Temperature-Aware Design Issues for SMT and CMP Architectures

James Donald and Margaret Martonosi
Department of Electrical Engineering
Princeton University, Princeton, NJ
{jdonald,martonosi}@princeton.edu

Abstract

With increasing power density in modern processors, management of on-chip temperature is fast becoming a bottleneck for chip designers. To address this, beyond conventional power and energy analysis it is necessary to apply temperature-aware analysis. In this paper we present thermal-aware experiments on simultaneous multithreaded (SMT) and chip multi-processor (CMP) architectures. Both SMT and CMP have been shown to improve ILP as well as energy efficiency, but in our experiments we also examine the temperature consequences of such multithreaded architectures. We use the SimpleScalar tool set combined with Wattch for power measurement in conjunction with the HotSpot thermal modeling tool. We begin with models for different processors using roughly equal silicon resources and develop parameters and floorplan layouts for each of these cases. Our findings show that large temperature gradients are prominent with either multithreading technique, but both architectures show promise as a basis for temperature-aware enhancements to mitigate the problem. We examine several techniques for managing peak temperature problems and find that allowing hot functional blocks to be allocated more die area can reduce our processors' hottest unit temperatures by as much as 12° Celsius. We scale our processors up to 4 contexts or 4 processor cores and find that these same temperature trends continue.

1. Introduction

Simultaneous multithreading and chip multi-processing were originally proposed as methods to increase microprocessor performance [10, 26]. Recently, studies have also shown that both techniques are capable of reducing overall power consumption [20, 21]. Until now, however, there has yet to be done an analysis comparing how each design methodology fares in the area of temperature-aware design.

With increases in density of digital circuits without commensurate reduction in density of power consumption, heat dissipation is fast becoming the limiting factor in microprocessor performance [24]. In order to examine specifics of this problem, temperature analysis brings in the requirement of understanding how the output heat is spatially distributed across a die to result in temperature gradients.

Microarchitectural methods to increase ILP have begun to involve steering toward multithreaded processors in the form of SMT and CMP, instead of continually increasing superscalar issue width. Because of the growing prominence of multithreaded processors, we perform an in-depth temperature analysis to examine the long term effects as processors move towards processing a greater number of threads. Some critical questions we wish to answer are:

- Does either of the two processor design paradigms inherently give better thermal management alongside performance and power efficiency consequences?
- With multithreading will thermal hotspots become even more of a problem?
- And do thermal management techniques such as migration of computation retain their utility as we continue to scale up the number of threads or processor cores?

In the next section we give brief overviews of the SMT and CMP concepts. Section 3 describes our simulation tools, benchmarks, and methodology. Section 4 summarizes our experiment results. Section 5 discusses future work, and Section 6 concludes.

2. Background and related work

Simultaneous multithreading (SMT) was first described by Tullsen et al. as a technique by which processor resources could be more efficiently utilized

due to issuing instructions from more than one thread [26]. SMT requires duplication of some essential resources such as architectural register files in order to maintain multiple thread contexts, but the principle theme of SMT is that performance is greatly increased by *sharing* most processor resources. SMT has been shown to be implementable by modifying existing superscalar processors with fairly minimal changes.

Chip multiprocessing (CMP) is the concept of placing multiple processor cores on a single chip. The primary motivation is that instead of designing processors as large monolithic units, they can be broken down into simpler albeit less powerful units [10]. By dividing up our task into smaller units the design process is greatly simplified and through the combination of these simplified units we may actually increase the overall performance.

Tullsen and Seng showed that simultaneous multithreading is capable of saving energy by reducing energy lost through mis-speculation [21]. Recent findings have shown CMP to be perhaps just as beneficial for energy savings when compared to SMT, and that in some cases chip multiprocessors are likely to be more energy-efficient than SMT processors [20]. Although the act of combining multiple units onto a single chip does not inherently save energy, energy savings are due to hardware simplification characteristic of the CMP theme.

Much work has been done on directly comparing the two methodologies. Tullsen, et al. showed that SMT outperforms a CMP of the same number of contexts [26], while Hammond et al. showed that the design simplification of CMP could in some cases allow it to outperform an SMT [10]. For power consumption, Kaxiras et al. demonstrated that on mobile phone workloads an SMT processor could outperform a CMP in terms of energy efficiency [13], and Li et al. recently performed an in depth study of the reasons for the energy efficiency of SMT [16].

To summarize the temperature differences across a chip, we can compare the nominal temperature and the hottest local temperature. The overall spatial average temperature represents how hot our chip is on the whole, and is likely to be consistent with values that we could calculate from power-aware analysis without involving spatial layout considerations. This “power envelope” technique has been used extensively because it serves as an accurate approximation as long as large spatial temperature gradients are not present [9]. However, from a temperature-aware design perspective these gradients are important to take into account.

3. Experimental procedure

3.1. Tools and benchmarks

We use the HotSpot temperature modeling tool developed at the University of Virginia [24]. HotSpot can be plugged into a power-performance simulator such as SimpleScalar-Wattch [1]. HotSpot allows the user to specify a processor floorplan that gives the layout of a processor and its functional units. From this floorplan, it creates an equivalent circuit model that models heat transfer in a silicon processor with specified thermal packaging. We use Wattch as our basis to provide the power numbers to HotSpot.

We have selectively chosen eight benchmarks from the SPEC 2000 suite. In order to rapidly simulate our benchmarks, we used empirically derived simulation points as described by Sherwood et al. to generate simulation statistics representative of each benchmark’s complete program behavior [18, 22].

We modified SimpleScalar to support simultaneous multithreaded and chip multiprocessors. For our SMT design, we allow complete sharing of the instruction issue bandwidth and functional units, and shared branch prediction. For simplicity, our current fetch policy is Icount, as described in [26]. Our CMP specifications require duplication of most resources for each effective processor. Communication between cores is done through the shared L2 cache, consistent with modern CMP designs such as Hydra and the IBM Power4 [10, 19]. Both our SMT and CMP simulator are designed for running distinct programs in parallel, although in the future we expect to add support for cooperating threads.

3.2. Experiment parameters

Our processor parameters are designed to use resources appropriate for a modern high performance multithreaded processor built on 0.13 micron technology such as had been planned for the Alpha 21464 (EV8) [7]. The EV8 sought to support 4-context multithreading, the degree to which we experiment with in this paper. Using 0.13 micron technology, we have assumed features such as a clock rate of 3 GHz, a relatively large L2 cache, and other parameters scaled up accordingly.

Our parameters for a single-context superscalar processor, SMT processor, and two CMP configurations are shown in Table 1. Our models assume the same parameters for SMT as the single-context superscalar and CMP resources are mostly divided proportionally.

Table 1. Processor parameters.

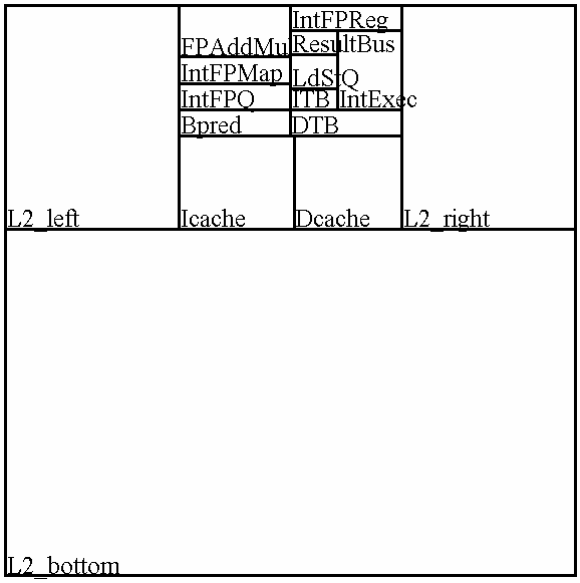
	Single-context Super-scalar	SMT	2-core CMP (per-core params)	4-core CMP (per-core params)
Clock speed	3 GHz	3 GHz	3 GHz	3 GHz
Ifetch queue size	16	16	8	4
Bpred table size	4096	4096	2048	1024
Issue width	8	8	4	2
Commit width	8	8	4	2
RUU window size	128	128	64	32
Ld/Store Queue size	64	64	32	16
L1 Dcache	128 KB	128 KB	64 KB	32 KB
L1 Icache	128 KB	128 KB	64 KB	32 KB
L2 cache	4 MB	4 MB	4 MB	4 MB

Hammond et al. pointed out that a simpler core would allow a CMP to be clocked faster [10]. However, support for this argument has not held up according to recent findings [2, 6]. Arguably the more complex logic of a superscalar or SMT processor can make issue delays longer, but sensible long-term design would result in additional pipeline stages and not necessarily a slower clock. As such, for comparisons of this nature we use the same clock rate of 3 GHz for our SMT and CMP simulations even for the smallest cores.

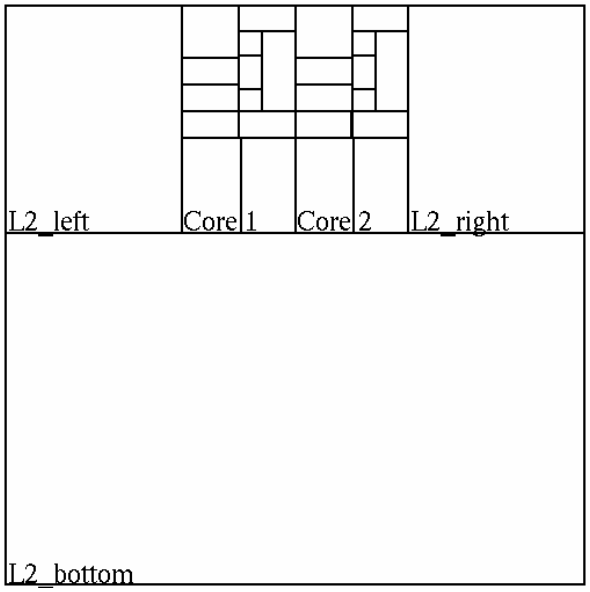
A number of different die area tradeoff trends have been proposed, but it remains challenging to define fair tradeoffs between the two paradigms. Burns and Gaudiot have developed sophisticated models that estimate die area consequences for both SMT and CMP [2, 3]. In this paper we have taken a simpler approach. For example, some resources such as the data cache are divided evenly when separated into two separate CMP cores. While we could attempt to appropriate processor resources using some more complex empirical formulas as done by [2], our overall purpose is not to do a strict head-to-head comparison. Instead, we are aiming to find the long-term temperature-aware trends seen for SMT and CMP and to identify solutions to such arising thermal problems.

3.3. Floorplans

HotSpot requires sketched floorplans to specify the relative physical locations of processor functional units. Our base floorplan and a floorplan derived from it are shown below in Figure 1 (a). Chip multiprocessors are formed using a script to duplicate, scale, and place new cores based on a starting floorplan while maintaining the same size L2 cache.



(a) The basic floorplan we use for an SMT or single-context superscalar processor.



(b) Floorplan of our basic 2-context CMP.

Figure 1. Examples of two HotSpot floorplans used in our experiments.

CMP has the benefit that almost all resources are reduced in size approximately proportionally to the degree of splitting. One such resource that is not sized down in this way is the set of architectural registers. However, since the register file did not become an object of excessive heat output even when compressed under our models, we used the simple approximation of resizing the register file along with the entire core.

4. Results and discussion

To gather an understanding of temperature characteristics, we begin by analyzing the related performance and energy consumption.

4.1. Performance and energy efficiency

Figure 2 shows our achieved IPC on three architectures: single-context superscalar, 2-context SMT, and 2-core CMP. The superscalar column is omitted for multithreaded workloads because it cannot run them without course-grained context-switching. As expected, both SMT and CMP achieve significantly higher IPC when running multiple benchmarks. CMP also carries its expected performance loss for single-threaded benchmarks as compared to a single-context superscalar or SMT processor.

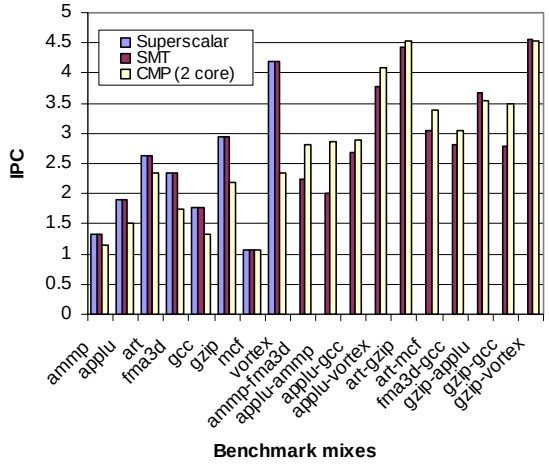


Figure 2. Performance on superscalar, 2-context SMT processor, and 2-core CMP.

The IPC of mixed workloads depends largely on the IPC of the individual programs making up each multithreaded workload. For example, `gzip` and `vortex` can both sustain high ILP when running alone, and achieve even higher IPC when multithreaded together. On the other hand, the workload of `art` and `mcf` shows low IPC, brought about because `mcf` is a very memory intensive benchmark. In the next section we will see a significant correlation between high IPC and high processor temperature, so it is relevant to know how IPC is affected by multithreading to see the temperature effects from mixing various programs.

For a more complete picture, consider also the energy per instruction (EPI) metric. Energy data for all three architectures is shown below in Figure 3. As with IPC, we see improvement in EPI on both multithreaded architectures.

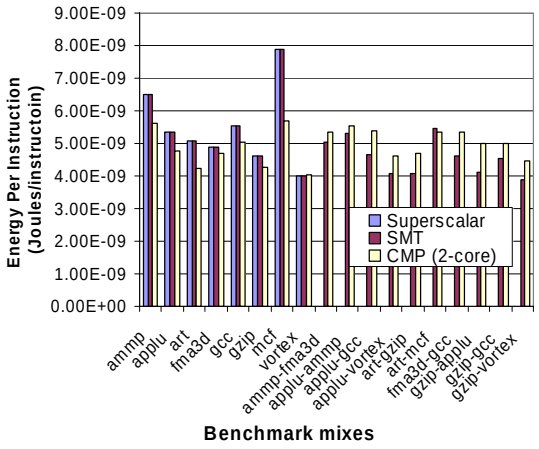


Figure 3. EPI for the three architectures. Lower EPI values indicate better energy efficiency.

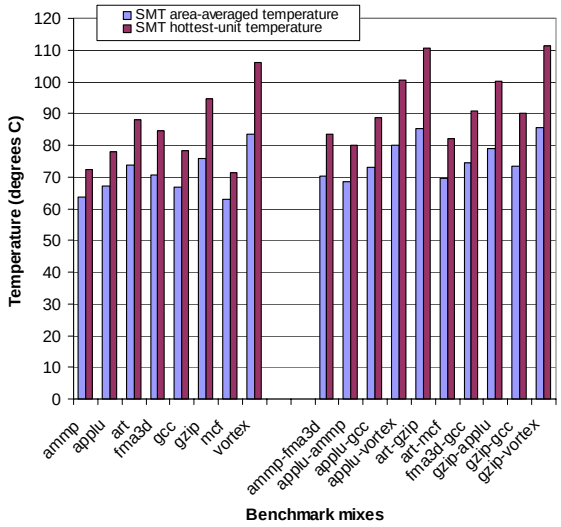
Neither the IPC nor EPI metric alone sufficiently represents overall execution quality, so we present both initial metrics so that we can examine the relation between performance, energy, and then temperature. As seen in Figure 3, both multithreading techniques improve energy efficiency as measured by EPI. SMT can accomplish this by reducing wasted energy due to lower penalties from speculated instructions such as mispredicted branches. Energy saved by CMP is essentially due to the hardware simplification of each smaller core. Both energy savings can be summarized as resulting from more efficient usage of hardware.

Given the less profound difference in EPI between different multithreaded workloads, IPC stands as a better candidate to correlate with thermal effects, and we shall shortly see the strong connection between high IPC and high processor temperatures.

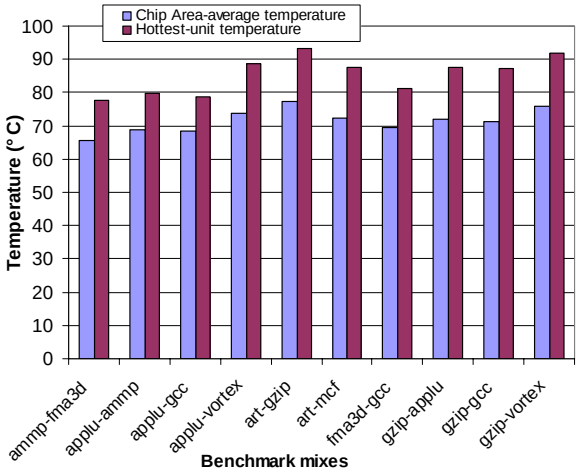
4.2. Temperature comparison

HotSpot allows us to find the predicted final temperatures for an execution in the case where we suppose that the power values from that sample of execution would remain consistent in the limit as time

approaches infinity. These *steady-state* temperatures are the primary measurements used in our experiments. Our steady-state temperatures for all benchmarks initially revealed large temperature gradients including prominent hot spot; in our case result bus and instruction window are typically the hottest units. The coldest component is always the L2 cache, as one would expect given its relative large size and infrequent usage. The difference between this apparent chip temperature and the temperature of the processor's hottest units ranges from 5° to 25°, and the area-average chip temperatures for various benchmarks differ widely as well.



(a) Initial temperatures from our SMT configuration. Single-threaded workloads represent also represent a superscalar.



(b) Initial temperature data from our CMP 2-core configuration.

Figure 4. Nominal temperatures and hottest unit temperatures running various workloads.

For example, multithreading `gzip` with `vortex` on an SMT processor results in an overall quite warm processor, registering 85.5° C as the apparent (area-averaged) chip temperature. On the other hand, when a low IPC benchmark such as `ammp` runs alone on a single-context superscalar processor, we see a much lower apparent temperature (63.7°) and the hottest unit on that die registers at 72.2°. Yet despite the difference in magnitude, we find the hottest units among each workload are usually the same: in our case, the instruction window and result bus. Our steady-state temperatures including the hottest unit temperatures and chip-area average temperatures for both our SMT architecture and CMP architecture are shown in Figure 4 (a). Likewise Figure 4 (b) contains corresponding temperature data for our 2-core CMP architecture.

The existence of similar spatial temperature profile trends extends to even 2-core CMP chips running different benchmarks. For example, when running `gzip` and `applu` on two separate cores of a CMP, we find that the core running `applu` is generally 6° cooler than the core running `gzip` with respect to each core's corresponding functional block units, and yet the instruction window and result bus are still found to be the hottest units on both cores.

Modern processors are designed for adequate *average* power consumption and then rely on various forms of dynamic thermal management (DTM) to deal with peak thermal constraints [4]. Although maximum chip temperatures for typical desktop microprocessors such as the Intel Celeron are marked at around 85-90° C, our architectures often reach beyond this range for SPEC benchmarks. As processors become more thermal-limited in the future, the common operating regime will likely come closer to the more thermal limited cases such as the ones analyzed in our experiments [11].

4.3. Temperature-aware design enhancements

Since the difference between the chip's apparent temperature and its hottest unit's temperature is a problem, we wish to seek techniques that reduce this gap even if they do not necessarily lower the temperature on the cooler locations of the processor. Skadron et al. proposed migrating computation (MC), whereby a single hot unit is duplicated and a backup structure located further away could be used when the original unit becomes too hot [24]. The motivation behind this was that intuitively it is most desirable to move hot units far apart from each other, but it is preferred to do this in some way that doesn't

significantly damage overall performance because of communication delay.

For our first thermal improvement test, we notice that CMP's property of separate cores enables moving entire cores around the processor layout. We target CMP cores because the flexibility inherent in separate cores makes this option most likely to have small performance consequences. Our modified layout reflecting relocated cores is shown below in Figure 5, which should be compared with Figure 1 (b).

Using CMP core relocation, we ran new simulations with the modified layout and compared our final steady-state hottest-unit temperatures with the original values. We find that insignificant temperature drops are achieved as shown in Figure 5.

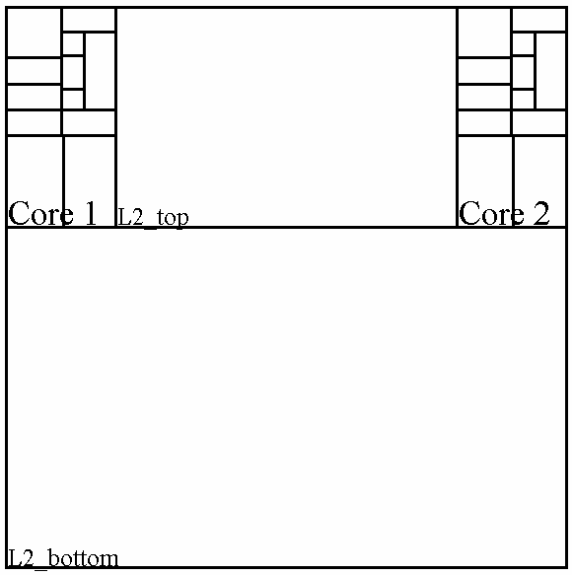


Figure 5. Modified 2-core CMP layout with cores relocated to corners of the chip.

Not shown here is the area-average chip temperature. Our new layout structure attempts to redistribute the heat, not reduce the average quantity of it on the die, and so there is no noticeable drop in the nominal temperature. What is surprising, however, is that the hottest unit temperatures as shown in Figure 6 showed a practically insignificant 1° reduction at most.

This disappointing result suggests that lateral heat transfer perhaps does not play a significant role in creating thermal hotspots. Since heat dissipation appears to be much stronger through the vertical packaging technology than through adjacent silicon, enlarging the area of computation rather than relocation may be the important factor.

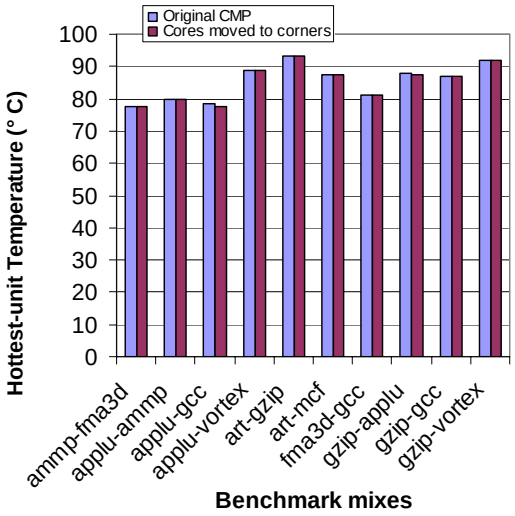


Figure 6. Changes in hot unit temperature due to moving CMP cores to corners of the layout.

Although somewhat larger improvements can be obtained by selectively relocating the hottest units themselves—which is likely to be more costly from a design standpoint—these gains are still small compared to what we find from our next technique. Heo et al. have pointed out that as die thickness reduces with respect to lateral chip dimensions that lateral heat transfer may become even less significant for future die technologies [11]. Since most of the heat is dissipated vertically through the heat sink, increasing the area of a unit could be much more relevant than actually separating the unit into distally located components.

MC actually causes both effects at the same time, but in our experiment we allow a unit only to consume extra area to see if temperature reduction might owe simply to increased thermal contact area. Our new layout involves increasing the size of the hot area, as shown in Figure 7, and the results of the layout changes are shown in Figure 8. We see a noticeable drop in temperature owing to enlarging the hot units. Moreover, despite the smaller sized cores of a 2-core CMP we see approximately the same benefit.

Relocating functional units or processor cores can result in unnecessary communication overheads. If little temperature benefit is achieved this way, unnecessary separation may be completely unnecessary to combine with simple unit enlargement.

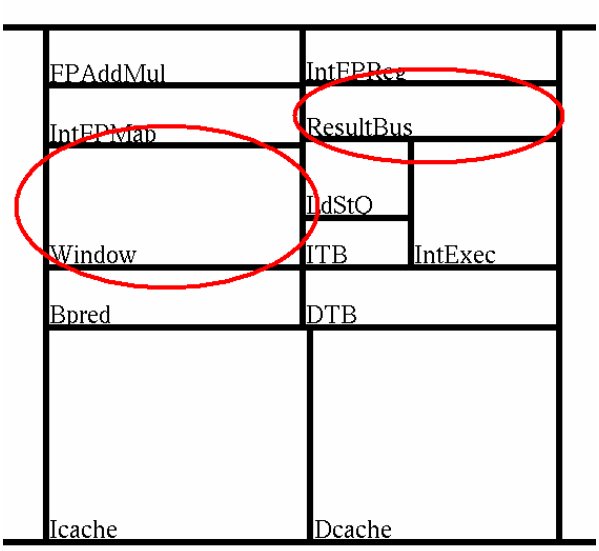


Figure 7. New core layout with increased spread of the result bus and instruction window.

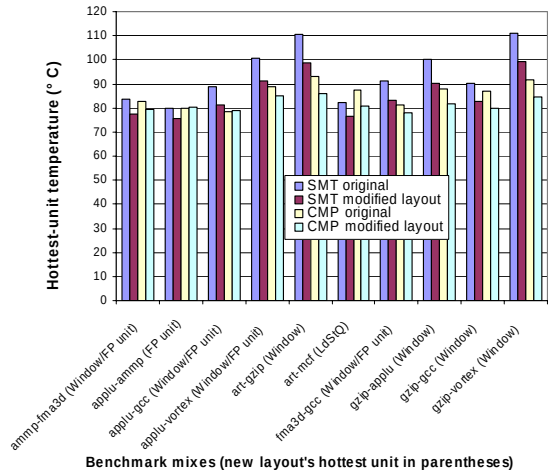


Figure 8. Temperatures resulting from allowing the heating structure to take up more local area.

Although we have now seen that increasing the die area of a particular unit can have temperature benefits, a question that arises now is what does it exactly mean to simply enlarge a processor unit. There are a number of ways by which extra space can be added to such units, and these methods along with their various costs are discussed in Section 4.5.

4.4. Four-context/four-core processors

To see how these trends continue as the number of threads is increased, we ran two four-benchmark mixes. We have used two thread mixes, the first described as all memory-intensive programs while the second consists of four high-ILP programs as classified by [25]. For the 4-context SMT we used the same layout as with single-context and 2-context executions, while the 4-core CMP was generated the same way we expanded our initial layout to the 2-core chip from Figure 1 (b). For the 4-thread cases, again relocating CMP cores to the die corner proved ineffective, so here we shall present only our results from hot unit enlargement. The modifications shown in Figure 7 were applied and duplicated to each core in the CMP 4-core layout. The original hottest-unit temperatures and improved temperatures are shown in Figure 9.

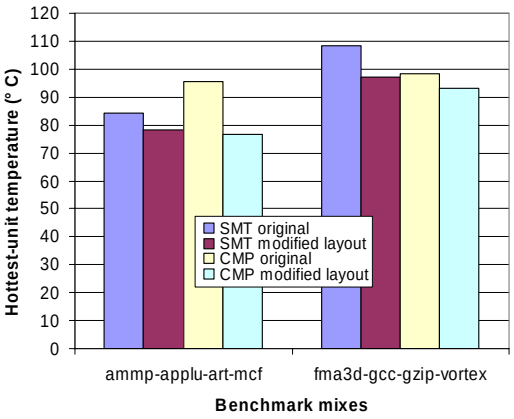


Figure 9. Original and final temperatures for four-context SMT and CMP workloads.

The improvements in the SMT configuration are still significant (6° and 11° respectively). When explaining the large improvement, one must also take into account that the original temperature gap was large as well. In the CMP configuration we see a similar benefit. Repetitive architectures have the benefit of reduced complexity of design, and recently these arguments have been extended to power savings in that power-efficient design on a repeated structure can create multiplicative power saving in reduced design time [14], and a similar argument can be made for temperature effects.

4.5. Detail level for area-enlarging techniques

In Sections 4.4 and 4.5 we found that enlarging hot units could significantly reduce temperature gradients, although it may not yet be clear what sort of a design technique could be used to accomplish this. Deeney has suggested that thermal hot spots can be mitigated by placing blank silicon nearby as a very coarse-grain method [5]. Unfortunately as we have observed and is

noted by [11], the lateral heat transfer may be so small that we cannot expect significant transfer between the active unit and the adjacent empty silicon.

If we place blank spots throughout the unit on a more fine grain level, however, we can eventually approach uniform heat distribution. To find what level of granularity is appropriate, we used our floorplan modeling to break the hottest units into granular blocks and measured how fine grain the distribution must be in order to adequately distribute heat. Since our end results for this granularity experiment are quite negative in terms of thermal manageability, we use as our single workload the SMT combination of `gzip-vortex` because it gives a large (111.2° down to 99.4°) temperature gap. In Figure 10 we show how heat spreading could be distributed within the layout of the hottest units, and the resulting thermal hot spot temperatures from such models are shown in Figure 11. The single-block case represents no blank silicon patches hence ideal uniform heat distribution throughout each unit. Breaking the hot units into two blocks (one on and one off) is a non-ideal situation that gives very little benefit over the original non-spread configuration that registered 111.2°. Increasing the granularity of spreading to 441 pieces for both the instruction window and result bus does improve heat spreading, but even at this level there is still a 5° difference between the average of the "on" and "off" spots. Exponentially increasing the number of pieces in our model is not feasible because the execution time of HotSpot is tied to the number of layout block units.

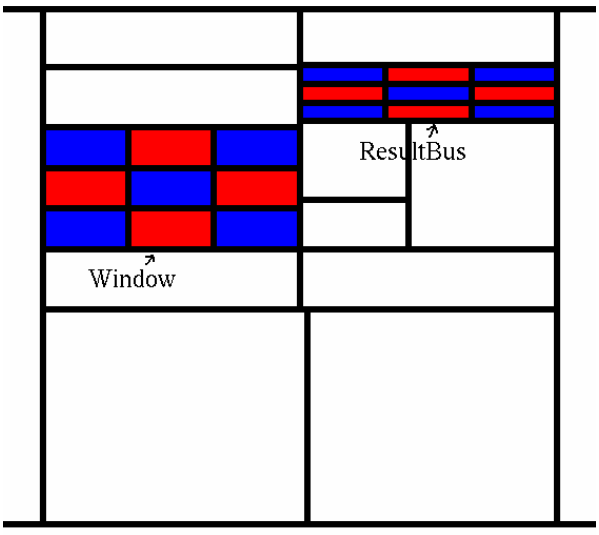


Figure 10. Sample (n = 9) core layout to find the granularity necessary for sufficient temperature distribution.

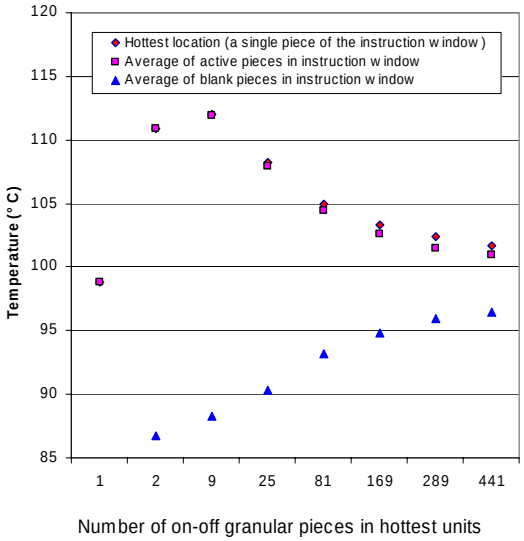


Figure 11. Temperatures of hottest unit (instruction window) for `gzip-vortex` under SMT with granularly enlarging hot units. At an infinite number of pieces, the averages of the active pieces and blank pieces should both converge to the single-piece temperature.

Disappointingly, here we have found that spreading a block must be done very finely in order to reduce its likelihood of it being a set of hot spots.

Fortunately, we need not restrict ourselves to designs that consist only of a component's original structure alongside blank silicon. MC is performed instead by creating only two duplicate units and alternating calculation between the two. Somehow the units must be able to copy over any stored data when switching between the two duplicate units. Since this copying operation may be expensive, it is important to limit how often this occurs. Heo et al. showed that this *activity migration* could be performed on the level of microseconds and achieve its desired effect [11]. Under the various methods presented, expanding of units can incur costs in terms of die area and timing difficulties, and these tradeoffs must be taken into consideration.

The granularity experiment presented in this section also gives us an idea of how precise our floorplan modeling scheme is. We have used strong assumptions that various blocks are monolithic units with a single temperature. Here we see evidence that we may need to model at a much more precise level for accurate temperature effects because assumptions of simple "spreading" may be too strong for large units with a heterogeneous structure.

4.6. Performance and thermal tradeoffs

When compared to SMT, the primary weakness of CMP is poor single-threaded performance. From the temperature-level design perspective, we see that this problem worsens. Because most heat dissipation is largely through the vertical heat sink directly above the core and not through an indirect path through other core, the heat generation of a single core is largely its own localized problem even when the other cores are idle. On the other hand, CMP research is largely geared toward compiler enhancements to enable thread level speculation and parallelize single threaded programs to run on multi-core processors [17]. With these enhancements, the isolation problem for single-threaded programs can be less significant. Furthermore, chip multiprocessors are largely geared toward server or network environments with many clients and so peak performance may not be a big issue. In the single-threaded case SMT did well in performance, but high-IPC executions often came alongside high temperatures almost as often as with multithreaded workloads.

For the corresponding temperature effects we saw in the beginning that high temperatures correlated strongly with high IPC workloads. This indicates that any ILP gains that we have achieved through multithreading may quickly be mitigated if we operate within the temperature limited regime. However, if we have methods to deal with these temperature problems in our architectures, there is certainly promise in ways to continue achieving improved performance. Our layout restructuring techniques tend to give better temperature improvements for the SMT architecture, although we also had worse temperature gradients for the SMT architecture to begin with. As such, it seems both SMT and CMP are viable multithreaded architectures amenable to thermal management techniques.

5. Future work

The models used in this paper to specify parameters for SMT and CMP were quite simple. Use of more detailed transistor-count estimation tools could automate comparative simulations [8, 23]. Their benefit is twofold: allowing us to estimate the microarchitecture parameters as well as providing information to generate more accurate HotSpot floorplans.

HotSpot's model has been validated using Floworks, an industry level computational fluid dynamics tool [24], and using a thermal test chip [12]. However, now that there are existing fabricated SMT and CMP processors, through the use of temperature sensors we have a possible real-chip method by which we can analyze and validate temperature-aware simulation techniques, and we expect to do this in the near future.

Also, all the experiments outlined in this paper were done using the steady-state temperatures, which fail to account for the transient nature of temperature. Thermal management is generally quite a dynamic problem that involves active techniques such as dynamic frequency and voltage scaling (DVFS). And so in the future we intend to do a temporal—not merely steady-state—study of temperature effects and thermal management.

6. Conclusions

Recall that this paper began with three questions:

- Does either of the two processor design paradigms inherently give better thermal management alongside performance and power efficiency consequences?

Our temperature results indicate higher temperatures and larger temperature gradients for SMT as opposed to CMP. However we also see these temperature problems receiving excellent improvement from our layout restructuring experiments particularly in the SMT case. When taking performance into account, we saw an IPC increase due to multithreading from both architectures, although the best IPC improvements came alongside high temperatures. All in all, both multithreaded architectures as a means to improvement ILP and both show promise in that we have ways of dealing with the ensuing thermal problems.

- With multithreading will thermal hotspots become even more of a problem?

While higher IPC results in overall high temperatures, we fortunately found that there was a high degree of predictability. Although the hot spots become hotter, we were not likely to see new hotspots in different locations. For the SMT case, large peak temperature problems are often present but these usually correspond with high IPC and the single-threaded temperature emergencies. Having consistently the same pattern in the sequence of hottest units, we see that there is much predictability that makes thermal problems manageable

In the CMP case, we have a very similar scenario. An added difference is that here we have multiple cores, and usually a single core sets the characteristic peak temperature even if the other cores carry much

lower temperatures. Fortunately the simpler cores inherent in CMP serve as somewhat of a balance to avoid extremes, which is why we see less extreme temperatures in CMP. Furthermore, possibly a single core approaching high temperatures may adversely affect other cores if we limit processor operation with DTM. Large-scale multiprocessors such as Raw and TRIPS use sophisticated interfaces that require synchronized operation. On the other hand, recent studies have shown the benefits of heterogeneous cores within a single architecture in order to gain flexibility [15].

- And do thermal management techniques such as migration of computation retain their utility as we continue to scale up the number of threads or processor cores?

Encouragingly, our tested solution for thermal management is increasing the size of only hot units, and we found that this general technique retains its utility for as many as four simultaneously multithreaded contexts or four processor cores.

7. Acknowledgements

Zhigang Hu and Philo Juang developed the SMT and CMP enhancements for SimpleScalar. We would like to thank the anonymous reviewers for their helpful comments. The authors' power-aware micro-architecture research is supported in part by grants from NSF, Intel, and SRC.

8. References

[1] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proc. ISCA-27*. pp. 83-94. May 2000.

[2] J. Burns and J. Gaudiot. Area and System Clock Effects on SMT/CMP Processors. In *PACT'01*. pp. 211-218. Sept 2001.

[3] J. Burns and J. Gaudiot. SMT Layout Overhead and Scalability. In *IEEE Transactions on Parallel and Distributed Systems, Volume 13, Issue 2*. pp. 142-155. Feb 2002.

[4] A. Cohen, L. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy. On Estimating Optimal Performance of CPU Dynamic Thermal Management. *Computer Architecture Letters, Volume 2*. Oct 2003.

[5] J. Deeney. Thermal Modeling and Measurement of Large High-Power Silicon Devices With Asymmetric Power Distribution. *35th International Symposium on Microelectronics*. Nov 2002.

[6] M. Ekman and P. Stenstrom. Performance and Power Impact of Issue-width in Chip-Multiprocessor Cores. In *ICPP'03*. pp. 359-368. Oct 2003.

[7] J. Emer. Simultaneous multithreading: Multiplying Alpha Pperformance. Microprocessor Forum. Oct 1999.

[8] M. Farrens, G. Tyson, and A. R. Pleszkun. A Study of Single-Chip Processor/Cache Organizations for Large Numbers of Transistors. In *UC Davis Computer Science Department Technical Report CSE-92*-24. Dec 1992.

[9] S. Gochman, R. Ronen, I. Anati, A. Berkovits, T. Kurts, A. Naveh, A. Saeed, Z. Sperber, and R. C. Valentine. The Intel Pentium M Processor: Microarchitecture and Performance. *Intel Technology Journal, Volume 07, Issue 2*. May 2003.

[10] L. Hammond, B. A. Nayfeh, and K. Olukotun. A Single-Chip Multiprocessor. In *IEEE Computer*, pp. 79-85. Sept 1997.

[11] S. Heo, K. Barr, and K. Asanovic. Reducing Power Density Through Activity Migration. In *Proc. ISLPED'03*.

[12] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy. Compact Thermal Modeling for Temperature-Aware Design. In *Proc. DAC-41*. June 2004.

[13] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads. In *CASES '01*. pp. 211-220. Nov 2001.

[14] J. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff. Energy Characterization of a Tiled Architecture Processor With On-Chip Networks. In *Proc. ISLPED'03*. pp. 424-427. Aug 2003.

[15] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. Tullsen. Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction. In *Proc. MICRO-36*. Nov 2003.

[16] Y. Li, K. Skadron, D. Brooks, and Z. Hu. Understanding the Energy Efficiency of Simultaneous Multithreading. In submission. April 2004.

[17] K. Olukotun, L. Hammond, and M. Willey. Improving the Performance of Speculatively Parallel Applications on the Hydra CMP. In *Proc. ICS'99*, June 1999.

[18] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood, and B. Calder. Using SimPoint for Accurate and Efficient Simulation. In *ACM SIGMETRICS*. June 2003.

[19] Power4 System Microarchitecture: White Paper. IBM website. http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.html. May 2004.

[20] R. Sasanka, S. V. Adve, Y. Chen, and E. Debes. Comparing the Energy Efficiency of CMP and SMT Architectures for Multimedia Workloads. *UIUC CS Technical Report UIUCDCS-R-2003-2325*. March 2003.

[21] J. S. Seng, D. M. Tullsen, G. and Z. N. Cai. Power-Sensitive Multithreaded Architecture. In *ICCD* '00. June 2000.

[22] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *Proc. ASPLOS-10*. pp. 45-57. Oct 2002.

[23] U. Sigmund, M. Steinhaus, and T. Ungerer. On Performance, Transistor Count and Chip Space Assessment of of Multimedia-enhanced Simultaneous Multithreaded Processors. *4th Workshop on*

Multithreaded Execution, Architecture, and Compilation. Dec 2000.

[24] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Microarchitecture. In *Proc. ISCA-30*, pp. 2-13. June 2003.

[25] D. Tullsen and J. Brown. Handling Long-latency Loads in a Simultaneous Multithreading Processor. In *Proc. MICRO-34*. Dec 2001.

[26] D. Tullsen, S. Eggers, and H. Levy. Simultaneous Multithreading: Maximizing On-Chip Parallelism. In *Proc. ISCA-22*, 392-403. June 1995.