# Reducing Power Density through Activity Migration *

Seongmoo Heo, Kenneth Barr, and Krste Asanović
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
{heomoo,kbarr,krste}@cag.lcs.mit.edu

## ABSTRACT

Power dissipation is unevenly distributed in modern microprocessors leading to localized hot spots with significantly greater die temperature than surrounding cooler regions. Excessive junction temperature reduces reliability and can lead to catastrophic failure. We examine the use of activity migration which reduces peak junction temperature by moving computation between multiple replicated units. Using a thermal model that includes the temperature dependence of leakage power, we show that sustainable power dissipation can be increased by nearly a factor of two for a given junction temperature limit. Alternatively, peak die temperature can be reduced by 12.4 $^o$C at the same clock frequency. The model predicts that migration intervals of around 20–200 $\mu$s are required to achieve the maximum sustainable power increase. We evaluate several different forms of replication and migration policy control.

## Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Advanced Technologies, Microprocessors and Microcomputers, VLSI*

## General Terms

Performance, Design, Reliability

## Keywords

Thermal Model, Activity Migration, Temperature Reduction

## 1. INTRODUCTION

Power dissipation is emerging as a key constraint on achievable processor performance. Elevated die temperatures reduce device reliability and transistor speed, and increase leakage currents exponentially. Providing adequate heat removal with low cost packaging is particularly challenging as processor power densities rise above 100 W/cm$^2$. Moreover, power dissipation is unevenly distributed across a microprocessor die. Highly active portions of the design such as the issue window or execution units may have more than twenty times the power density of less active blocks such as a secondary cache [5]. Even with dynamic thermal management, these hot spots limit performance because total power dissipation must be reduced until all hot spots have acceptable junction temperatures.

In this paper, we evaluate the use of *activity migration* to reduce power densities. Hot spots develop because silicon is a relatively poor heat conductor and cannot spread heat efficiently across a die. With activity migration (AM), we instead spread heat by transporting computation to a different location on the die. Computation proceeds in one unit until it heats past a temperature threshold. The computation is then transferred to a second unit allowing the first to cool down.

We show how AM can be used either to double the power that can be dissipated by a given package, or to lower the operating temperature and hence the operating power. Our thermal model predicts that the intervals between migrations needed to obtain maximum benefit can be much smaller than typical operating system context swap times, and will scale to smaller intervals in future technologies. We evaluate the performance impact of such small migration intervals, and examine various alternative architectural configurations that trade area for power/performance.

## 2. THERMAL MODEL

In this section, we develop a thermal model we later use to evaluate activity migration. We are primarily interested in the case where performance is constrained by thermal packaging and hence we are only worried about the worst case hot spot. When a processor is in thermal crisis, one block (for example, the scheduler or the regfile) will hit the peak temperature first. The remaining blocks will have lower power densities, with the average power density across the whole die being much lower than the hot spot.

### 2.1 Thermal and Process Properties

Table 1 shows the thermal properties and process technology data we used in this model. We present two technology cases: a present mature technology (Current) and a near-future technology (Future). The transistor models for 180 nm and 70 nm processes were based on TSMC 180 nm and BPTM 70 nm [2] processes respectively. Because most heat is removed through the back of the die, an important parameter in the thermal properties of microprocessors is die thickness. Wafers are approximately 750 $\mu$m thick during manufacture, but are then thinned to improve thermal properties. Smart card chips are already thinned to 100 $\mu$m thicknesses to reduce their size, and commercial vendors offer thinning to 50 $\mu$m. For the current case, we assume a 250 $\mu$m thickness, for the future case we assume the wafer is thinned to 100 $\mu$m.

*This work was funded by DARPA PAC/C award F30602-00-2-0562, NSF CAREER award CCR-0093354, and a donation from Intel Corporation.


*ISLPED'03,* August 25–27, 2003, Seoul, Korea.

| | Symbol | Current Case | Future Case |
|---|---|---|---|
| Die thickness ($\mu$m) | $t$ | 250 | 100 |
| Die conductivity (W/K/m) | $k$ | 100 | 100 |
| Die specific heat (J/K/$m^3$) | $c$ | 1e6 | 1e6 |
| Die area ($mm^2$) | $A_{die}$ | 100 | 100 |
| Hot spot area ($mm^2$) | $A_{block}$ | 2 | 2 |
| Hot spot active pwr. density (W/$mm^2$) | $PD_{act}$ | 5 | 7.5 |
| Hot spot leak. pwr. density (110$^o$C) | $PD_{leak}$ | 0.015 | 0.15 |
| Isothermal point ($^o$C) | $T_{iso}$ | 70 | 70 |
| Channel Length (nm) | $L$ | 180 | 70 |
| Supply Voltage (V) | $V_{DD}$ | 1.5 | 1.0 |
| NMOS threshold (V) | $NV_{th0}$ | 0.269 | 0.120 |
| PMOS threshold (V) | $PV_{th0}$ | -0.228 | -0.153 |

**Table 1: Process technology, thermal properties and transistor data.**

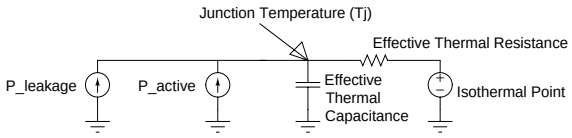


**Figure 1: Thermal model: an equivalent RC circuit. Isothermal Point is the temperature of the temperature source located at the package.**

The hot spot active power density numbers were chosen to be aggressive for current technology, at 5 W/mm$^2$. Under ideal scaling theory, power density remains constant. However, we expect power density to scale higher in the future as voltages will not scale down as rapidly as feature size, and clock frequency is increasing faster than transistor speed due to deepening of pipelines. We chose 7.5 W/mm$^2$ as the scaled future power density.

## 2.2 Equivalent RC Thermal Model

Junction temperature is modelled by a simple first-order equivalent RC circuit (Figure 1) using the well-known analogy between electrical circuits and thermal models [10]. Temperature and power are represented by voltage and current respectively.

Vertical thermal resistance comes from silicon and packaging and we can assume that those two resistances are connected in series.

$$R_{silicon,vertical} = \frac{t}{k \times A_{block}} \quad (1)$$

$$R_{package,vertical} = 120 \times \frac{t}{k} \times \frac{A_{die}}{A_{block}}[1] \quad (2)$$

$$R_{total,vertical} = (1 + 120 \times A_{die}) \times \frac{t}{k \times A_{block}} \quad (3)$$

Barcella et al show that this simple empirical formula for $R_{package}$, thermal resistance from the die to isothermal point in the package, matches the time-consuming 3D simulation results well [1].

Most heat is dissipated vertically in a chip, particularly when wafers are thinned, since horizontal cross-section is usually much smaller than the area of a typical hot spot. Therefore, we assumed infinite lateral thermal resistance, which will lead to the worst-case temperature gradients. Deeney [5] notes that lateral conductance can be used to spread the heat of hot spot and suggests the use of blank silicon around hot spots, but those numbers were for an unthinned 750 $\mu$m HP PA-8700 die. As die thickness drops, vertical conductivity improves and lateral conductivity decreases. The heatsink also provides a low thermal resistance path to spread heat

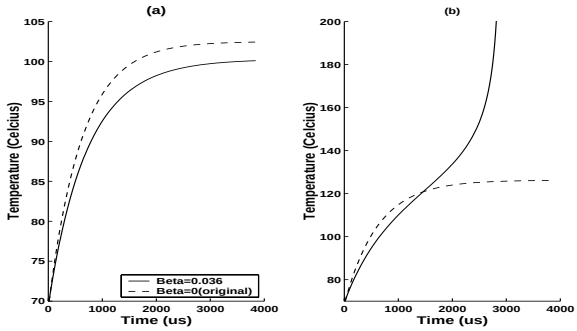


**Figure 2: Comparison of thermal models for Current case. We assume that $T_j$ starts from $T_{iso}$. Different active and leakage power conditions are assumed for simulation (a) and (b).**

across the die, but this is accounted for by the selection of the isothermal point.

Thermal capacitance of the die can be derived as

$$C_{silicon} = c \times t \times A_{block} \quad (4)$$

Thermal capacitance of the packaging was ignored, because the package capacitance is so massive that it effectively acts as a temperature source.

## 2.3 Temperature Dependency of Leakage

Leakage power is a significant part of total power for modern processes, and is exponentially dependent on temperature. This dependency was measured by simulating an inverter chain with Hspice for 180 nm and 70 nm processes, and then the results were curve-fitted to an exponential function($P_{leakage} = P_{leakage0} \times e^{\beta \times (Temp - Temp0)}$). Our simple exponential model agrees to the Hspice results to within 10% across temperatures of interest, with $\beta$ of 0.036 and 0.017 for 180 nm and 70 nm processes respectively.

We modeled leakage power in our thermal model as a voltage-dependent current source:

$$P_{leak} = P_{leak110} \times e^{\beta(T_j - 110)} \quad (5)$$

where $P_{leak110}$ is leakage power at 110 $^o$C.

Figure 2 compares the simulation results for a traditional thermal model with constant leakage power at the worst-case temperature 110 $^o$C and our temperature-dependent thermal model. In Figure 2 (a), we can see that the constant leakage model overestimates the junction temperature. Figure 2 (b) shows the thermal runaway phenomenon. Thermal runaway is caused by the exponential dependency of leakage power on temperature: Increased temperature increases leakage power, and increased total power (switching + leakage) increases temperature. This positive feedback loop can cause circuit failure. We only observed this phenomenon when certain conditions between active power and leakage power at certain temperature and $\beta$ are met. A constant leakage model cannot predict the thermal runaway phenomenon.

# 3. ACTIVITY MIGRATION

In this section, we use the thermal model to evaluate the possible gains from activity migration. We only consider the case where we implement activity migration by pingponging between two sites. More sites could be used and would give further benefits but would increase area overhead.

By pingponging between two sites, the activity for each site is halved, which means half the switching power density and half the

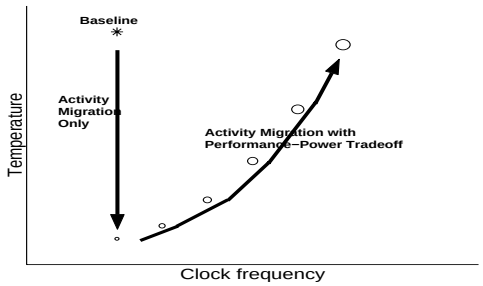


**Figure 3: Conceptual benefits of Activity Migration: reduced temperature or increased frequency.**

temperature increase from the isothermal point. The same benefit could be achieved without pingponging if the computational load could be parallelized across the two sites. We are primarily interested in improving single thread performance and so do not consider parallel execution further.

Reduced power density could be used either to increase performance or to reduce temperature. Figure 3 shows the qualitative benefit of activity migration in terms of new operating points. First, we can lower die temperature at the same clock frequency, which will also lead to lower leakage power. We can also exploit the time slack due to cooler operation by lowering $V_{DD}$ slightly to save some active power. Second, we can burn more power to increase clock frequency while maintaining lower temperature than the baseline.

As an example of how this can be used, consider a laptop with limited heat removal capability. When running from battery power, we can use AM to lower die temperature which lowers leakage and allows more energy-efficient execution. When plugged into a wall socket, we can use activity migration to allow more power to be burned to raise performance without raising die temperature.

There are many techniques that provide power-performance tradeoffs such as dynamic voltage scaling (DVS), dynamic cache configuration modification, fetch/decode throttling, or speculation control [3]. We selected DVS as it is the simplest technique to illustrate the potential benefit of AM. To get the maximum benefit from the frequency-power tradeoff, supply voltage and threshold voltage should be scaled simultaneously. But for circuits with high activity factor, dynamic supply voltage scaling alone is adequate, and hence we assume fixed threshold voltages.

## 3.1 Model

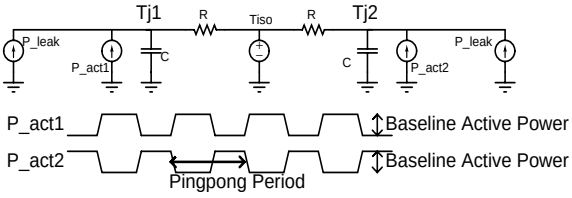


**Figure 4: Pingpong thermal model : an equivalent RC circuit. R, C, and P_leak are the same as the baseline thermal resistance, thermal capacitance, and leakage power.**

Figure 4 shows the equivalent RC circuit for our AM technique. When a circuit is in thermal crisis, the best thing to do is to pingpong between the two sites to cool the hot spot. To determine the maximum achievable benefit, the duty cycle of pingponging is fixed at 50% in our results below. In practice, migration would be triggered by thermal sensors, giving rise to a variable duty cycle. However, in future designs as absolute power grows and the gap between maximum and typical power dissipation grows, we expect dynamic thermal management schemes will try to keep a thermal-constrained active processor on the verge of over heating (otherwise we're wasting money on packaging), leading to duty cycles of around 50%. Energy-constrained systems, will similarly obtain the lowest die temperature by alternating between the two sites.

We can calculate the approximate benefits of activity migration technique analytically (Figure 5). For the simplicity of computation, it is assumed that temperature changes exponentially.

$$T_{high} = (T_{low} - T_{base})e^{-\frac{Period}{2\tau}} + T_{base} \quad (6)$$

$$T_{low} = (T_{high} - T_{iso})e^{-\frac{Period}{2\tau}} + T_{iso} \quad (7)$$

$\tau$ is the time constant of the equivalent RC circuit. Solving for $T_{high}$:

$$T_{high} = \frac{T_{base} - T_{iso}}{1 + e^{-\frac{Period}{2\tau}}} + T_{iso} \quad (8)$$

The temperature increase from the isothermal point is decreased by $1 + e^{-\frac{Period}{2\tau}}$ through AM. With very small $Period$, the temperature increase will be halved. On the other hand, we can increase the power by a factor $1 + e^{-\frac{Period}{2\tau}}$, which makes $T_{high}$ the same as $T_{base}$. If $Period$ is considerably smaller than $\tau$, we can double power at the same temperature as the baseline, which means increasing the clock frequency by a factor of $2^{\frac{1}{3}} = 1.26$ with DVS since active power is approximately proportional to $Vdd^3$.

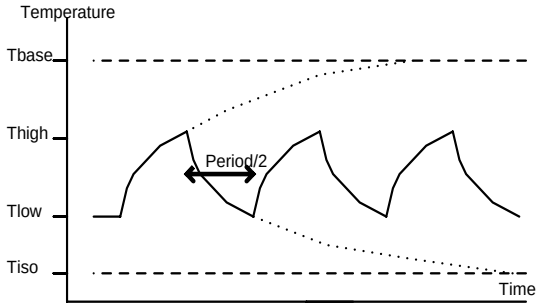


**Figure 5: Analytical calculation of benefits of Activity Migration. $T_{base}$ is the baseline temperature, $T_{iso}$ is the isothermal point, $T_{high}$ and $T_{low}$ are the upper-bound and lower-bound temperatures for AM technique, and $Period$ is AM period.**

## 3.2 Simulation Results

### *3.2.1 Reducing Temperature and Power*

Table 2 shows the SPICE simulation results of the AM technique for the assumed hot spot block with various pingpong periods.

AM drops temperature more than 12 $^o$C (7 $^o$C) for the Current (Future) case at the same clock frequency, which is almost half the baseline temperature increase from the isothermal point. AM reduces leakage and active power consumption. Reduced temperature leads to reduced leakage power, more than 37% (12%) for Current (Future) case. Reduced latency due to increased on-state drain current at lower temperature can be exploited by reducing $V_{DD}$ to save active power. Almost a 10% power reduction was attained for both cases.

| | Current Case | | | Future Case | | |
|---|---|---|---|---|---|---|
| Pingpong period ($\mu$s) | 1800 | 600 | 200 | 600 | 200 | 60 |
| Temperature drop (K) | 9.2 | 11.5 | 12.4 | 3.4 | 6.4 | 7.5 |
| Leak power reduction (%) | 29.6 | 35.3 | 37.6 | 5.9 | 10.8 | 12.6 |
| Act power reduction (%) | 3.7 | 7.6 | 9.7 | 3.3 | 9.5 | 9.7 |

**Table 2: Simulation measurements of AM for the hot spot block. When calculating temperature drop and power reduction, the upper-bound AM temperature was used.**

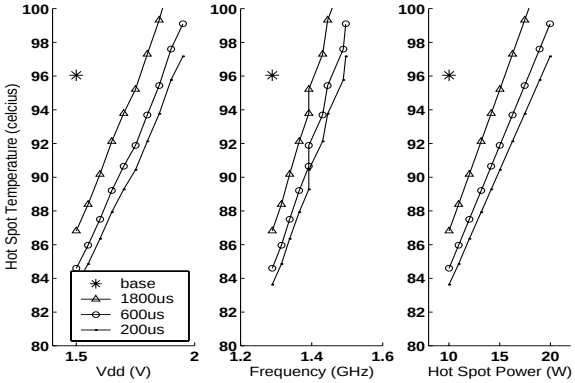


**Figure 6: Simulation results of AM and DVS for various pingpong periods for the hot spot block (Current case).**

### *3.2.2 Increasing Clock Frequency*

Figure 6 and Figure 7 show the simulation results for AM with DVS. DVS was modelled based on the Hspice simulation of a 15-stage ring-oscillator. We assumed 30 times FO1 delay, the period of the ring-oscillator, as the clock period of the processor. Active power, leakage power, and period were measured changing $V_{DD}$ for both technology cases. Activity factor was assumed to be as high as 0.2 since usually high activity areas cause hot spots.

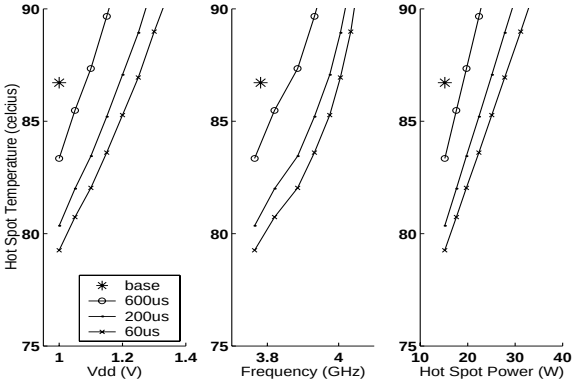


**Figure 7: Simulation results of AM and DVS for various pingpong periods for the hot spot block (Future case).**

Dynamic voltage scaling exploits the temperature drops by AM to achieve a frequency increase at the same or lower temperature. Table 3 shows the clock frequency and power increase at the same temperature as the baseline through AM with DVS. Around 16% (6%) clock frequency increase could be attained for the Current (Future) case.

| | Current Case | | | Future Case | | |
|---|---|---|---|---|---|---|
| Pingpong period ($\mu$s) | 1800 | 600 | 200 | 600 | 200 | 60 |
| Clock freq incr (%) | 10.5 | 14.1 | 15.9 | 2.3 | 5.0 | 5.9 |
| Power incr (%) | 56.8 | 79.5 | 90.9 | 25.0 | 61.4 | 79.6 |

**Table 3: Summary of effects of AM with DVS for the hot spot block.**

# 4. AM ARCHITECTURES

Activity migration incurs a variety of overheads. Each migration event can cause a performance loss as data is copied to the new activity site. There can be further performance loss between migration events as the additional wiring required to connect replicated units adds to pipeline latencies. This additional wiring can also increase power consumption. There is an area overhead for the replicated units. The additional circuitry can add to leakage power, but the increase is small because the idle unit is at lower temperature than the active unit and consequently has much lower leakage. The leakage power of the idle unit could also be reduced further by placing it into a low leakage state using techniques such as sleep vectors, sleep transistors, or body biasing.

In this section, we evaluate a number of different architectural configurations for achieving activity migration in a superscalar processor. Such an evaluation is important to ensure that the clock frequency increase permitted by AM is not outweighed by its potential CPI penalties.

## 4.1 AM Architectural Configurations

We developed five different AM architectural configurations shown in Figure 8 to examine performance-area tradeoffs. In each configuration the hot spot is assumed to be the execution core.

The baseline configuration is a single complete processor core and primary caches. Each configuration A–D has split execution cores, and each core contains its own register file and functional units. Though a structure may be shared by two execution cores, its size remains the same: as only one core can access it at a time, no additional ports are needed. Machine A contains duplicates of all structures in the baseline machine. Machine B consists of two cores sharing the same D-Cache while in Machine C the cores share both I-Cache and D-Cache. When D-Cache is shared, we pessimistically add an extra cycle to the cache access time to account for driving a longer bus. When I-Cache is shared, we pessimistically add a cycle to the branch-misprediction penalty to account for the increased pipeline latency before a branch misprediction can be detected by the execution core. Machine D is similar to Machine C except that it shares an issue queue and rename table so the pipeline need not be drained upon thermal interrupts. In this configuration, only the contents of the physical register file need to be copied for which a 32 cycle penalty is assessed twice every pingpong period (16 cycles of pipeline drain plus 16 cycles spent copying four values per cycle). We increase the branch mispredict latency by one cycle for machine D to account for additional wire delay between the front-end and the two distinct register files.

Each core's caches are 32 KB and require two cycles to access. We also experimented with single-cycle, 8 KB caches (not shown) and found similar trends for applications which fit in the cache. The sizes of each block in the figure roughly correspond to the sizes of the same structures on the Alpha 21264 as measured from a published floorplan [6]. We represent 32 KB caches in the figure by halving the size of the Alpha's 64 KB caches.

When half the pingpong period has elapsed, the pipeline is drained to prepare for execution on the inactive core. Before a core

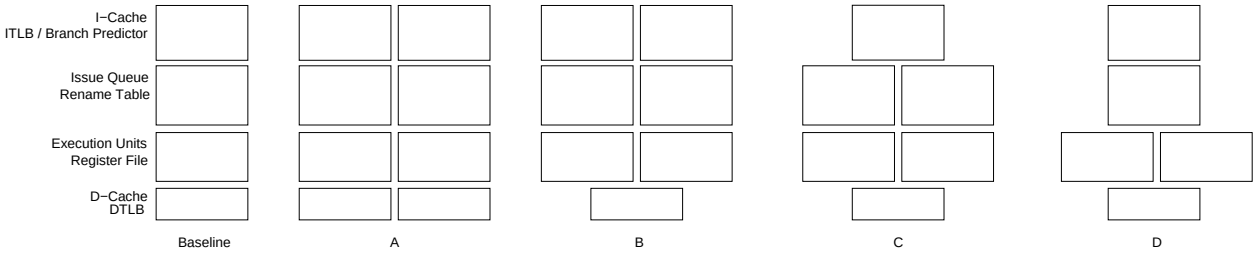


**Figure 8: Pingpong Configurations**

becomes inactive, its structures should be put into a low-leakage, data-preserving idle state. If a core is powered-off so that it loses state, the performance penalty often exceeds that provided by the clock speed increase. (We do not show these results here due to space limitations).

In configurations with split D-Caches, the soon-to-be-inactive D-Cache may contain dirty lines which need to be taken into account. We simulate two policies for handling this condition. In the "naive" policy, all dirty lines are written back to the next level cache in parallel with updating lines in the the soon-to-be-active cache before execution resumes. We assume this takes a constant time of 2 cycles per line as the duplicate cores share the same inclusive level 2 cache. Alternatively, a cache-coherence protocol can be used to transfer lines on demand. The inactive cache participates in the cache coherence protocol to move lines to the active cache.

Activity migration can be achieved either in software or in hardware. If we perform AM between multiple processor cores on a die, then long pingpong periods would be easy to handle in software as part of a regularly scheduled OS context swap, but Table 3 shows that small pingpong periods are necessary to realize performance gains. According to our thermal model, the Future case has a time constant more than six times smaller than that of the Current case since thermal $RC$ is proportional to the square of die thickness. It is also clear that the optimal pingpong intervals are much shorter than typical OS context swap times of 1–10 ms, suggesting that it might not be efficient to control pingponging at the operating system level, and hardware or firmware level pingponging might be necessary. We assume a hardware pingpong scheme in our results below.

## 4.2 Methodology

To investigate the performance impact of AM, we used the SimpleScalar simulator version 3.0b [4] to model AM architectures for a four-issue out-of-order superscalar processor. The Simplescalar simulation parameters are given in Table 4.

| | |
|---|---|
| Pipeline Width | 4 |
| # of RUUs | 64 |
| Load/Store Queue Depth | 64 |
| # of Integer ALUs (Mult/Div) | 5 (1) |
| # of FP ALUs (Mult/Div) | 3 (2) |
| # of Load/Store Units | 2 |
| Instruction length | 32 Bits |
| L2 (64B Blocks) | 2MB/Direct mapped/7cycles |
| L1 I$/D$ (64B Blocks) | 32KB/4Way/2cycles |
| Branch Prediction | 1024 entries combined:<br>$2^{11}$ entry bimodal<br>$2^{10}$ entry GAp (8 bit global history) |
| Misprediction Latency | 3 cycles |
| Memory Latency | First 100 cycles, Next 2 cycles |

**Table 4: Simulated Processor Configuration**

SimPoints [9] were used to fast-forward benchmark initialization phase and choose a representative 100 Million-Cycle section of each program. To further reduce simulation time, a subset of the SPEC2000 benchmarks [11] was chosen to find the upper and lower bound on performance change. The integer benchmarks (`vpr`, `mcf`, `perkbmk`, and `gap`) include those with the highest and lowest miss rates for reasonably-sized L1 I-Cache and L1 D-Cache as reported by [8]. The SPEC Floating point benchmarks all tend to have low instruction cache miss rates; the floating point benchmarks `art` and `lucas` are two which reach their SimPoint relatively quickly and have contrasting data cache miss rates. Details of each benchmark's performance on our baseline machine are noted in Table 5. In all cases, L1 I-Cache miss rate is less than 1%. The benchmarks were compiled with optimization on an Alpha 21264 with Digital's C compiler (V5.9-008) and Compaq Fortran (V5.3-915) [12].

| | art-110 | lucas | gap | perl | mcf | vpr-rte |
|---|---|---|---|---|---|---|
| CPI | 1.04 | 1.15 | 0.47 | 0.55 | 3.52 | 1.08 |
| L1 D$ Miss Rate | 0.32 | 0.10 | 0.00 | 0.00 | 0.21 | 0.06 |

**Table 5: Benchmark Characteristics.**

A short pingpong period favors thermal considerations, but increases the frequency of overhead-causing events. A long pingpong period favors architectural performance as there are fewer writebacks and pipeline drainings. It is unnecessary to simulate performance with a pingpong period greater than 200,000 cycles as any overhead is amortized over the large number of cycles.

# 5. OVERALL RESULTS AND DISCUSSION

Figure 9 shows the architectural performance effect of activity migration. The policies referred to in Section 4.1 are denoted by shaded bars and labelled in the format "Machine: I-Cache/D-Cache" (e.g., A: i-split/d-naive describes the split I-Cache of machine A with the "naive" policy for the D-Caches upon pingponging). Architectural performance is measured in cycles-per-instruction normalized to a non-pingponged architecture.

Through AM with DVS, clock frequency can be increased by allowing more power, but there is a CPI penalty. Therefore, net performance should be represented by $\frac{clock\ frequency}{CPI}$. The area increase is the main penalty of the AM technique. Table 6 shows the tradeoff between area and performance with AM and DVS. Best case performance is shown for machine A.

For the Current case, configuration D is the best choice. With only 30% area increase, we can increase performance by 12% and maintain the same temperature. The core area increase would be reduced much further if more modern microprocessors with level 2 or 3 caches are considered. For example, if the Alpha 21364, which is an Alpha 21264 core with on-chip L2 cache, is used as a base model, the area increase is reduced to 6%. Alternatively,

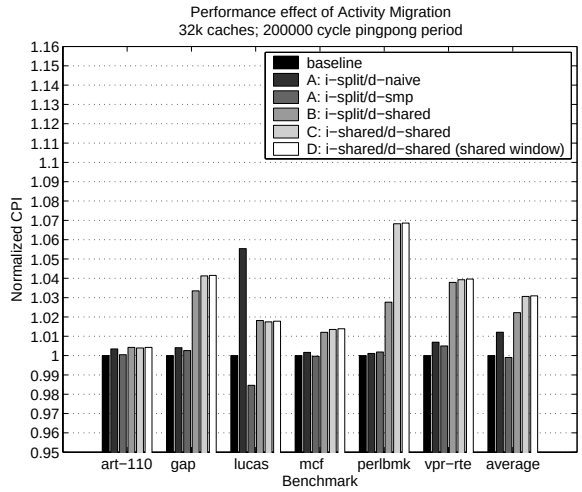


**Figure 9: CPI with 32KB Caches.**

| | Current Case | | | | Future Case | | | |
|---|---|---|---|---|---|---|---|---|
| Config. | A | B | C | D | A | B | C | D |
| Area | 2.00 | 1.84 | 1.56 | 1.30 | 2.00 | 1.84 | 1.56 | 1.30 |
| Perf. | 1.16 | 1.13 | 1.12 | 1.12 | 1.06 | 1.04 | 1.03 | 1.03 |

**Table 6: Effects of AM for area and performance normalized to baseline. Data from Figure 9 and Table 3 was used assuming 200 $\mu$s (200,000 cycles) for Current Case and 60 $\mu$s (200,000 cycles) for Future Case pingpong periods.**

we can cool down the hot spots significantly and save some power at slightly reduced performance due to IPC decrease. As for the Future case, configuration D is again the best choice. However, the performance gain by AM and DVS is relatively small, so we might prefer to use AM only to cool down hot spots.

Another penalty of AM is the extra power for driving increased wire lengths. For example, for configuration B, the connection between the D-cache and an execution unit is longer than that of the baseline. However, we found that the extra wire power is negligible compared to total power, in fact, less than 50 mW even under the worst-case estimates (4 mm increase of 64 top-layer metal wires, as high as 300 fF/mm wire cap, and as high as 0.2 activity factor).

Although we model a fixed pingpong interval, in practice, pingponging will likely be triggered by thermal sensors. This should alleviate any problems caused by an application having execution phases whose period matches that of the pingponging.

# 6. RELATED WORK

An alternative approach to reducing power density is to spread out blocks (such as a register file), but this is not an attractive option as it increases latency and switching power consumption to drive the longer wires. With activity migration, we can make the longer wires be infrequently used and out of the critical paths (as when we pingpong between whole cores). As mentioned in Section 2, using empty space on the die can help spread heat [5] for thicker wafers, but with thinner wafers this becomes much less effective. Another advantage of pingponging, rather than spreading out a single core, is that the two cores can be run in parallel to increase throughput when an application has the appropriate form of parallelism.

The only reference to a technique similar to activity migration we have found was in an Intel press release [7]. Intel describes a scheme, "core hopping," which appears to be similar to the first configuration we evaluated. As shown above, swapping threads at OS timescales could be too slow to get maximum benefit and has a large area overhead. Time constants also shrink in absolute time in future technologies.

# 7. CONCLUSION

We have examined the use of activity migration, which reduces peak junction temperature by moving computation between multiple replicated units. Using a thermal model that includes the temperature dependence of leakage power, we show that sustainable power dissipation can be increased by nearly a factor of two for a given junction temperature limit. This increased power capacity can permit dynamic voltage scaling to achieve nearly 16% clock frequency increase using 180 nm technology. The model predicts that migration intervals of around 20–200 $\mu$s are required to achieve the maximum sustainable power increase, and that this interval will scale down with future fabrication technologies. We have introduced several architectural alternatives for implementing activity migration in an out-of-order superscalar processor, and simulated the performance impact of activity migration. Lost cycles to support activity migration cause only a 2% IPC drop on average, leaving a net performance gain of 12%. Alternatively, peak die temperature can be reduced by 12.4 $^o$C resulting in reduced active and leakage power dissipation. Our lowest area overhead configuration requires an estimated area increase of 6% to accommodate an additional register file and execution units for a processor with on-chip caches.

# 8. ACKNOWLEDGMENTS

We'd like to thank Christopher Batten, Ronny Krashinsky, Heidi Pan, and the anonymous reviewers for comments on earlier drafts.

# 9. REFERENCES

[1] M. Barcella et al. Architecture-level compact thermal R-C modeling. Technical Report CS-2002-20, Univ. of Virginia Comp. Sci. Dept., Jul. 2002.

[2] Predictive technology model. Technical report, UC Berkeley, 2001.

[3] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, Jan. 2001.

[4] D. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. Technical Report CS-TR-97-1342, Univ. of Wisconsin Comp. Sci. Dept., Jun. 1997.

[5] J. Deeney. Thermal modeling and measurement of large high-power silicon devices with asymmetric power distribution. In *IMAPS*, Sep. 2002.

[6] B. A. Gieseke et al. A 600MHz superscalar RISC microprocessor with out-of-order execution. *ISSCC*, pages 176–177, Feb. 1997.

[7] Intel. Intel to introduce new technologies to reduce power consumption of mpus, Aug. 2002. http://www.esi-online.com.sg/news/view/default.asp?newId=10.

[8] S. Sair and M. Charney. Memory behavior of the SPEC2000 benchmark suite. Technical Report RC 21852, IBM, Oct. 2000.

[9] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *ASPLOS*, Oct. 2002.

[10] K. Skadron et al. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *HPCA*, Feb. 2002.

[11] Standard Performance Evaluation Corp. CPU2000, 2000.

[12] C. Weaver. SPEC2000 Alpha binaries (little endian).