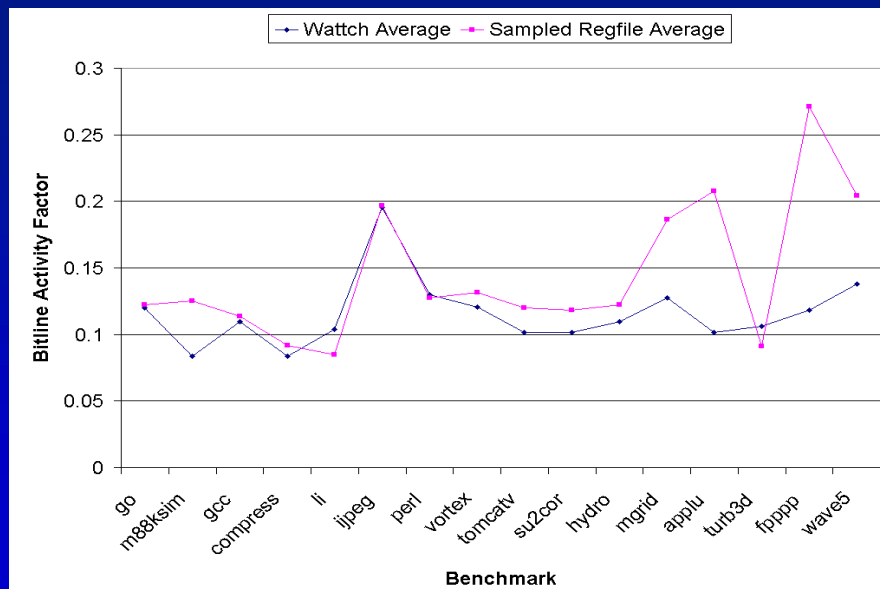
Component power is typically accurate within 5-10%.

Validating Solution Two – Counter Sampling



Average error is within 5%.

Validating Solution Three - Register File Sampling



Implementation on Pentium Pro



System Configuration: Intel Pentium Pro 200MHz, 128 MB RAM, 2 GB
Internal SCSI HD, Linux 2.2.16

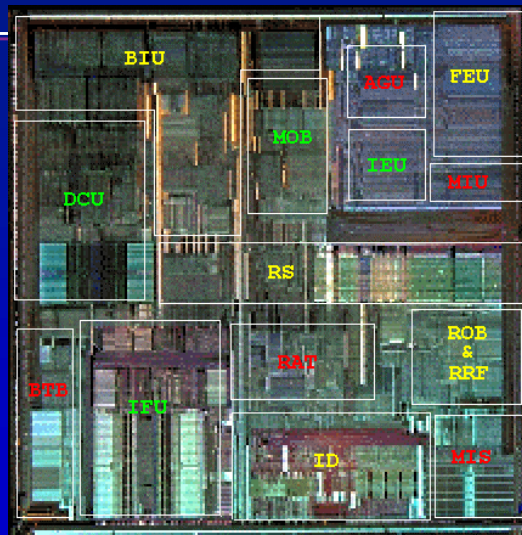
Estimation Techniques

Utilization Approximations - Fully Implemented

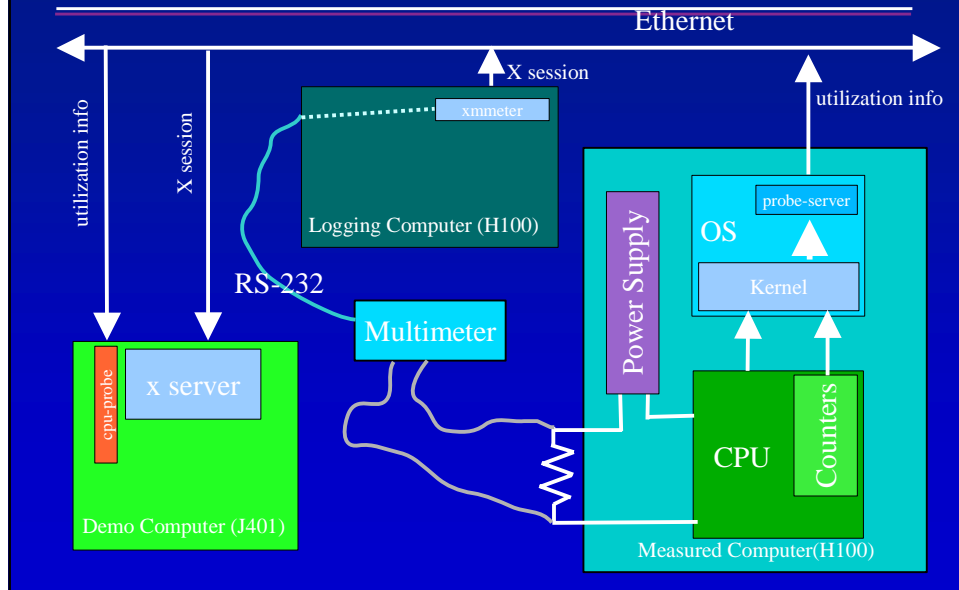
Data Value Estimates - Assume constants

Validation: HP 34401A Multimeter with automatic data logging

Pentium Pro Power Model



Demo Setup (Not To Scale)



Other Measurement Techniques

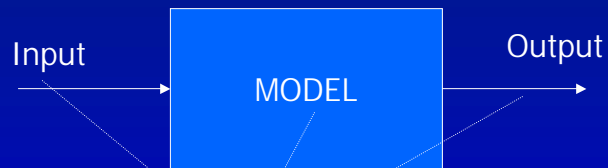
- Thermal sensors
 - [Sanchez et al. 1995]
 - PowerPC includes thermal sensor and allows for real-time responses to thermal emergencies.
 - Eg. Reduce instruction fetch rate

Comparing different measurement/modeling techniques

- Choice of technique depends on experiment to be done
- Measuring different software on unchanging platform
 - Real platform probably better
- Measuring impact of hardware design changes
 - Need simulations, since real hardware doesn't exist...

Validation

Validation



Need to ensure integrity at all 3 stages

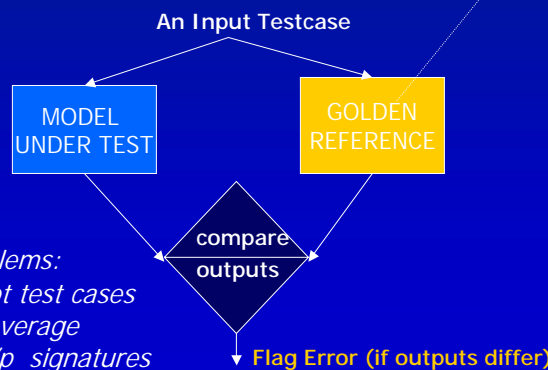
Input Validation: making sure that the input, e.g. trace, is representative of the workloads of interest

Model Validation: ensuring that the model itself is accurate

Output Validation: interpreting the results correctly

Model Validation

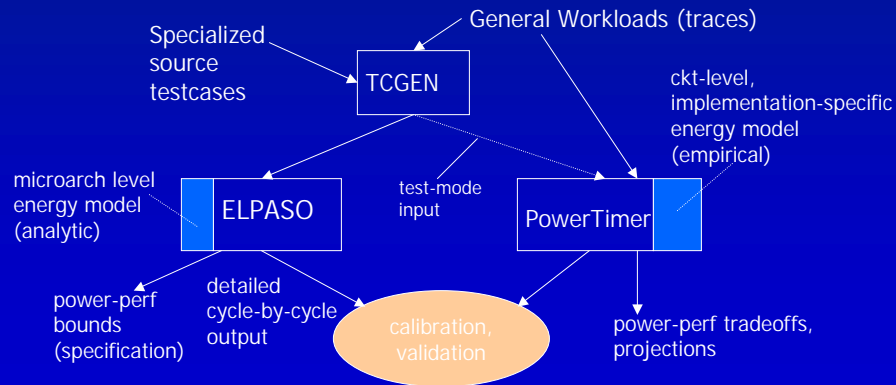
- Main challenge: **defining a specification reference**



- *Secondary problems:*
 - generate apt test cases
 - test case coverage
 - choice of o/p signatures

The Elpaso Reference Model

- Early Stage Power/Perf Analysis, Specification, Optimization
 - > a validation reference model, *under development*
 - > at IBM Watson; working in conjunction with PowerTimer



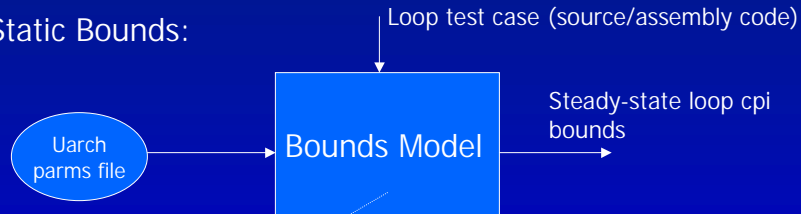
Bounding Perf and Power

- Lower and upper bounds on expected model outputs can serve as a viable "spec", in lieu of an exact reference
- Even a single set of bounds (upper or lower) is useful

Test Case Number	Performance Bounds				Utilization/Power Bounds	
	Cpi (ub)	Cpi (lb)	T (ub)	T (lb)	Upper bound	Lower bound
TC.1						
TC.2						
.						
TC.n						

Performance Bounds

- Static Bounds:



* IBM Research, Bose et al. 1995 - 2000: applied to perf validation for high-end PPC
* U of Michigan, Davidson et al. 1991 - 1998

- Dynamic Bounds:

- analyze a trace; build a graph; assign node/edge costs;
 - process graph to compute net cost (time)
- (e.g. Wellman96, Iyengar et al., HPCA-96)

Static Bounds - Example

```
fadd fp3, fp1, fp0
lfdu fp5, 8(r1)
lfdu fp4, 8(r3)
fadd fp4, fp5, fp4
fadd fp1, fp4, fp3
stfdu fp1, 8(r2)
bc loop_top
```

Consider an in-order-issue
super scalar machine:

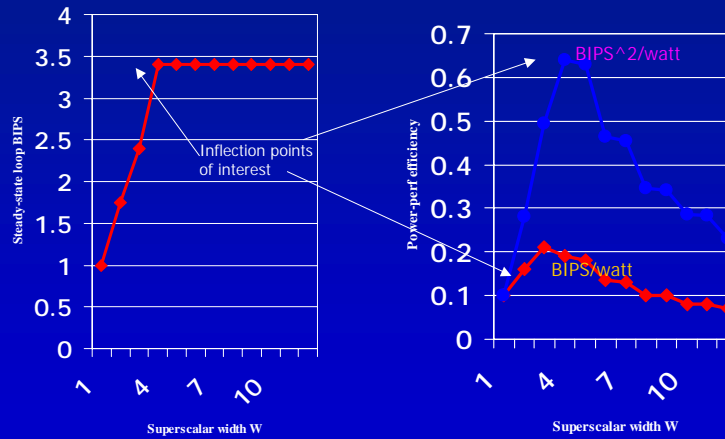
- disp_bw = iss_bw = compl_bw = 4
- fetch_bw = 8
- l_ports = ls_units = 2
- s_ports = 1
- fp_units = 2

N = number of instructions/iteration = 7

- Steady-state loop cpi performance is determined by the narrowest (smallest bandwidth) pipe
 - above example: CPI_{iter} = 2; cpi = 2/7 = 0.286

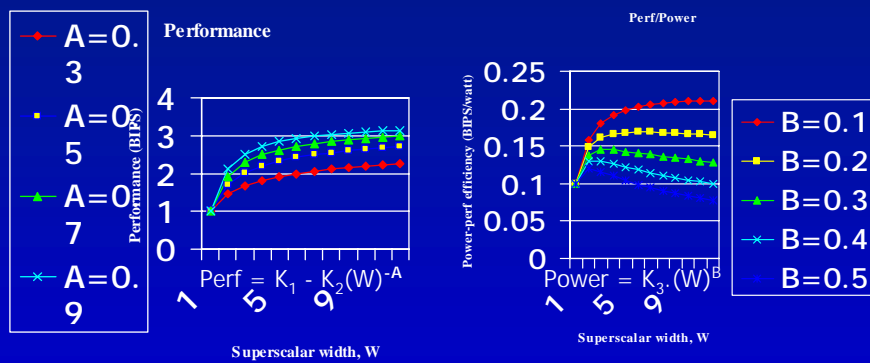
Power-Performance Bounds (El Paso)

$$\text{Power} \sim (W^{*0.5} + \text{ls_units} + \text{fp_units} + \text{l_ports} + \text{s_ports})$$



(adapted from: Brooks, Bose et al. IEEE Micro, Nov/Dec 2000)

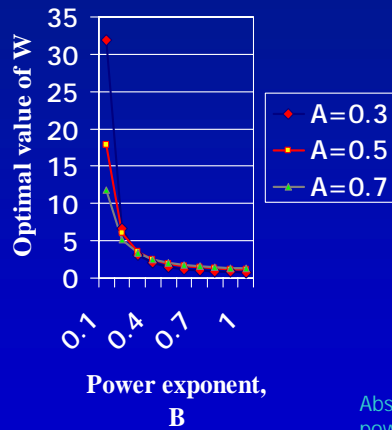
Machine Width Limits (analytical characterization)



$$K_1 = 3.4, K_2 = 2.4 \text{ (to fit prior slide data at end points)}$$

Note: sharp decline in W_{opt} with increase in B

W_{opt} Variation with Power Exponent



$$\text{Log } W_{opt} = (1/A) \cdot \text{log}[(A+B) \cdot K_2 / (B \cdot K_1)]$$

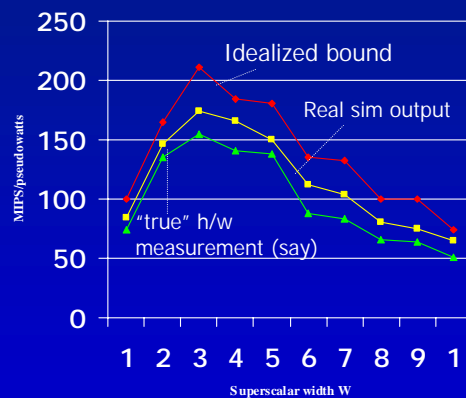
Note:

w/o gated clocks, $B \geq 0.5$;
for that case, $W_{opt} \leq 3$
- not very sensitive to B
in that scenario

Absolute accuracy in estimating
power or B is *not crucial* for $B > 0.5$!!

Efficiency metric used is: Performance/Power

Absolute vs. Relative Accuracy



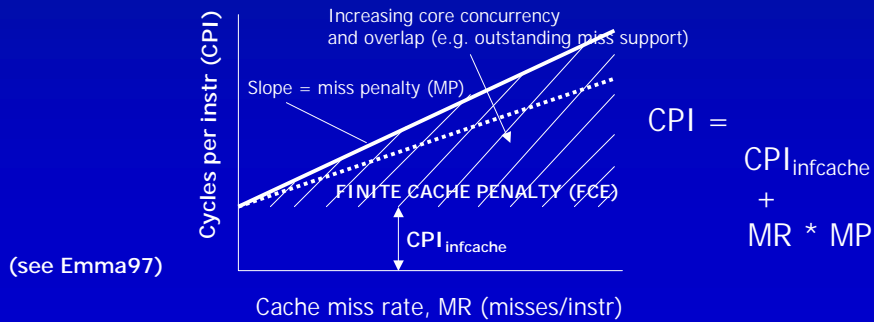
- Poor "absolute" accuracy of simulator
- But, good "relative" accuracy

In real-life, early-stage design tradeoff studies, relative accuracy is more important than absolute accuracy

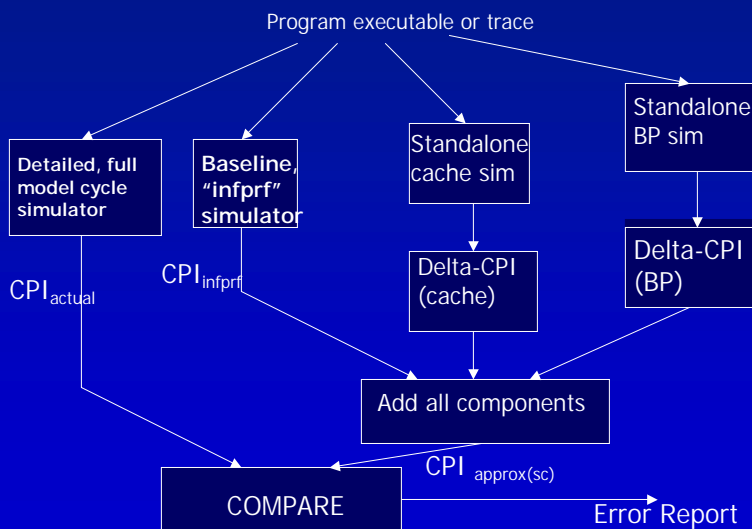
Abstraction via Separable Components

The issue of absolute vs. relative accuracy is raised in any modeling scenario: be it "performance-only", "power" or "power-performance."

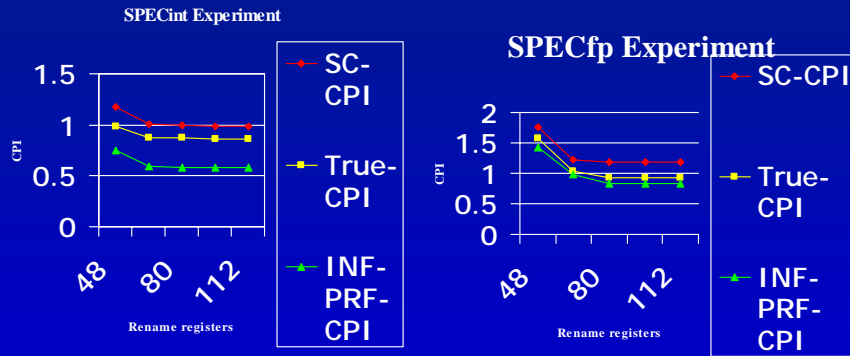
Consider a commonly used performance modeling abstraction:



Experimental Setup



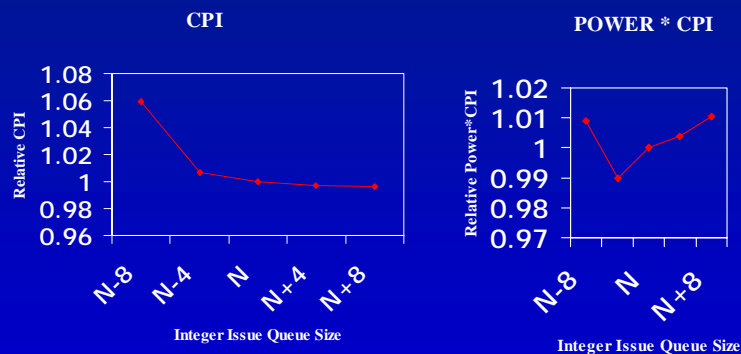
Experimental Results (example)



TRUE-CPI curve: generated using PowerPC research, high-end simulator at IBM (*Turandot* simulator: see IEEE Micro, vol. 19, pp. 9-14, May/June 1999)

Model Output (Data) Validation

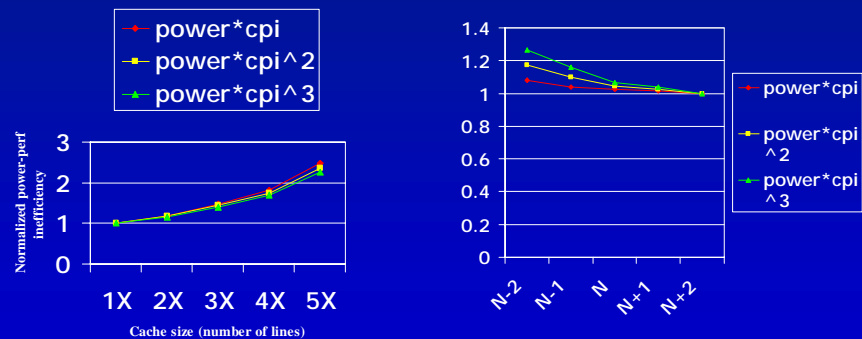
(example, typical data output from PowerTimer)



The sharp slope change of CPI-curve at N-4, coupled with the linear scaling assumption of the energy model validates N-4 as the optimal point for POWER*CPI curve

Output (Data) Validation, contd.

(example, typical data output from PowerTimer)



Conclusion from data: increasing BHT size and reducing data cache size => would have been better choice

Again, validation possible by analyzing CPI data and energy model scaling equations

Accuracy Conclusions

- Separable components model (for performance, and *probably* for related power-performance estimates):
 - > good for relative accuracy in most scenarios
 - > absolute accuracy depends on workload characteristics
- Detailed experiments and analysis in:

Brooks, Martonosi and Bose (2001):

"Abstraction via separable components: an empirical study of absolute and relative accuracy in processor performance modeling," IBM Research Report, Jan, 2001 (submitted for external publication)

- Power-performance model validation and accuracy analysis:
 - > work in progress

Trends and conclusions...

Leakage Power: Models and Trends

- Currently: leakage power is roughly 2-5% of CPU chip's power dissipation
- Future: without further action, leakage power expected to increase exponentially in future chip generations
- The reason?
 - Supply Voltage ↓ to save power =>
 - => Threshold voltages ↓ to maintain performance
 - => Leakage current ↑

Other technology trends and needs

- Need:
 - Good models for leakage current
 - Ways of handling chips with more than one V_t
 - Models that link power and thermal characteristics

Other resources

- Tutorial webpage
 - Web versions of demos
 - Access to slides
 - Semi-comprehensive Power Bibliography...