Power-Aware Microarchitectures: Design, Modeling and Metrics

Pradip Bose
IBM Corporation
pbose@us.ibm.com
Acknowledgements

- Victor Zyuban, IBM
- Alper Buyuktosunoglu, IBM
- Zhigang Hu, IBM
- Viji Srinivasan, IBM
- Hans Jacobson, IBM
- Jude Rivers, IBM
- Phil Emma, IBM
- Hendrik Hamann, IBM
- .....  
- Plus, many others at IBM!

- Kevin Skadron, U of Virginia
- Yingmin Li, U of Virginia
- Margaret Martonosi, Princeton Univ.
- Sarita Adve, Univ. of Illinois

... plus their students

Some of the slides are based on content published in IEEE or ACM sponsored publications; permission to reproduce that content for this lecture material has been applied for
Outline

- Power breakdown data: where does power go?
  - Why is processor or chip-level power important?
  - Power vs. power density vs. temperature
  - Power delivery versus power dissipation
    - Product cost vs. cost of ownership

- Power-performance efficiency metrics
  - Workload and market dependence

- Hierarchical power modeling (levels of abstraction)
  - Microarchitecture-level power-performance-temperature simulators
  - Validation methods
Outline (contd).

- Microarchitectural Techniques for Low Power
  - Defining a power-efficient design point to begin with
    - Optimal core pipeline depths
    - Optimal number of cores in multi-core designs
  - Microarch. support for clock-gating: current vs. future extensions
  - Microarch. support for (predictive) power-gating
  - Adaptive microarchitectures
    - Changing resource sizes, bandwidths, etc on workload demand
    - Dealing with Ldi/dt, on-chip variability, aging and soft error rates
    - Towards on-chip controllers (with software management)

- Summary and Wrap-Up (Q&A)
Understanding power breakdowns….

• *Where does all that power go??*

• *Remember to invoke Amdahl’s Law when developing designs and power models....*
Current Generation Laptop Power Pie (IBM Thinkpad R40)

Idle Power

Max Power Workload

Data courtesy Mahesri et al., U of Illinois, 2004
Typical Server Box Power Pie

Processor motherboard piece (17 %): significant but not dominant
However, power density-wise it is indeed the “hot spot” fraction
Server-Class Processor: Unconstrained Power

Pre-silicon, POWER4-like superscalar design

D. Brooks, et. al. MICRO-03 (tutorial)
Processor Power Pie-Chart: Another View

- High performance processors (prior/current generation) typically burn most of their power in the clocked latches and arrays (registers, caches).

(taken from: Bose, Martonosi, Brooks: Sigmetrics-2001 Tutorial)

Example data

Pre-silicon ckt-sim based; assumes: no clock-gating
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
    - CPI * W: equivalent inverse metric
  - MIPS$^2$/W or SPEC$^2$/W ~ energy*delay (EDP)
  - MIPS$^3$/W or SPEC$^3$/W ~ energy*(delay)$^2$ (ED$^2$P)
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

For a systematic and mathematically sound treatment of the metrics issue, i.e. the right choice of k in SPEC^k/W, see Zyuban et al. ISLPED-02
Choice of metric matters!

Data source: Berkeley CPU Center and http://www.specbench.org; and other single processor-level data (estimated)

- IBM
- Sun

Specint/Watt

Data source: Berkeley CPU Center and http://www.specbench.org; and other single processor-level data (estimated)
Performance-power efficiency on the decline since 1995

Source: David Yen, Sun Microsystems, IRPS-2005 keynote speech

again, we need to be tracking the right metrics in inferring problem trends

How do we quantify temperature-perf efficiency? $1/(\text{execution time}) \times \text{peak temperature}$?
Hierarchical Power and Temperature Modeling
Modeling Hierarchy and Tool Flow

- Early analytical performance models
  - Trace/exec-driven, cycle-accurate simulation models
- Microarch parms/specs
- RTL MODEL (VHDL)
  - RTL sim
- Circuit-level (hierarchical) netlist model
  - gate-level model (if synthesized)
- Layout-level physical design model
  - Cap extract, sim
- Design rules
  - design rule check, validate
- Bitvector test cases
  - edit/tune/debug
- (Architectural) Sim Test Cases
  - edit/debug
- Performance Test Cases
  - edit/debug
- set of workloads

Energy Models

- refine, update

RTL level

- gate-level

ckt-level

layout-level
Power/Performance abstractions at different levels of this hierarchy...

- **Low-level:**
  - Hspice
  - PowerMill

- **Medium-Level:**
  - RTL, Gate-level Models

- **Architecture-level:**
  - PennState: SimplePower
  - Intel: Tempest; ALPS
  - Princeton: Wattch
  - IBM: PowerTimer
  - U of Michigan: PowerAnalyzer
  - ...

**Note:**
Recent work in statistical performance models is a smart abstraction on top of current detailed simulators (L. Eeckhout, et al., Noonburg and Shen, Carl, Nussbaum, Smith, …)
IBM PowerTimer Methodology

**Cycle-by-Cycle Performance Timer (Turandot)**

- **Microarch. Parameters**
  - Cycle-level Hardware access Counts/utilization

**Power Models**

- **Circuit/Tech Parameters**

(Benchmarks and kernels)

Program
Executable
Or trace

8-issue, out-of-order POWER4-like model

Ref: 1) Brooks et al. IEEE Micro, Nov/Dec 2000,
2) PACS-2000 workshop
3) MICRO-2003 tutorial; IBM J. R&D 2003

Performance Estimate

Power Estimate

Drives separate temperature model
New generation, integrated modeling infrastructure


Power Timer: core-level modeling

Power Modeling Enhancements → Package RLC models, Ldi/dt analysis

Temperature Modeling → Reliability Modeling

VALIDATION

Substrate simulator: Turandot

Latch-counts + array power models
Latch-counts + scaled CPAM based models + refined array power models
Trace/exec driven simulation

U of Virginia’s HotSpot, later modified

System interconnect and tech. scaling parameters, models
Uniprocessor CPI and Power sensitivities

Multi-Core Power-Performance Modeling

chip-level microarchitecture modeling

Program traces

Data from device and circuit level

Cycle acc. Processor Simulator

Soft error model

Architectural derating factor

To Interconnect
Layer Thermal Model
Heat Sink
Silicon Die
Heat Spreader
Thermal Interface Material
Fin-to-air convection thermal resistor

L2 C7 L2 L2 C C 4 C C 8

Latch-counts + array power models
Latch-counts + scaled CPAM based models + refined array power models
Trace/exec driven simulation

microarch design and definition
Power Modeling Infrastructure with PowerTimer

Circuit Power Data (Macros) → Tech Parms → uArch Parms → Program Executable or Trace → Architectural Performance Simulator

AF/SF Data → Compute Sub-Unit Power

SubUnit Power = f(SF, uArch, Tech)

Power → CPI

D. Brooks, et. al. MICRO-03 (tutorial)
PowerTimer: Energy Models

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

![Energy Models Diagram](image)

D. Brooks, et. al. MICRO-03 (tutorial)
Key Activity Data

- **SF** => Moves along the Switching Power Curve
  - Estimated on a per-unit basis from RTL Analysis

- **AF** => Moves along the Clock Power Curve
  - Extracted from Microarchitectural Statistics (Turandot)

D. Brooks, et. al. MICRO-03 (tutorial)
Example: Fixed Point Issue Queue

- Made up of 5 macros
  - fxq_control, fxq_data, fxq_gtag, fxq_pointer, fxq_wdl

D. Brooks, et. al. MICRO-03 (tutorial)
Overall Validation Methodology (PowerTimer)

1. PowerTimer
   - elpaso bounds timer
   - cpi and utilization bounds
   - detect anomalies

2. Reference Model (e.g. M2 or M3)
   - timeline output
   - detect mismatch

3. Temperature Model
   - simulated chip temp profile

4. LaSpecs
   - cpi and utilization stats
   - tabular (html) web specs
   - detect mismatch

5. IR Thermometry Setup
   - direct image of chip temp profile

H. Hamann, M. McGlashan-Powell, et al. (planned future path)
U of Virginia HotSpot Thermal Model

- **Thermal Modeling**
  - Want a fine-grained, dynamic model of *temperature*
    - A model that microarchitects and system architects can use
    - At a granularity that they can reason about
    - That accounts for adjacency and package
    - That is fast enough for practical use
  - Averaging power dissipation is not accurate
    - Chip-wide average will not capture hot spots
    - Localized average will not capture lateral coupling
    - Does not account for block areas (i.e. power density)
  - HotSpot – a new model for localized temperature
    - Computationally efficient for use in power/performance simulators
    - Validated against FEM models (physical validation coming soon)
    - Publicly available
HotSpot Thermal Model

- **A compact thermal**
  - Models all parts along both primary and secondary heat transfer paths
  - At arbitrary granularities
  - Fast and accurate

![Diagram of thermal components]

Courtesy, W. Huang, K. Skadron et al. U of Virginia
DAC-2004 talk
Electrical-Thermal Duality

- $V \propto \text{temp (T)}$
- $I \propto \text{power (P)}$
- $R \propto \text{thermal resistance (Rth)}$
- $C \propto \text{thermal capacitance (Cth)}$
- $RC \propto \text{time constant}$

Courtesy, W. Huang, K. Skadron et al. U of Virginia
DAC-2004 talk
http://www.dac.com/42nd/talkindex.html
Typical CMP Thermal Map [PowerTimer/Turandot]
SMT Example: Swim + Swim

Temperature Variation of Each Unit (swim.swim)
Power-related issues in chip design

Capacitive (Dynamic) Power

Static (Leakage) Power

Di/Dt (Vdd/Gnd Bounce)

Temperature

Minimum Voltage

<table>
<thead>
<tr>
<th>Current (A)</th>
<th>Voltage (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.100</td>
</tr>
<tr>
<td>50</td>
<td>1.050</td>
</tr>
<tr>
<td>0</td>
<td>1.000</td>
</tr>
</tbody>
</table>

20 cycles
Power Consumption vs. Cooling Cost

The architecture community needs to understand the thermal cost metrics better

S. H. Gunther et al., Intel Technology Journal, 2001

A more appropriate x-axis metric to plot might be: watts/sq.mm per degree Kelvin above ambient
Power, temperature and reliability-awareness at the highest levels of design abstraction

- **(Micro)-Architecture & Compilers**
  - Optimize basic pipeline depth for power-perf-reliability
    - Optimize number of cores per die
    - Optimize core complexity and threading
  - Shrink structures; reduce complexity
  - Shorten wires; link early definition to floorplan
  - Reduce activity factors:
    - gate clock, Ifetch, ...
    - adapt resource sizes
  - Turn on units on-demand; gate \( V_{dd} \) (predictive)
  - Trade off parallelism against clock frequency
  - Reduce wasted work: “standard” operations

- **Operating Systems**
  - Natural: OS is traditional resource manager
  - Equal energy scheduling
  - Thermally-aware adaptation

- **Application/Algorithm**
  - Additional opportunities; open research issues..
Power-Performance Efficient Processor Core
Pipelines: *definition and analysis*
Factors Affecting Choice of Pipeline Depth

- Cycles-Per-Instruction, CPI (drops due to latencies)
- Clock Frequency
- Growth in the latch count (# of stages and width)
- Clock Gating Opportunities (more idle cycles)
- Growth in logic size
- Growth in # of buffers (slew constraints)
- Glitching Activity (latches filter out glitches)
Pipeline Power-Performance Basics

Consider an ideal, hazard-free pipeline flow;
T = total time per operation (without the latches); L is the latch overhead

\[ E = eK + C, \]  
where,
\[ e = \text{latch energy per pipe stage}, \quad L = \text{latch overhead} \]
\[ C = \text{energy expended in the logic} \]

\[ \text{Energy, } E \]
\[ \text{ops/sec (mips)} \]
\[ \text{Number of pipeline stages, } K \rightarrow \]
\[ \frac{1}{T/K + L} \]
\[ K \rightarrow \]

So, highest freq. design is not the most energy efficient!
Parallelism (SIMD, CMP, SMT) \( \rightarrow \) extends scalability hierarchically
Pipeline Scaling

4 Stage FPU = 16FO4 Logic + 3FO4 Latch = 19 FO4 ~ 2.0GHz

5 Stage FPU = 13FO4 Logic + 3FO4 Latch = 16FO4 ~ 2.4GHz

6 Stage FPU = 11FO4 Logic + 3FO4 Latch = 14FO4 ~ 2.7GHz

9 Stage FPU = 7FO4 Logic + 3FO4 Latch = 10FO4 ~ 3.8 GHz

Cumulative FO4 Depth (Logic + Latch Overhead)  Srinivasan, et. al., MICRO'02
Scaling of a single core: pipeline depth

Growth in latch count in for deeper pipelines

- The number of latches may grow super-linearly with the pipeline depth
- The latch count growth can be modeled as
  \[ \text{LatchCount} = \text{LatchCount}_{\text{base}} \times \left( \frac{\text{FO4}_{\text{base}}}{\text{FO4}} \right)^{\text{LGF}} \]
- Here FO4 is the logic delay per stage (excluding latches)

Example: FPU Multiplier (Booth recorder and Wallace tree)

Pipepline Cuts FO4 (including 3FO4 of latch)

<table>
<thead>
<tr>
<th>Pipeline Cuts</th>
<th>FO4 1</th>
<th>FO4 2</th>
<th>FO4 3</th>
<th>FO4 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10FO4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13FO4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16FO4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19FO4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Booth Recode and Booth Mux with Aligner
Cumulative number of latches in the multiplier pipelined for various FO4

For this example LGF is in the range of 1.4 to 1.9 depending on which two cut points are compared average LGF is 1.5 (for 19FO4 and 10FO4)

Glitching activity vs. FO4

Measured for a set of eLite functional units

Zyuban, Transactions on VLSI
Several effects put together

![Graph showing the relationship between total power and total FO4 per stage. The graph includes data points for latch growth factor, frequency, clock gating factor, glitch factor, and leakage power. The y-axis represents power relative to 19FO4, while the x-axis represents total FO4 per stage.](image)

Zyuban, et. al., Transactions on Computer’ 04
Deducing Optimal Pipe Depths

V. Srinivasan et al., MICRO-35, 2002

Power-performance optimal

Performance optimal

Relative to Optimal FO4

0.8

0.6

0.4

0.2

0

1

37 34 31 28 25 22 19 16 13 10 7

Total FO4 Per Stage

SPEC2000 suite

bips

bips/W

bips^2/W

bips^3/W

bips

bips/W

bips^2/W

bips^3/W
Workload impact:
TPCC Trace

Power-performance optimal

Performance optimal

Relative to Optimal FO4

Total FO4 Per Stage
Some observations

- **Active power grows approximately as a square of the pipeline depth**
  - Superlinear growth in the number of latches
  - Linear growth in frequency

- **Leakage power grows sublinearly with the pipeline depth**
  - Growth in latch area
  - Growth in logic area
  - Growth in buffer sizes

- **In a leakage dominated design it is less prohibitive to go to deeper pipelines**
Impact on Design

- Tradeoff via pipeline depth
- Tradeoff via changing Vdd
- Tradeoff via frequency

Relative Power

Relative Time per Instruction

Optimal BIPS^3/W

Maximum Power Budget

Zyuban, et al., Transactions on Computer' 04
Integrating Multiple Cores on Chip

- With uniprocessor performance improvements slowing, multiple cores per chip (socket) will help continue the exponential system performance growth
- Exploit performance through higher levels of integration in chips, modules, and systems
- Invest power in chip-level performance rather than core performance

POWER 4: 2001
180 nm, Cu, SOI
2 cores / chip

POWER 4+: 130 nm

POWER 5: 2004
130 nm, Cu, SOI
2 cores / chip
2 way SMT / core
Building Blocks for Chip-level Integration

For a given power budget, higher throughput is achieved by multiple simple cores on both SMP workloads and independent threads.

A complex core provides much higher single-thread performance; scaling up a simple core by reducing FO4 and/or raising Vdd does not achieve this level of performance.

It may be worthwhile to have multiple heterogeneous cores on chip.

The appropriate design point depends on the workload that is being supported.

Source: Zyuban et al. IBM tech. report 2004
Clock-gating: classical techniques + new advances
Some issues with clock gating

- **There are two styles of gating (early OR-style and late AND-style)**
  - Early style intercepts C1 (more efficient, but more difficult to time)
  - Late style intercepts C2 (may require re-timing L2 latch)

- **Both styles work with pulsed latches**
  - Pulsing C1 is more power-efficient
  - Pulsing C2 gives more time for clock gating logic

- **Typically cannot blindly replace data recycling multiplexor with clock gating**
Functional clock gating

Clock gating with “chicken switches” or “loose” gating

V. Zyuban, INTELLECT low power course, Sweden, 8/2004
Clock-gating Efficiency: single-threaded vs SMT

H. Jacobson et al.
HPCA-2005
Floating Point Unit: Levels of Clock Gating

- Unit gating
- Stage gating
- Register gating

Relative clock power

H. Jacobson et al.
HPCA-2005
Active Power Savings from Clock-Gating (% over baseline) (POWER5-like processor core; pre-silicon projections)

<table>
<thead>
<tr>
<th>Workload</th>
<th>IFU</th>
<th>IDU</th>
<th>ISU</th>
<th>FXU</th>
<th>LSU</th>
<th>FPU</th>
<th>CORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes (i)</td>
<td>9.8%</td>
<td>53.3%</td>
<td>23.7%</td>
<td>15.8%</td>
<td>40.0%</td>
<td>41.8%</td>
<td>31.0%</td>
</tr>
<tr>
<td>SAP (i)</td>
<td>11.5%</td>
<td>48.4%</td>
<td>25.9%</td>
<td>16.3%</td>
<td>40.1%</td>
<td>42.5%</td>
<td>31.6%</td>
</tr>
<tr>
<td>TPC-C(i)</td>
<td>10.4%</td>
<td>53.0%</td>
<td>23.2%</td>
<td>15.9%</td>
<td>40.0%</td>
<td>42.6%</td>
<td>31.3%</td>
</tr>
<tr>
<td>TPC-C(p)</td>
<td>12.0%</td>
<td>45.7%</td>
<td>25.9%</td>
<td>16.3%</td>
<td>39.9%</td>
<td>44.2%</td>
<td>32.2%</td>
</tr>
<tr>
<td>DAXPY</td>
<td>17.1%</td>
<td>67.3%</td>
<td>6.2%</td>
<td>16.3%</td>
<td>16.9%</td>
<td>21.8%</td>
<td>19.3%</td>
</tr>
<tr>
<td>SparseMV</td>
<td>11.2%</td>
<td>64.2%</td>
<td>10.5%</td>
<td>15.6%</td>
<td>24.2%</td>
<td>33.6%</td>
<td>24.6%</td>
</tr>
<tr>
<td>TPP</td>
<td>23.5%</td>
<td>79.0%</td>
<td>9.3%</td>
<td>16.5%</td>
<td>20.1%</td>
<td>37.5%</td>
<td>26.4%</td>
</tr>
</tbody>
</table>

Note: post-silicon hardware-based analysis shows good agreement at the full core level
Clock-gating helps reduce leakage power as well!

POWER5 Chip

w/o CG  with CG

Thermal Image Plots (measured)
Conventional Clock Gating: summary

- Effective, low-complexity, low-overhead scheme for reduction of active power in microprocessors
  - was already prevalent in embedded processors and ASICs
  - now the main power management technique in server-class processors
  - 20 to 50 % reduction in active power, depending on workload and granularity of gating
  - temperature reduction leads to leakage power savings as well
Pipeline Clocking Re-Examined

- **Traditional opaque-mode clock gating is not optimal**
  - Generates significant amount of clock pulses that are redundant to the correct operation of the pipeline
  - Problem is that latches are held opaque by default (when gated off)
  - Requires every latch to be clocked in order to pass a data item through the pipeline

- **Idea: hold latches in the transparent mode by default (when gated off)**
  - Data items can pass through pipeline without clocking if they are sufficiently spaced in time
  - Latches are only clocked when needed to avoid data races for closely spaced data items
  - Called “Transparent Clock Gating”

- **Transparent gating allows gating clock to active pipeline stages**

H. Jacobson, ISLPED-04
Clocking in transparent clock gating

- **Requirement to avoid data race**
  - For each pair of distinct adjacent data items (A,B) propagating through a linear pipeline, where A is downstream of B, at least one opaque latch stage must separate A from B.

- **Criteria for optimum clocking**
  - For each pair of adjacent data items (A,B) propagating through a linear pipeline, where A is downstream of B, only the latch stage for A is clocked, and only when B overwrites A’s current state holder.

H. Jacobson, ISLPED-04
Opaque vs. Transparent Pipeline Clocking

Pipeline with traditional clock gating (opaque gating)

Pipeline with transparent clock gating

H. Jacobson, ISLPED-04
Propagation of two data items separated by one cycle through pipeline with transparent gating

Transparent clock gating (1 clock pulse)

Traditional opaque clock gating (6 clock pulses)

H. Jacobson, ISLPED-04
Implementation: 2-stage Control Logic

- Valid-base look-behind (or ahead?) logic gates each LCB
- The timing of the propagation of “valid” signal may be challenging for longer sections of the pipeline
- Propagation of glitches may reduce potential savings for long pipeline sections

H. Jacobson, ISLPED-04
32x32 Multiplier/Adder

- High performance Booth encoded with carry select final adder (by Peter Cook)
- Six pipeline stages
- Latches in stages 1, 2 and 4, 5 are clock gated in transparent mode
- Latches in stages 0, 3, 6 are clock gated in opaque mode

H. Jacobson, ISLPED-04
Results: 32x32 Multiplier/Adder

- Max relative clock power savings is 60% achieved at 20% pipeline utilization
- Max absolute clock power savings is 43mW achieved at 50% pipeline utilization
Results: 32x32 Multiplier/Adder

- Introduced data glitch power < 10% of clock power savings

H. Jacobson, ISLPED-04
Clock Power Reduction via Transparent CG

H. Jacobson, P. Bose et al., HPCA-2005
Transparent Pipelines: summary

- **Transparent pipeline allows clock gating even of active pipeline stages**
  - Latch stages are transparent, rather than opaque, when gated off

- **Reduces clock power**
  - Absolute clock power savings optimal for 50% pipeline utilization (0101… case)
  - Relative clock power savings optimal for 20% pipeline utilization

- **Limitations**
  - Valid bit signal distribution over multiple stages restricts the practical length of a transparent pipeline segment to 2-3 stages depending on cycle time
  - Signal feedback within the same stage, e.g., state signals in control logic, may restrict the use of transparent latches. However, transparent and opaque latches can be mixed and matched freely, so only signals with direct feedback need to operate in opaque mode.

H. Jacobson, ISLPED-04
Elastic pipeline implementation

- Idea: MS latch allows storing two data items in one master/slave latch
- Use master half of the latch as a stall buffer
  - Data gets compressed as stall propagates up the pipeline
  - Data gets decompressed as un-stall propagates up the pipeline
  - During normal operation mode latch is clocked as a normal master/slave latch
  - During stall, latch stores two data items, one in the master and one in the slave

H. Jacobson, et. al., ASYNC-03
Elastic pipelines: summary

- Useful for progressively stalling a pipeline, one stage per cycle
- Reduces power in stallable pipelines
- Improves slack on data signals
  - Improved slack since no mux needed
  - Less capacitance on data wire feeding into latch since no stall buffer latch needed

- Cost
  - Master and slave latches are clock gated separately (requires two gating signals)
  - Additional scan latches may be needed for bring-up purposes (not needed for testing)

H. Jacobson, et. al., ASYNC-03
SOC – level power gating vs. fine grain power gating

Potential wireless SOC

- SOC-level power gating
  - Put unused cores into sleep mode
  - Typically under OS control
- Fine grain power gating in microprocessor core
  - Fine-grain gating of unused resources in the active mode
  - Timely waking up of gated resources

High-performance microprocessor core
Virtual Vdd discharge in the power gated mode
Key Intervals in the Power Gating Cycle

- $T_{idle\ detect}$
- $T_{idle\ delay}$
- $T_{break\ even}$
- $T_{full\ discharge}$
- $T_{busy\ delay}$
- $T_{wakeup}$

$T_{breakeven} \sim 10-17$ cycles

Z. Hu, et. al., ISLPED'04
Power Gating Potential (I)

FPU, FXU gating potential for different values of $T_{\text{breakeven}}$ running SPECfp2K benchmarks

Z. Hu, et. al., ISLPED’04
Power Gating Potential (II)

FXU gating potential for different values of $T_{\text{breakeven}}$ running SPECint2K benchmarks

Z. Hu et al. ISLPED’04
Time-Based Power Gating Results (I)

for FPU running SPECfp2K benchmarks

Z. Hu, et. al., ISLPED’04
Drowsy and Decay Caches

- **Key idea**
  - Reduce leakage power by lowering Vdd
  - Kim, Flautner, Blaauw, Mudge [2002-04]
  - Least upper bound that preserves state
  - Prior decay cache idea (Kaxiras, Zhu, Martonos) uses Vdd-gating (loses state, but more savings)
  - Word lines in the drowsy state until accessed – penalty
  - Periodically clear all lines to the drowsy state – simple circuitry
Drowsy Cache

- Control for power and word lines:

Kim et al. U of Michigan
Drowsy Cache

- **SPICE Simulations**
  - Berkeley models + International Technology Roadmap for Semiconductors
  - Used 0.07 µm in simulations
  - 3 GHz
- **1 cycle wake up**
  - Also examined 2 and 3 cycle wakeup for a 10 GHz machine
- **4000 cycles between resets**
- **Power saving in dcache**
  - 80-90%
- **Comparison – gated Vdd**
  - 10-15% better, but state is lost
  - Complex policy

Kim et al. U of Michigan
Adaptive Microarchitectures
A mismatch between:

- front-end producer rate and back-end consumer rate
- the supplied instruction window from the front end and the required instruction window to exploit the level of application parallelism results in additional front-end energy
Exploiting Workload Variability: On-Demand Reconfiguration

- Adapt queue/buffer sizes or cache configuration on-demand, to save power (ISLPED-02)
- Adapt instruction fetch/dispatch rates (fetch gating, ISLPED-02 ISCA-03)
- Adapt clock speeds or voltages dynamically
Saving Energy with Just-In-Time Instruction Delivery

Tejas S. Karkhanis, James E. Smith, Pradip Bose

Published in ISLPED-2002
TYPICAL PROCESSOR

Insn. Delivery

Decode pipeline:

Increase with deeper pipes
Energy Activity w/o Energy Saving Mechanism

![Graph comparing energy activity for different tasks](image-url)

- **GCC**
  - Fetch: 0.8
  - Decode Pipe: 0.6
  - Issue Queue: 0.4

- **BZIP**
  - Fetch: 0.4
  - Decode Pipe: 0.2
  - Issue Queue: 0.4
Just-In-Time (JIT) Insn. Delivery Microarchitecture

- I-$
  - Decode Pipeline
  - MAXcount
  - compare
  - Insn. count
  - Exec. Units
  - Issue Queue
  - Re-order Buffer

- Stop fetch if Insn. Count > MAXcount
- Decrement on commit or flush
- Incr. on fetch
- Insn. Fetch gating

Insn. Fetch gating
Control Algorithm

- Programs go through phases
  - Dynamically change MAXcount
- Coarse Grain configuration
  - Window size: 100K committed instructions

Phase changed or timeout

STABLE -> Stable phase detected

STABLE <-> UNSTABLE
Prior Approaches

- **Pipeline (Confidence) Gating, Manne et. al. ISCA 1998**
  - Saves “flushed” cycles
  - Requires a confidence table

- **Adaptive Issue Queue, Buyuktosunoglu et. al. 2001-2003**
  - Saves “stalled” cycles in the Issue Queue
  - Increase “stalled” cycles in the Decode Pipe
  - Intrusive on the issue queue logic
Energy Savings

Not including energy savings in:
- Re-order Buffer
- Register File Accesses
- Load-Store Queues
- Data Cache
Performance Impact

Normalized IPC

- Base
- Oracle
- AIQ
- PG
- JIT{2%}
- JIT{5%}
- JIT{10%}
Summary: JIT Instruction Fetch

- **JIT reduces wasted energy for:**
  - Active Flushed
  - Stalled Used
  - Stalled Flushed

- **Simpler hardware than the previous work**
  - MAXcount, Total_insn_count, Adder/Subtractor and comparator

- **Coarse grain control algorithm**
  - Implement in hardware
  - Implement in VMM
Issue Centric Fetch Gating Algorithm

Buyuktosunoglu et al.
Co-adaptive Instruction Fetch and Issue

- CPI degradation is small
- Fetch gating has a much greater energy-delay impact
- 20% greater reduction in energy-delay and 44% greater reduction in issue queue energy than previously published fetch gating scheme
- The additional fetch stalls with dynamic adaptation increases the performance degradation
- Combined approach achieves a significant reduction in issue queue energy as well as overall energy-delay

Buyuktosunoglu et al., ISCA2003
GALS/MCD Architectures [Marculescu et al., Albonesi et al.]

- Variation-tolerant
- Power-Efficient
Inductive Noise and its Control in Adaptive Designs

- Ongoing research: practically feasible on-chip adaptive control to ensure reliable operation

Microarchitectural prediction techniques used effectively to anticipate workload phases and events - i.e., periods of inactivity and specific noise (Ldi/dt) events

✓ preliminary workload characterization studies have yielded encouraging results
✓ predictive feature helps minimize performance overheads
Intel’s Montecito: A Real Example

(on-chip controller)

1.72 Billion transistors

1MB L2I

27.72 mm

Dual Cores

21.5 mm

2 X 12MB L3 Caches with Pellston

Caveat: full functionality and benefit of Foxton will not be available in initial systems

Foxton Power Controller

2-Way Multi-Threading

Soft Error Detection and Correction
Per-Chip Optimization

Power @ Fixed Voltage/Frequency

Power after Per-Chip Optimization

Determine Optimal $V_{DD}$ Per chip

Reduced Peak Power

$V_{DD}$ Distribution
Power-Aware Microarchitecture: summary

- **Power-perf efficient choice of pipeline depth (F04/stage) and #cores**
  - A fundamental error here could lead to post-silicon power-performance (and hence, cost-performance) deficiency

- **Area and leakage-efficient design**
  - Simpler cores; balanced single- vs. multi-thread performance
  - Fine-grain power-gated to further reduce leakage
    - Predictive support built into microarchitecture & compiler to minimize overhead

- **Gated clock, bandwidth (fetch, issue, …), register ports …**
  - Granularity of application determines active power savings
  - Cycle-time pressure may inhibit pervasive gating throughout the logic
    - Verification complexity is another concern

- **Adaptive (reconfigurable, resizable) resources**
  - Applicable to on-chip logic and buffers (caches, registers, etc)
  - Potentially save active and leakage power

- **GALS/MCD architectures**
  - Addresses on-chip variability; also improves power-efficiency
Latest Chip Microarchitecture Paradigms:  
*SMT and CMP: Power and Temperature Impact*
Multithreaded Instruction Flow in Processor Pipeline

- **Instruction Fetch**
  - IF
  - IC
  - BP
  - D0
  - D1
  - D2
  - D3
  - Xfer
  - GD

- **Branch Prediction**
  - Program Counter
  - Branch History Tables
  - Target Cache

- **Instruction Translation**
  - Instruction Cache
  - Instruction Buffer 0
  - Instruction Buffer 1
  - Thread Priority

- **Group Formation, Instruction Decode, Dispatch**
  - Group Formation and Instruction Decode

- **Dynamic Instruction Selection**
  - Shared Issue Queues
  - Shared Register Mappers

- **Shared Execution Units**
  - LSU0
  - FXU0
  - LSU1
  - FXU1
  - FPU0
  - FPU1
  - BXU
  - CRL

- **Out-of-Order Processing**
  - MP
  - ISS
  - RF
  - EX
  - DC
  - Fmt
  - LD/ST
  - WB
  - Xfer

- **Store Queue**
  - Group Completion
  - Data Translation
  - Data Cache
  - L2 Cache

- **Interrupts & Flushes**
  - BR
  - WB
  - Xfer
POWER5 High Level Diagram

Legend:
- IFU:
- IDU:
- ISU:
- FXU:
- LSU:
- FPU:

Ifar

br pred Link stk Count$

Instruction cracking group forming

register mapping

register mapping

GCT register mapping

ISSU:

GPR

FPU:

FPR

fpscr

SDQ

SRQ

LRQ

TLB

SLB

DPrefetch

LMQ

D$
Power and performance Efficiency (SMT)

Power and performance efficiency (SMT)

Power Breakdown by units

Three Categories:
ISU FXU
IFU LSU
IDU

Unit power change compared to ST

Sensitivity to leakage power

SMT Power overhead ratio decreases as leakage factor increases

Sensitivity to resource power scaling


The trend does not change with the variation of PowerFactor!!!

PowerFactor = 1.0
PowerFactor = 1.1
PowerFactor = 1.2
PowerFactor = 1.3
PowerFactor = 1.4

Energy delay 2 product change compared to ST

Resource scaling factor
Conclusions about SMT Power Efficiency

- SMT is a power-efficient design paradigm for modern, superscalar microarchitectures
  - performance gains of nearly 20% with a power uplift of roughly 24% leading to significant reduction in $ED^2$

3 heat-up mechanisms

- Unit **self heating** determined by the power density of the unit
- **Lateral thermal coupling** between neighboring units
- **Global heating** through TIM (thermal interface material), heat spreader, and heat sink

Y. Li, Z. Hu et al., P=AC², 2004
SMT vs. CMP Performance and Power Efficiency Analysis (without DTM)

SMT is superior for memory bound (high-L2-cache-miss rate) benchmarks while CMP is superior for non memory bound benchmarks

(Y. Li, Z. Hu et al., P=AC^2, 2004)
SMT vs. CMP Performance with DTM

Y. Li, Z. Hu et al., P=AC², 2004

Localized DTM method favors SMT while global DTM method favors CMP
Localized method will lead to higher global power consumption, but the performance advantage of localized method for SMT will lead to better energy-delay product result for localized method compared to global method in some cases.

Y. Li, Z. Hu et al., P=AC², 2004
Conclusions about temp. efficiency (SMT, CMP)

For POWER4/5-like architecture,

- CMP and SMT temperatures are comparable with current generation process technologies, but their thermal heating mechanisms are quite different.
  - SMT heating is primarily caused by localized heating within certain key micro-architectural structures such as the register file, due to increased utilization.
  - CMP heating is primarily caused by the global impact of increased energy output.

- When leakage power is significant, CMP machines are clearly hotter than SMT.

- With the same chip area, SMT performs better than CMP for memory bound benchmarks while CMP wins for non-memory bound workload.

- Localized DTM schemes perform better for SMT while global DTM schemes favor CMP.

Y. Li, Z. Hu et al., P=AC², 2004
Power-Aware vs. Temperature-Aware

Test case: chip 18 x 12 mm² in a standard cooling package....

<table>
<thead>
<tr>
<th></th>
<th>power-aware design</th>
<th>temperature-aware design</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{total}}$</td>
<td>$25 \text{ W}$</td>
<td>$100 \text{ W}$</td>
</tr>
<tr>
<td>Power map</td>
<td>185 W/cm²</td>
<td>46 W/cm²</td>
</tr>
<tr>
<td>0 W/cm²</td>
<td>41 K</td>
<td></td>
</tr>
<tr>
<td>93 K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temp. map</td>
<td>98 K</td>
<td>111 W/cm²</td>
</tr>
<tr>
<td>185 W/cm²</td>
<td>96 K</td>
<td></td>
</tr>
<tr>
<td>21 W/cm²</td>
<td>61 K</td>
<td></td>
</tr>
<tr>
<td>7 K</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

save 10 W in low power density region ($P_{\text{total}}=40\text{ W}$)

save 10 W in high power density region ($P_{\text{total}}=40\text{ W}$)

Courtesy: Hendrik Hamann, Thermal Physics Dept., IBM Research
Power, performance, temperature, reliability…

- **Chip-level functional robustness declining in future technologies**
  - increase in transient and permanent errors:
    - *power density and temperature problems (hot spots) is an example factor*
    - *Ldi/dt noise (exacerbated by dynamic power or temp management)*
    - *Full chip burn-in limited by leakage power*
    - *Soft error rates on the rise due to technology factors*
  - power, area, yield (cost) pressures ➔ less scope for redundancy thru’ replication
  - increase in chip complexity ➔ impacts verification time (cost)
  - variability will impact design and CAD tools at all levels of abstraction

We may be entering a disruptive period where tradeoffs between single-chip performance, power, temperature and reliability become mandatory
Reliability-aware microarchitecture research at IBM: progress so far

- **RAMP**: **R**eliability **A**ware **M**icro**P**rocessor **D**esign [ISCA’04]
  Architecture-level model for lifetime reliability analysis
  Uses state-of-the-art device level models for wear-out failures

- **Scaling analysis** on POWER4-like core [DSN’04]
  Quantified impact of scaling on lifetime reliability

- **Dynamic Reliability Management** [ISCA’04]
  Architectural technique for reliability control

- **Exploiting Structural Duplication for Lifetime Reliability** [ISCA’05]
  Performance-area-reliability tradeoffs with selective duplication

- **SoftArch**: microarchitecture-level MTTF projection for given incident soft error rates [DSN’05]

Collaborative Work with Sarita Adve’s Group at UIUC
Integrated, SoC-like Server-Class Microarchitectures

- Multi-core processors will need complex, on-chip management to maintain balance between power, temperature, reliability and performance
  - Adjusting dynamically to temperature-sensitive variabilities will also be required
  - Field BIST may augment pre-silicon verification
  - Graceful, managed degradation and/or managed replacement/sparing may be needed to extend chip lifetime or control degree of soft error tolerance
  - Managing redundant resources for dynamic reliability-performance tradeoffs
- Integrated hardware-software solutions to minimize hardware complexity and added power, are likely
- Server-class processor chip designs are likely to resemble SoC-like architectures with attended design methodologies in future
Technology-Aware Integrated Microarchitectures

- Frequency growth curb has led to the trend of lower frequency, multi-core chip microarchitectures
  - recent announcements by Intel and Sun consolidate industry-wide trend

- Technological trends coupled with design trends dictated by power-awareness is leading to the prospect of degraded chip-level reliability and/or reduced performance growth even after the “right hand turn”, to lower frequency, multi-core designs
  - unused cores or sub-cores must be power-gated off, depending on workload demand, to save power, perhaps at some performance cost
  - temperature-aware floorplan and dynamic activity migration will be needed to mitigate hot spot problems, again at some performance cost
  - on-chip power and temperature management must be balanced against performance and reliability budgets; error tolerance must be done at low cost (area overhead)
  - intra- and inter-chip variability will require variation-tolerant design, one that adapts to chip-specific operating frequency and thermal design point on “power-on” and perhaps dynamically

- multi-dimensional optimization and continuous self-calibration \(\rightarrow\) will require integrated, on-chip controller that manages multiple, possibly heterogeneous computing cores and storage resources
  - area pressure (leakage, yield) will force such multi-dimensional optimizers (controllers) to be hardware-software solutions (i.e. will involve the compiler, hypervisor, OS)
  - server processor chips will increasingly become SoC like
Hardware Integration in BlueGene/L: System-on-a-Chip ASIC

- IBM CU-11, 0.13 μm
- 11 x 11 mm die size
- 25 x 32 mm CBGA
- 474 pins, 328 signal
- 1.5/2.5 Volt

**Integrated functionality**
- Two PPC 440 cores
- Two “double FPUs”
- L2 and L3 caches
- Torus network
- Tree network
- JTAG
- Performance counters
- EDRAM
Concluding Remarks (for whole tutorial)

- **Power-performance tradeoff analysis must be integral part of early-stage definition of microprocessors**
  - Fundamental design decision errors can lead to post-silicon power overruns and/or performance shortfalls
  - Pre-silicon power-performance modeling and validation methodology: key investment needed to prevent post-silicon surprises
  - Power analysis and tuning must percolate through all stages of design, with closed loop feedback to higher levels.

- **Power “optimization” in server-class, high-end systems can be quite different from that in embedded systems**
  - Performance cannot be sacrificed
  - Reducing the worst case power is the primary goal; but, average power is also important
  - “Low power” means “max performance at given power budget”
  - Power-performance efficiency metric is different for different target applications/markets
Concluding Remarks (contd.)

- **Power-aware microarchitecture techniques: one of the biggest levers in future power reduction at the chip and system level**
  - But, co-design with logic/circuit, technology and software/application groups is key
  - System clocked latch power is the primary target
    - Advanced, fine-grain clock-gating techniques hold the promise of providing the single biggest reduction slice at minimal performance and area cost
  - Leakage reduction in active mode is of primary importance in future designs
    - Predictive power-gating holds the promise of maximum payoff
- **Power balanced architectural trade-offs are extremely important**
  - Starts with the choice of a near-optimal pipeline depth
  - Investment in *system-level* power-performance optimization rather than just processor core-centric view may have a higher payoff
    - Should include the software stack in such a system view
Concluding remarks (contd.)

- Power and temperature hot spots are a potential reliability problem in future systems; scaling trends add to other concerns: soft error rates and chip-level variability
  - Prior and existing new approaches in error tolerance are important again!
  - Variation-tolerant design will bring a key new paradigm shift
  - Other new temperature- and reliability-aware design methods are being examined in research groups
  - Accurate pre-silicon modeling of on-chip temperatures and failure rates is again the key enabler of cost-effective solutions to these hard problems

Paper Reference List: Separately Distributed
THANK YOU !!!!