Designing Cool Chips in an Era of Gigascale Integration:

*History, Challenges, and Opportunities*

Kevin Skadron

LAVA/HotSpot Lab
Dept. of Computer Science
University of Virginia
Charlottesville, VA

Luncheon presentation at Semi-Therm 21, Mar. 2005
“Cooking-Aware” Computing?
ITRS Projections

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tech node (nm)</td>
<td>100</td>
<td>70</td>
<td>45</td>
<td>32</td>
<td>22</td>
</tr>
<tr>
<td>Vdd (high perf) (V)</td>
<td>1.2</td>
<td>1.1</td>
<td>1.0</td>
<td>0.9</td>
<td>0.8</td>
</tr>
<tr>
<td>Vdd (low power) (V)</td>
<td>1.0</td>
<td>0.9</td>
<td>0.7</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>Frequency (high perf) (GHz)</td>
<td>3.0</td>
<td>6.8</td>
<td>15.1</td>
<td>23.0</td>
<td>39.7</td>
</tr>
<tr>
<td>Max power (W)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-perf w/ heatsink</td>
<td>149</td>
<td>180</td>
<td>198</td>
<td>198</td>
<td>198</td>
</tr>
<tr>
<td>Cost-performance</td>
<td>80</td>
<td>98</td>
<td>120</td>
<td>138</td>
<td>158</td>
</tr>
<tr>
<td>Hand-held</td>
<td>2.1</td>
<td>2.4</td>
<td>2.8</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

- These are targets, doubtful that they are feasible
- Growth in power density means cooling costs continue to grow
- High-performance designs seem to be shifting away from clock frequency toward # cores
Leakage – A Growing Problem

- The fraction of leakage power is increasing exponentially with each generation
- Also exponentially dependent on temperature
- Curiously, ITRS 2004 projections are lower than what industry is currently reporting
- Changes tradeoffs! Idle logic hurts, e.g. CMPs

Static power/ Dynamic Power

(Data derived from ITRS 2001)
Thermal Packaging is Expensive

- Nvidia GeForce 5900 card – “dustbuster”

Source: Tech-Report.com
Architecture Trends

- High-performance market
  - “Fat” (wide, superscalar) CPUs and high frequencies giving way to multiple cores, plateau in frequencies
    - Huge number of multi-core product announcements
    - # cores might be the next marketing buzz
  - Multiple threads per core
    - This probably won’t scale – limit of 2-4 thread contexts
  - Interesting example: Sun Niagara
    - 8 4-threaded cores

- Across all market segments
  - Growing integration (SoC)
  - Specialized co-processors and offload engines

- Growing heterogeneity
  - Part of the programming model in SoCs
  - Not part of the programming model in CMPs!
Basketball Analogy

- Recent trends in high-performance processors are like building a team around Shaq when you have a limited budget
  - Huge salary (power) to one player
  - Huge ego, team friction (heat)
  - Shaq can’t get much better (except possibly his free throws) (diminishing returns)
- New trend: multiple CPUs on a chip (CMP/SoC)
  - Don’t need superstars (less power per core, better energy efficiency)
  - Choose team players (better heat distribution)
  - Performance scales linearly with cores
  - Heterogeneous cores possible (SoCs)
  - Detroit Pistons
Talk Outline

- Different philosophies of Power-Aware design
  - Energy efficient vs. low power vs. temperature-aware
- Power Management Techniques
  - Dynamic
  - Static
  - Temperature
- Summary of Important Challenges

- *My perspective tends to be architecture-centric, and slanted toward high-performance desktop/server/etc. CPUs*
Metrics

- **Power**
  - Average power, instantaneous power, peak power
- **Energy**
  - Energy (MIPS/W)
  - Energy-Delay product (MIPS²/W)
  - Energy-Delay² product (MIPS³/W) – *voltage independent!*
    - (Zyuban, GVLSI’02)
- **Temperature**
  - Correlated with power density over sufficiently large time periods
  - No good figures of merit for trading off thermal efficiency against performance, area, or energy efficiency

- Design for power delivery
- Low-Power Design
- Power-Aware/Energy-Efficient Design
- Temperature-Aware Design
Dynamic Power
Circuit Techniques

- Transistor sizing
- Signal and clock gating
- Dynamic vs. static logic
- Circuit restructuring
- Low power caches, register files, queues

- *These typically reduce the capacitance being switched*
Clock Gating, Signal Gating

“Disabling a functional block when it is not required for an extended period”

- Implementation
  - Simple gate that replaces one buffer in the clock tree
  - Signal gating is similar, helps avoid glitches
  - Delay is generally not a concern except at fine granularities

- Choice of circuit design and clock gating style can have a dramatic effect on temperature distribution
Circuit Restructuring

- Pipeline (tolerate smaller, longer-latency circuitry)
- Parallelize (can reduce frequency)
- Reorder inputs so that most active input is closest to output (reduces switched capacitance)
- Restructure gates (equivalent functions are not equivalent in switched capacitance)

Example: Parallelizing (maintain throughput)

<table>
<thead>
<tr>
<th>Logic Block</th>
<th>Vdd</th>
<th>Freq = 1</th>
<th>Vdd = 1</th>
<th>Throughput = 1</th>
<th>Power = 1</th>
<th>Area = 1</th>
<th>Pwr Den = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Block</td>
<td>Vdd/2</td>
<td>Freq = 0.5</td>
<td>Vdd = 0.5</td>
<td>Throughput = 1</td>
<td>Power = 0.25</td>
<td>Area = 2</td>
<td>Pwr Den = 0.125</td>
</tr>
</tbody>
</table>

Source: Shekhar Borkar, keynote presentation, MICRO-37, 2004
Architectural-Level Techniques

- Sleep modes
- Pipeline depth
- Energy-efficient front end
  - Branch prediction accuracy is a major determinant of pipeline activity -> spending more power in the branch predictor can be worthwhile if it improves accuracy
- Integration (e.g. multiple cores)
- Multi-threading
- Dynamic voltage/frequency scaling
- Multi clock domain architectures (similar to GALS)
- Power islands
- Encoding/compression
  - Can reduce both switched capacitance and cross talk
- Application specific hardware
  - Co-processors, functional units, etc.
- Compiler techniques

Prevalent
Growing or Imminent
Optimal Pipeline Depth

• Increased power and diminishing returns vs. increased throughput
• 5-10 stages, 15-30 FO4
• Srinivasan et al, MICRO-35, Hartstein and Puzak, ACM TACO, Dec. 2004

![Pipeline Stages Graph](https://example.com/graph.png)
Architectural-Level Techniques

- Sleep modes
- Pipeline depth
- Energy-efficient front end
  - Branch prediction accuracy is a major determinant of pipeline activity -> spending *more power* in the branch predictor can be worthwhile if it improves accuracy
- Integration (e.g. multiple cores)
- Multi-threading
- Dynamic voltage/frequency scaling
- Multi clock domain architectures (similar to GALS)
- Power islands
- Encoding/compression
  - Can reduce both switched capacitance and cross talk
- Application specific hardware
  - Co-processors, functional units, etc.
- Compiler techniques

Prevalent

Growing or Imminent
Multi-threading

- Do more useful work per unit time
  - Amortize overhead and leakage
- Switch-on-event MT
  - Switch on cache misses, etc. (Ex: Sun Niagara “throughput computing”)
  - Can even rotate among threads every instruction (Tera/Cray)
- Simultaneous Multithreading/HyperThreading
  - For superscalar – eliminate waste
  - Intel Pentium 4, IBM POWER5, Alpha 21464
Architectural-Level Techniques

- Sleep modes
- Pipeline depth
- Energy-efficient front end
  - Branch prediction accuracy is a major determinant of pipeline activity -> spending more power in the branch predictor can be worthwhile if it improves accuracy
- Integration (e.g. multiple cores)
- Multi-threading
- Dynamic voltage/frequency scaling
  - Limits
- Multi clock domain architectures (similar to GALS)
- Power islands
- Encoding/compression
  - Can reduce both switched capacitance and cross talk
- Application specific hardware
  - Co-processors, functional units, etc.
- Compiler techniques

Prevalent
Growing or Imminent
• Basic idea is for the compiler to identify opportunities for using low-power modes

• Compiler-guided DVS
  • Reduce voltage in memory-bound program regions
    – Hsu and Kremer, ISLPED’01, PLDI’03; Xie et al, PLDI’03

• Dynamic resource configuration/hibernation
  • Deactivate modules when they won’t be used for a long time (>> sleep/wakeup time)...avoids waiting for timeout
    – Heath et al, PACT’02

• Profile/compiler-guided adaptation
  • Subroutine-guided ("positional") adaptation (Huang et al, ISCA’03)
    – Uses profiling and a hierarchy of low-power modes

• Much work in this area – this only touches the surface
Static Power Dissipation

- Static power: dissipation due to leakage current
- Exponentially dependent on $T$, $V_{dd}$, $V_{th}$
- Most important sources of static power: subthreshold leakage and gate leakage
  - We will focus on subthreshold
  - Gate leakage has essentially been ignored
    - New gate insulation materials may solve problem
Thermal Runaway

• The leakage-temperature feedback can lead to a positive feedback loop
  • Temperature increases $\Rightarrow$ leakage increases $\Rightarrow$
    temperature increases $\Rightarrow$ leakage increases $\Rightarrow$
  • ...

Source: www.usswisconsin.org
A Smorgasbord

- Transistor sizing
- Multi $V_{th}$
- Dynamic threshold voltage – reverse body bias – Transmeta Efficeon
  - Transmeta uses runtime compilation and load monitoring to select thresholds
- Stack effect
- Sleep transistors
- DVS
  - Coarse or fine grained
- Low leakage caches, register files, queues
- Techniques for reducing gate leakage
- Hurry up and wait
  - Low leakage: maintain min possible V, f
  - High leakage: use high V/f to finish work quickly, then go to sleep
Sleep Transistors

• Recent work suggests that a properly sized, low-Vth footer transistor can preserve enough leakage to keep the cell active (Li et al, PACT’02; Agarwal et al, DAC’02)
  • Great care must be taken when switching back to full voltage: noise can flip bits
  • Extra latency may be necessary when re-activating
• Similar to principles in sub-threshold computing
  • Ex – sensor motes for wireless sensor networks
• Concerns about susceptibility to SEU
A Smorgasbord

- Transistor sizing
- Multi $V_{th}$
- Dynamic threshold voltage – reverse body bias – Transmeta Efficeon
  - Transmeta uses runtime compilation and load monitoring to select thresholds
- Stack effect
- Sleep transistors
- DVS
  - Coarse or fine grained
- Low leakage caches, register files, queues
- Techniques for reducing gate leakage
- Hurry up and wait
  - Low leakage: maintain min possible V, f
  - High leakage: use high V/f to finish work quickly, then go to sleep
Temperature
Worst-Case leads to Over-design

- **Average case** temperature lower than worst-case
  - Aggressive clock gating
  - Application variations
  - Underutilized resources, e.g. FP units during integer code

- **Currently 20-40% difference**

Source: Gunther et al, ITJ 2001
Temporal, Spatial Variations

Temperature variation of SPEC applu over time

Localized hot spots dictate cooling solution
Temperature-Aware Design

- Worst-case design is wasteful

- Power management is not sufficient for chip-level thermal management
  - Must target blocks with high power density
  - When they are hot
  - Spreading heat helps
    - Even if energy not affected
    - Even if average temperature goes up
  - This also helps reduce leakage
Role of Architecture?

- **Dynamic thermal management (DTM)**
  - Automatic hardware response when temp. exceeds cooling
  - Cut power density at runtime, on demand
  - Trade reduced costs for occasional performance loss
- **Architecture natural granularity for thermal management**
  - Activity, temperature correlated within arch. units
  - DTM response can target hottest unit: permits fine-tuned response compared to OS or package
  - Modern architectures offer rich opportunities for remapping computation
    - e.g., CMPs/SoCs, graphics processors, tiled architectures
    - e.g., register file
- **Thermal engineering must consider role of architecture**
- **Thermal engineers and architects need to collaborate**
Existing DTM Implementations

- Intel Pentium 4: Global clock gating with shut-down fail-safe
- Intel Pentium M: Dynamic voltage scaling
- Transmeta Crusoe: Dynamic voltage scaling
- IBM Power 5: Probably fetch gating
- ACPI: OS configurable combination of passive & active cooling

These solutions sacrifice time (slower or stalled execution) to reduce power density

Better: a solution in “space”
  - Tradeoff between exacerbating leakage (more idle logic) or reducing leakage (lower temperatures)
Alternative: Migrating Computation

This is only a simplistic illustrative example.
Space vs. Time

- Moving the hotspot, rather than throttling it, reduces performance overhead by almost 60%.

The greater the replication and spread, the greater the opportunities.
Future Challenges
Sources of Variations

Random Dopant Fluctuations

Sub-wavelength Lithography

Heat Flux (W/cm²)
Results in Vcc variation

Temperature Variation (°C)
Hot spots

Source: Mark Bohr, Intel
Source: Shekhar Borkar, keynote presentation, MICRO-37, 2004
Impact of Static Variations

Frequency ~30%
Leakage Power ~5-10X

Source: Shekhar Borkar, keynote presentation, MICRO-37, 2004
Parameter Variations

- Parameter variations mess everything up!
- $T \Rightarrow$ variation in $V_{cc}$, leakage $\Rightarrow T$
- $V_{cc} \Rightarrow$ speed variation, leakage $\Rightarrow T$
- Manufacturing ($L$, $W$, $V_{th}$, etc) $\Rightarrow$ speed, $V_{cc}$, $T$
- Packaging variations ($TIM$, roughness) $\Rightarrow T$

- Some transistors/functional units won’t work, some will be lousy, some will fail over time, and some will be intermittent

- Guard banding won’t work
  - Design devolves to worst component, can’t easily bound intermittent behavior
- $T/P$ problems may no longer be limited to specific units
- Makes dynamic logic even more difficult
Future Architectures

- Asymmetry unavoidable
  - Specialized units (part of programming model)
  - Power management (can try to hide this)
  - Thermal throttling (hard to hide this)
  - Parameter variations (hard to hide this without extreme performance loss)
Future Architectures

- Increasing integration, e.g. increasing # cores, e.g. Niagara
- Clustered architectures
- Tiled architectures
- Multiple voltage islands

- Asymmetry unavoidable
  - Specialized units (part of programming model)
  - Power management (can try to hide this)
  - Thermal throttling (hard to hide this)
  - Parameter variations (hard to hide this without extreme performance loss)
  - Increasing problems with yield, failures in time (Redundancy: costly; graceful degradation: introduces asymmetry)
Power and Thermal Security

- A consequence of designing for expected rather than worst-case conditions
- Energy-drain attacks
- Voltage stability attacks (\(\frac{dI}{dt}\))
- Thermal attacks
  - Thermal throttling
  - Denial of service
  - Direct physical damage
- [Puyan and Skadron, Semi-Therm 2005]
Summary

• Reviewed current techniques for managing dynamic power, leakage power, temperature
  • A major obstacle with architectural techniques is the difficulty of predicting performance impact

• Continuing integration makes power an ever-present concern

• Thermal limits and parameter variations are becoming serious obstacles
  • Spread heat in space, not time
  • Architecture can develop pipelines and structures that adapt to manage static, dynamic variations

• Security challenges
Soap-Box

• Architecture solutions are essential
• Thermal engineers, circuit designers, CAD designers, and architects all need to work together
• Joint infrastructure
  • Simulators – esp. pre-RTL tools
    – Parameter variations are especially challenging to model
  • Test chips
    – Ex: Combine architecture and circuit research on a single test chip
questions
More Info

http://www.cs.virginia.edu/~skadron

LAVA Lab