Low-Overhead Core Swapping for Thermal Management

Eren Kursun[1], Glenn Reinman[1], Suleyman Sair[2], Anahita Shayesteh[1], and Tim Sherwood[3]

[1] Computer Science Department, University of California, Los Angeles
[2] Department of Electrical and Computer Engineering, North Carolina State University
[3] Department of Computer Science, University of California, Santa Barbara

Abstract. Technology scaling trends and the limitations of packaging and cooling have intensified the need for thermally efficient architectures and architecture-level temperature management techniques. To combat these trends, we evaluate the thermal efficiency of the microcore architecture, a deeply decoupled processor core with larger structures factored out as helper engines. We further investigate activity migration (core swapping) as a means of controlling the thermal profile of the chip in this study. Specifically, the microcore architecture presents an ideal platform for core swapping thanks to helper engines that maintain the state of each process in a shared fabric surrounding the cores. This results in significantly reduced migration overhead, enabling seamless swapping of cores. Our results show that our thermal mechanisms outperform traditional Dynamic Thermal Management (DTM) techniques by reducing the performance hit caused by slowing/swapping of cores. Our experimental results show that the microcore architecture has 86% fewer thermally critical cycles compared to a conventional monolithic core.

1 Introduction and Motivation

Thermal characteristics of contemporary processors are creating significant challenges to microprocessor design. Various trends threaten to make things even worse: the number of on-chip transistors is quickly approaching one billion, clock frequencies are dramatically increasing, feature sizes are dropping to deep submicron levels, and supply voltage reduction is expected to slow down as it approaches noise margin barriers. As a result, power densities and on-chip temperatures are expected to increase even faster for the next generation of processors.

Thermal issues have gained significant importance in the past few years. Processor heating raises number of problems that threaten vital aspects of the microprocessor design, such as proper functionality, reliability, cost, and performance. Reliability of an electronic circuit is exponentially proportional to the junction temperature. A 10°C increase in temperature usually translates to ∼2X difference in the lifespan of the device [16]. At higher operating temperatures the microprocessor operates at relatively lower speeds [23].

B. Falsafi and T.N. Vijaykumar (Eds.): PACS 2004, LNCS 3471, pp. 46–60, 2005.

Furthermore, temperatures are not constant across the chip. $30-40°C$ thermal gradients are quite common, which causes potential timing and data errors [2]. There is a non-linear relationship between cooling capabilities and the cost of a cooling solution. The cost of cooling increases at a higher (almost exponential) rate for higher temperatures [10].

In recent years, dynamic thermal management (DTM) [4, 8, 14, 25, 15] has become an integral part of microprocessor design to adapt to increasing on-chip temperatures. The disparity between the maximum possible power dissipation and typical power dissipation has become more pronounced. This, along with the exponential increase in cooling device costs, has created a new trend where cooling systems are designed for the typical worst case power dissipation instead of the maximum possible power dissipation. Therefore, dynamic thermal management has become essential to ensure that processor temperature does not reach or exceed the maximum tolerable temperature.

Many power optimization techniques do not seem to address problems caused by processor heating, as they are targeting relatively cooler parts of the chip, such as caches. With the expected increases in power consumption and temperature, there is no doubt that more DTM techniques specific to microprocessor designs are needed.

DTM usually targets the removal of excessive heat from the processor after a certain temperature threshold is reached. Thermal management can cause performance degradation, as a result of reduced clock frequency, voltage or temporarily shutting down the entire chip. Therefore, thermal efficient architectures with less overall heating are extremely desirable, as they do not require very aggressive DTM.

In this paper we explore the thermal efficiency of the microcore architecture [18]. The microcore architecture features a small, fast pipeline augmented with helper engines [22]. All large structures are factored out of the microcore and are relocated as helper engines, taking advantage of locality in the first level structures. In this paper, we explore the use of swapping applications between multiple microcores when a given core exceeds a thermal threshold. The helper engines buffer state during core swaps and help reduce the overhead of swapping. We compare this approach to current DTM techniques.

The rest of this paper is organized as follows. In Section 2 we discuss the prior work, followed by an introduction of the architectures we investigate in Section 3. Section 4 presents the methodology. We present the experimental results in Section 5 and concluding remarks are in Section 6.

2 Related Work

The circuit design community has proposed a great deal of work on dynamic power optimization techniques, which are also used as dynamic thermal management techniques in microprocessors in various forms. Such techniques include dynamic voltage scaling (DVS) and dynamic frequency scaling (DFS). In this section we will focus on the studies that are close to our own and specifically target microprocessor power/thermal optimization.

The Pentium 4 [12] incorporates a low cost, yet reliable, thermal management system based on processor power modulation that has been commonly used in mobile systems. It utilizes the existing *stoplock*, an architectural low-power logic mechanism that halts the clock signal to the bulk of the processor [10]. Thermal management is automatically invoked whenever any of the thermal sensors indicates that the die is hotter than a predetermined critical temperature. The mechanism stays active until the die temperature drops below the critical value. The clock signal is gated at certain intervals or permanently, depending on the thermal and power management state.

Brooks and Martonosi introduced an adaptive thermal management system through speculation control in [4]. They also compared commonly used DTM techniques such as clock frequency scaling, voltage and frequency scaling, decode throttling, speculation control and instruction cache toggling [8]. An energy-management framework that combines energy efficiency and temperature management, DEETM, was presented by Huang et al. [25]. They propose several power optimization techniques such as global clock gating, DVS, sub-banking, filtered instruction cache. Although these studies provide valuable DTM techniques with significant thermal alleviation, detailed resistance-capacitance thermal models were not available at the time. As a result some of the overheating blocks were not addressed.

Him, Daash and Cai introduced a dual pipeline processor, with a secondary low-power pipeline in [15]. The power efficient single-issue, in-order pipeline only gets activated, when the primary pipeline exceed a threshold temperature. When the superscalar core overheats, it is flushed and the secondary pipeline is activated until the primary pipe cools down to a safe temperature. Register file, fetch engine and the execution units are shared among the two pipelines. However, it is important to note that this technique is mainly targeting mobile devices and applications that can tolerate low performance. There is a significant performance penalty when the architecture transitions to the secondary pipeline.

In [11] Heo, Barr and Asanovic proposed an activity migration technique for power density reduction. Activity migration reduces the temperature by moving the computation between multiple replicated blocks. This thermal reduction yields lowered leakage power values and can also be improved with a dynamic voltage scaling technique to further reduce the power and temperature.

Heo et al. [11] analyze multiple configurations with some of the microprocessor units replicated or shared. The study concludes that the best configuration has a shared Icache, Cache, rename table, and issue queue. Although, duplicated microprocessor units reduce the on-chip temperatures, they argue that this is dominated by the overhead due to activity migration.

HotSpot [14] provides an accurate thermal model and a corresponding software implementation that enables more detailed and localized thermal analysis of the microprocessor. It is based on the equivalent circuit of thermal resistance and capacitances that model the microarchitectural blocks and other aspects of the chip thermal package. Hotspot highlights the inaccuracy in estimating the temperature based on the power density only. The software models can be

integrated with the other cycle accurate power estimators such as WATTCH [5] and Hotleakage [27], in order to provide a complete thermal and power analysis. In [14] Skadron et al. also provide and analyze several DTM techniques such as: temperature-tracking frequency scaling, localized toggling and computation migration. We incorporate HotSpot models for an accurate RC thermal analysis of the various architectures investigated in this study. We also make use of an idealized version of dynamic frequency scaling as a comparison point for the our core swapping approach.

3 Factored Architectures

Figure 1 illustrates a factored architecture as proposed in [18]. The main idea behind factored architectures is to move a set of larger structures out of the regular processor core, resulting in a tiny core with only the necessary components included.

While structures such as caches are fairly easy to factor, other structures require more consideration. In [18], Shayesteh et al. looked at three different types of factored structures, and their challenges:

- Hierarchical extensions: Caches and branch predictor (shown in light gray)
- Complete factorization: Value predictor and data prefetcher (shown in dark gray)
- Hybrid factorization: Register file and ROB (shown with gray stripes)

In a typical factored design, the level one data and instruction caches are moved out of the core processor pipeline and replaced with a smaller L0 cache. The L0 extends the cache hierarchy, and therefore the L1 data cache is accessed on an L0 miss.

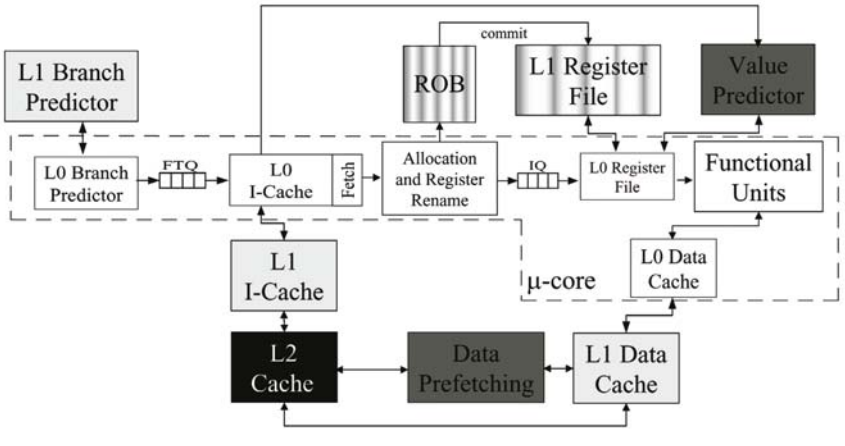

Fig. 1. The factored μ-core architecture

The architecture includes a stream buffer architecture [13] guided by a stride-filtered markov predictor as proposed in [20]. The address predictors are moved further away from the core pipeline in the microcore. There is also a hybrid value predictor [24], predicting only load instructions. To factor the value predictor, the predicted value is stored in the register allocated to the load instruction we are predicting. If the predicted value and the actual value do not match, a checker engine similar to the ARB [9] detects the misprediction and squashes the mispredicted result and its dependents.

The factored architecture makes use of a basic block target buffer (BBTB) [26], a branch address predictor that predicts an entire basic block each cycle. The microcore design has a reduced size BBTB in the core pipeline and adds a second level BBTB as done in [17]. Similarly the fetch target queue (FTQ) decouples branch prediction from the instruction cache. On a first level BBTB miss, the second level BBTB is probed and fetch stalls until a response is received from the second level. If the second level misses, we guess a fixed fetch block size and continue fetching until a misprediction is detected.

In the factored architecture, a multi-level register file is used similar to the one proposed in [3]. The basic differences are that they model an inclusive register file hierarchy where the second level register file (RF1) includes all the state contained in the first level register file (RF0). On a branch misprediction, the second level register file recovers the state of the first level register file. This is a hybrid of complete factorization and hierarchical extension, as the register file is extended with a second level structure, but the commit hardware and ROB are completely factored, with only tag allocation in the ROB impacting the core timing.

The results in [18] showed that the microcore architecture is able to reduce total processor power dissipation by 20% on average, while it attains comparable performance to a deeply pipelined monolithic design at the same clock frequency. The inherent power efficiency of the microcore, makes it an attractive design for temperature aware architectures. Figure 2 illustrates how different components contribute to the overall power for monolithic and microcore architectures. Our methodology and processor parameters are described in following sections.We

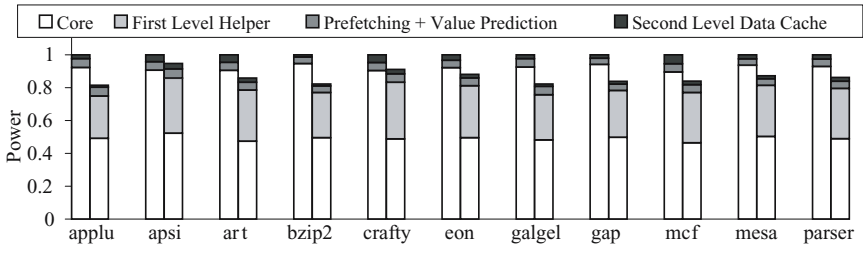

Fig. 2. Power breakdown for Monolithic and μ-core architectures. (Normalized by monolithic architecture power for each benchmark).

use the microcore framework of Shayesteh et al. to make our contribution in the analysis of the temperature efficiency and the examination of core swapping on the microcore.

3.1 Core Swapping

Swapping between multiple cores has been proposed as a dynamic thermal management technique. Heo et al. [11] look at several architectural alternatives for implementing activity migration and its overhead on processor performance. We propose a dual pipeline version of the microcore architecture, with factored components shared between the cores. Unlike [11], our core swaps are triggered by thermal sensors. When one core exceeds a thermal threshold, the application workload is swapped to the other core.

Core swapping can impact processor performance significantly. On a core swap, we flush the pipeline similar to a branch misprediction. Register file state is copied to the other core, and dirty cache blocks are written back to the level one cache (the helper engine), which is shared between the cores. We assume that copying register file state and writing back dirty blocks can be overlapped with the startup cost of the new core.

The cold start effect of caches and predictors causes an even more severe impact on the second core. These structures need to warm up and depending on their size, there is an overhead involved. In a conventional monolithic architecture, recovering from loss of data on relatively large in-core caches and predictors can degrade performance significantly. The microcore architecture, with less state in the core and more buffering between the cores, provides a very tolerant framework for core swapping. We present this feature in Section 5 by comparing the performance degradation of a monolithic core vs. a microcore in the presence of core swapping.

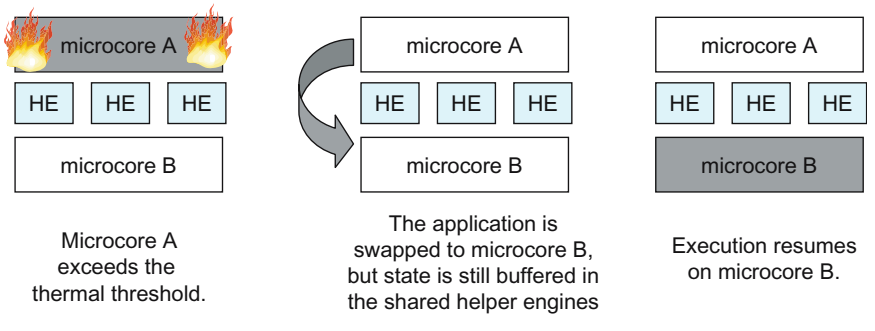

Fig. 3. Core Swapping

4 Methodology

The simulator used in this study was derived from the SimpleScalar/Alpha 3.0 tool set [6], a suite of functional and timing simulation tools for the Alpha

AXP ISA. The timing simulator executes only user-level instructions. Simulation is execution-driven, including execution down any speculative path until the detection of a fault, TLB miss, or branch misprediction. Our processor operates at a 5.6 GHz clock frequency.

We used the SPEC2000 benchmark set for our experiments. Although the results are gathered for all the benchmarks, we only show results for a randomly selected subset of 6 integer and 6 floating point programs in the suite to conserve space in this paper. Details for all benchmarks will be available as a technical report (citation removed for blind review process). The programs were compiled on a DEC Alpha AXP-21164 processor using the DEC C and C++ compilers under OSF/1 V4.0 operating system using full compiler optimization (`-O4 -ifo`). We simulate 100 Million instructions after fast-forwarding application-specif number of instructions as proposed by Sherwood et. al in [19]. All benchmarks were simulated using the *ref* inputs.

4.1 Architectural Model

We have made significant modifications to SimpleScalar to model the various speculative techniques and different configurations in this study. We have modified SimpleScalar to include a cycle accurate, execution driven model of microcore and monolithic architecture models.

Table 1. Simulation parameters for the monolithic and microcore architectures

	Monolithic Core	Microcore	
		L0	Helper Engines
Instruction Window and Physical RF	256 entry ROB 256 entry RF1	128 entry RF0	256 entry ROB 256 entry RF1
BBTB	2048-entry 4-way set associative	256-entry 4-way set associative	2048-entry 4-way set associative
L1 Data Cache	64KB 4-way set associative, dual port with a 32 byte block size, 4 cycle latency	8KB 4-way set associative, dual port with a 32 byte block size, 3 cycle latency	16KB 64-way set associative, single port with a 32 byte block size, 6 cycle latency
L1 Instruction Cache	64KB 2-way set associative, single port with a 32 byte block size, 4 cycle latency	8KB 2-way set associative, single port with a 32 byte block size, 2 cycle latency	64KB 2-way set associative, single port with a 32 byte block size, 5 cycle latency
Value Predictor (1 prediction per cycle)	2K-entry stride 8K-entry markov	none	2K-entry stride 8K-entry L2 markov
Address Predictor (1 prediction per cycle)	2K-entry stride 4K-entry markov	none	2K-entry stride 4K-entry markov
Stream Buffer	32-entry FA buffer	none	32-entry FA buffer
Branch Misprediction	26 cycles	20 cycles	
Core Width	8-way issue, 4-way decode, 4-way commit		
Memory and L2 Cache	150 cycle memory latency, 512KB 4-way set associative unified (instruction and data) cache with a 64 byte block size and 12 cycle latency		
Functional Units	8 integer ALUs, 2 integer MULT/DIV, 2 FP ALU, 2 FP MULT/DIV, 2 load/store		

Table 1 presents the simulation parameters for the monolithic and microcore architectures we explore in this paper. Cache and register file access latencies are extracted from Cacti [21] for a *70nm* Technology at 5.6 GHz frequency.

Note that the difference in branch misprediction penalty is the extra latency attributed to the larger branch predictor, register file and instruction cache in the monolithic core.

4.2 Power and Thermal Simulator

A complete analysis of the static and dynamic power consumption and resulting temperature characteristics of different architectures is crucial to our study. Our power/thermal simulator performs cycle-accurate analysis of investigated architectures based on the following recently developed power and thermal models. We used process parameters for a *70nm* process at 5.6GHz with 1V supply voltage, in order to have a better understanding of next generation submicron, low supply voltage, aggressively clocked microprocessors.

We have incorporated Wattch [5] models for dynamic power analysis of the microprocessor blocks. The experimental results we present are extracted with the most aggressive conditional clocking strategy, where the dynamic power scales linearly with access to the ports.

For submicron technologies, such as 70nm, leakage power constitutes a significant portion of the overall power. ITRS [1] predicts that leakage power is likely to increase exponentially and make up 50% of the total power dissipation for the next deep submicron processes. Hence, an accurate and reliable leakage power analysis is a necessity. We adapted leakage models from Hotleakage [27] in our power/thermal simulator. Hotleakage models are extended and improved versions of the well-known Butts and Sohi leakage equations [7]. The public version of Hotleakage only provides a software implementation of the leakage models for the data cache. We have extended and modified the tool significantly to accommodate other caches and cache-like structures in the microprocessor. We also used leakage parameters from Hotleakage's predetermined values specific to the 70nm process technology.

A detailed and accurate thermal analysis of the different architectures we explore in this study is crucial. It has been shown by [14] that thermal metrics based on power consumption or power density of individual blocks do not provide accurate thermal estimation. We used Hotspot's thermal resistance/capacitance models and RC solvers for our analysis.

Dynamic and leakage power consumption for each microprocessor unit are collected over a predetermined thermal sampling interval, as the temperatures change over periods greater than every cycle. We experimented with various sampling interval lengths, in order to explore the trade off between error rate and computational overhead. Hotspot [14] proposes a 10K instruction sampling interval for 180nm and 3.3GHz, our results showed similar error rates for 10K sampling interval for 70nm and 5.6 GHz as well.

Our power/thermal simulator also incorporates the thermal runaway phenomena enabled by Hotleakage and Hotspot models. Thermal runaway is caused by the exponential dependency of leakage power on temperature: increased temperature increases leakage power, increased leakage power causes even further increase in temperature. The positive feedback loop between leakage power and temperature is quite significant and can cause device failure.

Heo, Barr and Asanovic [11], argue that most heat is dissipated vertically on the microprocessor chip, as the wafer thickness is much smaller than the chip area. Therefore, they assume infinite lateral resistances, although it leads to the

worst case temperature gradients. We follow their example, and tune HotSpot to only consider the vertical component of temperature. Lateral modeling, while possible with HotSpot, is unrealistic without a more accurate floorplan of the various architectures we consider.

Hotspot also requires a floorplan and the areas of the individual blocks of the microprocessor. We used area values based on our analysis with Cacti [21], along with a floorplan generated according to the minimum wirelength constraints. (Area values for the blocks are not presented in this version because of the page limitations.)

4.3 Dynamic Thermal Management Techniques

We assume that the critical thermal threshold is 82°C and the safety thermal threshold is 79°C for the 70nm technology process we are investigating according to the ITRS [1] projections and results from [14].

We have incorporated an idealized version of dynamic frequency scaling for the experimental analysis. Our DFS has two different frequency settings: 5.6GHz for the normal operation and 4GHz for thermal relief, which gets activated as soon as on-chip temperatures reach the 82°C critical thermal threshold. Usually there is a large latency (on the order of usecs) incurred every time the frequency is adjusted, which results in significant performance penalties in dynamic frequency scaling schemes. Skadron et al. [14] report 10usec for the non-idealized version of DFS. In our dynamic frequency scaling implementation there is no overhead, delay or penalty involved with changing the frequency of the processor.

Global clock gating is commonly used in many of todays microprocessors, such as the Pentium 4 as discussed in Section 2. We implemented a similar global clock gating mechanism for thermal analysis. The global clock signal is shut down, whenever on-chip temperatures exceed the critical thermal threshold of 82°C. The processor resumes normal operation after the chip temperatures cool down below the safety threshold of 79°C.

Our thermally-triggered core swapping mechanism gets activated as soon as a core reaches 82°C. The runs with this architecture assume an extra core (identical to the main core) that can be used to offload an application when one core overheats. The computation is migrated to the cooler core until the active core heats above the critical thermal threshold and another swap is required. Thermally-triggered core swapping minimizes the swapping overhead relative to approaches that swap at fixed intervals regardless of core temperature.

5 Experimental Results

In this section we evaluate the performance of the microcore architecture alone and in the presence of different DTMs. In particular, we examine the ability of the microcore to buffer state when core swapping, and compare this to a conventional monolithic architecture.

5.1 Thermal Characteristics of Microcore vs. Monolithic

Figure 4 compares the performance and thermal behavior of a conventional monolithic core and the microcore architecture on some of the SPEC 2000 benchmarks. The upper half of the figure shows performance in BIPS for different benchmarks, and the lower half illustrates the heating behavior of the investigated architectures. This latter component shows the percentage of cycles for which at least one block exceeds the indicated temperatures: 75°C, 79°C, 82°C and 85°C. Darker colors in the lower graphs indicate higher temperatures. The rest of the figures in this section are similarly constructed.

For example, `galgel` sees comparable performance with either the microcore or monolithic architecture, but the monolithic core sees a temperature greater than 85°C almost 97% of the time. The microcore only exceeds 85°C around 18% of the time, and stays below 82°C around 42% of the time.

Note that for many benchmarks, and particularly in monolithic architectures, temperature frequently exceeds the thermal threshold, 82°C. These results should be considered as an upper bound for performance that are not be achievable without some form of thermal management. On-chip temperatures for the microcore architecture are significantly lower than the monolithic core, but it still retains good performance comparable to that of the monolithic core. This can be attributed to the significantly smaller structures in the microcore that are much more power efficient.

Our detailed thermal analysis considers all of the possible overheating blocks. Although some of the hotspots were common among different benchmark, such as the register file, load-store queue, etc, others varied across the different bench-

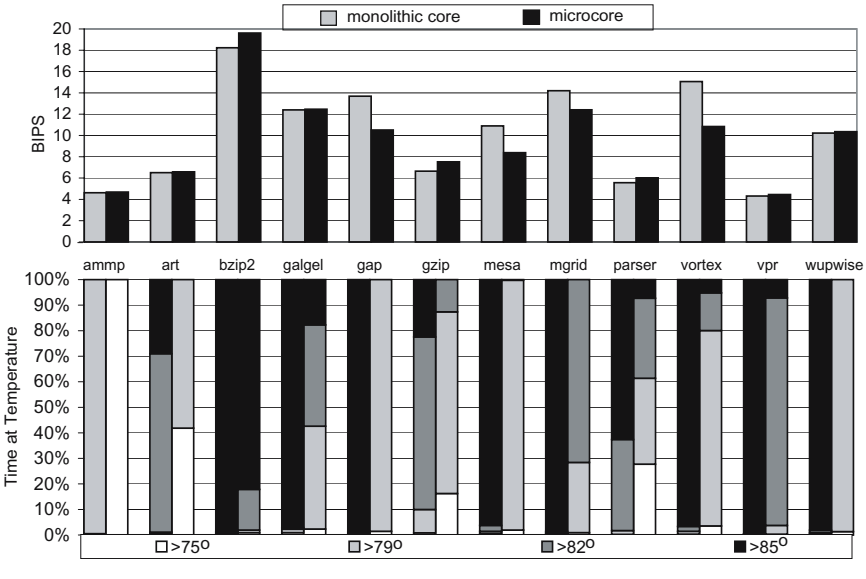

Fig. 4. Performance and thermal behavior of a microcore vs a monolithic core

marks and configurations. Even though the location of hotspots can provide a level of insight, the thermal behavior of the architecture can also be captured by the number of cycles that any of the blocks exceed a given thermal threshold.

The smaller structures of the microcore consume less power on each access compared to larger blocks in the monolithic architecture. Moreover, the larger helper engines are not accessed as frequently. Their inherent latency tolerance provides opportunities for power optimization. The microcore architecture shows performance comparable to the monolithic core, but with a 20% reduction in power on average.

It is important to note that the ITRS projects a reduction in maximum permitted junction temperatures for the future generations of process technologies. The maximum tolerated junction temperatures are around 85°C for 130nm and even lower for smaller process technologies.

The inherent thermal efficiency of the microcore also enhances the effective temperature reduction when used with DTM techniques. Next, we evaluate the performance and thermal behavior of DTM techniques, including core swapping, on the monolithic core and microcore.

5.2 Dynamic Thermal Management on Monolithic Architecture

Figure 5 shows core swapping results compared to no DTM, global clock gating, and an idealized version of dynamic frequency scaling on the monolithic architecture. The upper section of the graph displays performance in BIPS, the

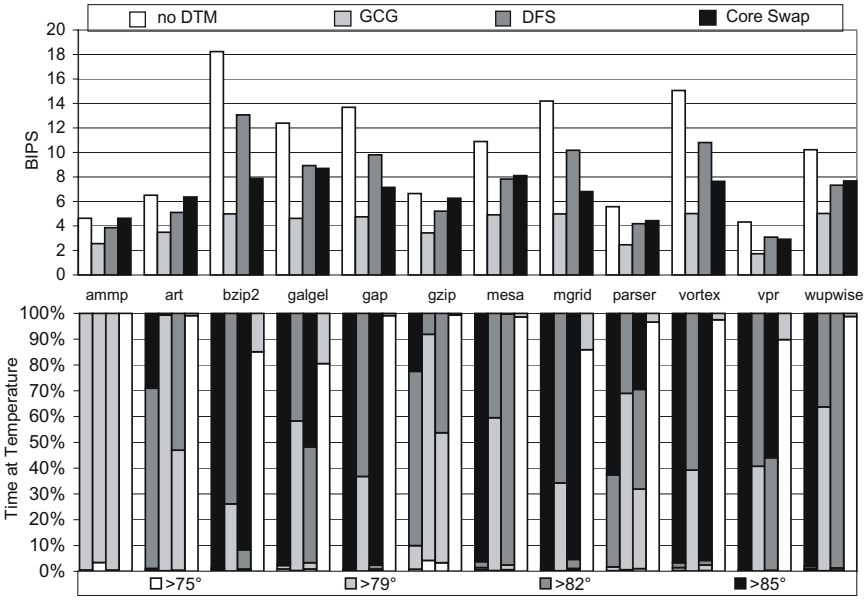

Fig. 5. DTMs on the monolithic architecture

lower part is dedicated to the thermal behavior of the same benchmark and DTMs, similar to the previous figure. Darker shades in the lower part of the figure indicate higher temperatures as well.

Core swapping results are shown in black bars (at the top part of the Figure), idealized dynamic frequency scaling in dark gray and global clock gating are in light gray. White bars demonstrate results without thermal management of any kind, no-DTM. As mentioned earlier in Section 1, performance degradation is commonly experienced with dynamic thermal management techniques. The degradation usually comes from various sources such as frequency decrease, voltage reduction, clock gating. Performance degradation might be quite significant depending on the DTM technique.

As a result no-DTM has the best performance results in BIPS among all cases. However, it is almost impossible to achieve comparable performance in reality since it would require sustained operation at a temperature beyond the critical thermal threshold, and a processor operating under such conditions would likely have timing, data and reliability complications. Although global clock gating seems to be more effective in reducing the temperature in most benchmarks than DFS, it has a very significant performance penalty as a result of disabling the global clock signal frequently.

Core swapping is extremely effective at thermal management, reducing the temperature below 79°C at least 80% of the time for all benchmarks and well above 95% of the time for many benchmarks. On the monolithic core, some applications are able to tolerate the performance impact of core swapping, but there is a pronounced degradation for many benchmarks, like `bzip2` and `mgrid`.

For the monolithic case, temperatures were still above the threshold for many applications with DFS, such as `bzip2`, `gap` and `mgrid`. This may indicate that our DFS strategy requires an even lower frequency to provide thermal relief to these applications, but at an even greater cost to performance. Despite a 70% drop in performance `mgrid` is still above 85°C around 95% of the time with DFS. `gap` operates in lower frequency mode almost 99% of the time in order to reduce the temperature, yet it is still above the 85°C temperature threshold 97% of the time.

5.3 Dynamic Thermal Management on Microcore

Figure 6 shows the behavior of the microcore with DTM techniques. We observe significantly improved thermal behavior compared to the monolithic architecture (Figure 5), and see less performance degradation from core swapping.

It is important to note that state buffering provided by the shared helper engines minimizes the core swapping overhead in the microcore architecture. Core swapping is always able to outperform the other DTMs on a microcore architecture, in most cases coming close to the performance of the architecture without any DTM. It has an equally dramatic impact on temperature in the microcore architecture. Temperatures are lower than 82°C with core swapping, for all of the benchmarks. Even `galgel`, which spends over half its execution time over 82°C is able to reduce its temperature below 79°C around 93% of the time using core swapping, with only an 8% degradation in BIPS.

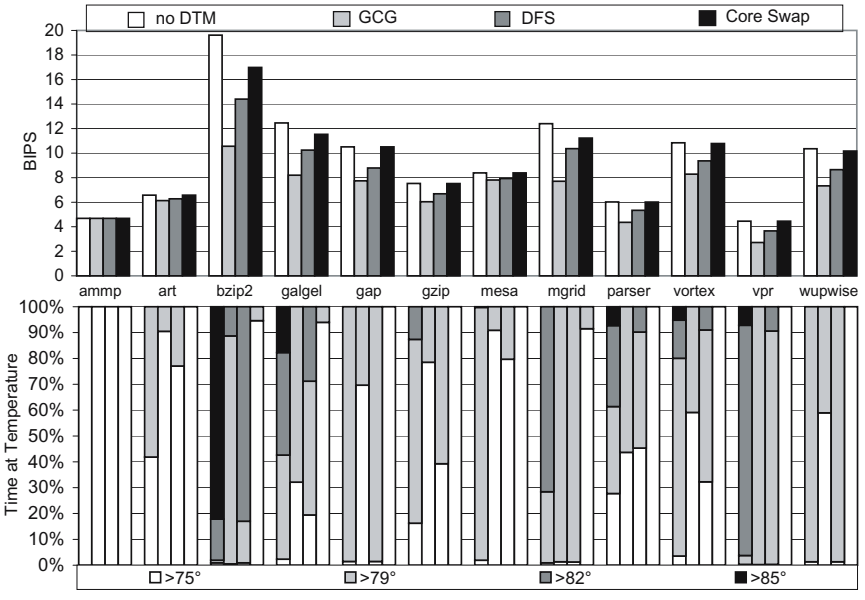

Fig. 6. DTMs on the microcore architecture

Note that we have used an idealized DFS implementation (see Section 4). This behavior can cause a significant performance degradation if frequency switching is used often. Notice that the idealized DFS is given competitive advantage against a core swapping approach with a realistic performance penalty. Despite this, core swapping is still able to outperform DFS.

6 Summary

In this paper, we investigated the thermal behavior of the microcore architecture, and examined the use of core swapping as a legitimate alternative to conventional DTMs.

We demonstrated that the microcore architecture enables lower on-chip temperatures compared with a conventional monolithic architecture. Factoring large, power-hungry units out of the core limits the number of accesses to such blocks and prevents them from heating as much. Our experiments show that the microcore reduces number of cycles over the critical thermal threshold by 86% on average, even without any thermal management use.

Furthermore, we have proposed a thermally-triggered core swapping mechanism as a dynamic thermal management technique. Microcores enable efficient core swapping by buffering processor state in shared helper engines that reduce startup costs when switching to a new core. Our experimental results indicate that a microcore is able to attain comparable IPC to a monolithic core, but with 94% fewer cycles above the critical thermal threshold.

The core swapping mechanism shows promising thermal reduction ability. It does not suffer any cycles in thermal violation for any of the benchmarks we examined. It also has favorable performance (as measured in BIPS) when compared to other DTM techniques such as GCG and the idealized DFS.

Future microprocessor generations have great thermal challenges awaiting them. Thermally efficient architectures and dynamic thermal management techniques are both critical to overcoming these challenges. Architectures like the microcore can help to achieve this without sacrificing performance.

References

1. In *International Technology Roadmap for Semiconductors*, 2003.
2. A. Ajami, K. Banerjee, M. Pedram, and L. van Ginneken. Analysis of non-uniform temperature-dependent interconnect performance in high performance ics. In *41st Design Automation Conference*, pages 567–572, June 2001.
3. R. Balasubramonian, S. Dwarkadas, and D. Albonesi. Reducing the complexity of the register file in dynamic superscalar processors. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, December 2001.
4. D. Brooks and M. Martonosi. Adaptive thermal management for high-performance microprocessors. In *Workshop on Complexity Effective Design*, June 2000.
5. D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimization. In *27th Annual International Symposium on Computer Architecture*, pages 83–94, June 2000.
6. D. C. Burger and T. M. Austin. The simplescalar tool set, version 2.0. Technical Report CS-TR-97-1342, U. of Wisconsin, Madison, June 1997.
7. J.A. Butts and G.S. Sohi. A static power model for architects. In *27th Annual International Symposium on Computer Architecture*, pages 191–201, June 2000.
8. D.Brooks and M.Martonosi. Dynamic thermal management for high-performance microprocessors. In *International Symposium on High-Performance Computer Architecture (HPCA-7)*, pages 171–182, January 2001.
9. M. Franklin and G. S. Sohi. Arb: A hardware mechanism for dynamic reordering of memory references. *IEEE Transactions on Computers*, 46(5), May 1996.
10. S. Gunther, F. Binns, D. Carmean, and J. Hall. Managing the impact of increasing microprocessor power consumption. In *Intel Technology Journal Q1*, 2001.
11. S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *International Symposium on Low Power Electronics and Design*, August 2003.
12. G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel. The microarchitecture of the pentium 4 processor. *Intel Technology Journal Q1*, 2001.
13. N. Jouppi. Improving direct-mapped cache performance by the addition of a small fully associative cache and prefetch buffers. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, May 1990.
14. K.Skadron, M.Stan, W. Huang, S.Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *30th Annual International Symposium on Computer Architecture*, pages 2–13, June 2003.
15. C-H. Lim, W. Daasch, and G.Cai. A thermal-aware superscalar microprocessor. In *International Symposium on Quality Electronic Design*, pages 517–522, March 2002.

16. L.T.Yeh and R.Chu. Thermal management of microelectronic equipment. In *American Society of Mechanical Engineers - ISBN:0791801683*, 2001.
17. G. Reinman, T. Austin, and B. Calder. A scalable front-end architecture for fast instruction delivery. In *26th Annual International Symposium on Computer Architecture*, May 1999.
18. A. Shayesteh, E. Kursun, S. Sair, T. Sherwood, and G. Reinman. An evaluation of deeply decoupled cores. In *University of California Los Angeles Tech Report CS-2004-09*, 2004.
19. T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2002.
20. T. Sherwood, S. Sair, and B. Calder. Predictor-directed stream buffers. In *33rd International Symposium on Microarchitecture*, December 2000.
21. P. Shivakumar and Norman P. Jouppi. Cacti 3.0: An integrated cache timing, power, and area model. In *Technical Report*, 2001.
22. J. E. Smith. Instruction-level distributed processing. *IEEE Computer*, 34(4):59–65, April 2001.
23. R. Viswanath, V. Wakharkar, A. Wathe, and V.Lebonheur. Thermal performance challenges from silicon to systems. In *Intel Technology Journal Q3*, 2000.
24. K. Wang and M. Franklin. Highly accurate data value prediction using hybrid predictors. In *30th Annual International Symposium on Microarchitecture*, pages 281–290, December 1997.
25. W.Huang, J.Renau, S-M.Yoo, and J. Torrellas. A framework for dynamic energy effiency and temperature management. In *33rd International Symposium on Microarchitecture*, pages 202–213, December 2000.
26. T. Yeh and Y. Patt. A comprehensive instruction fetch mechanism for a processor supporting speculative execution. In *Proceedings of the 25th Annual International Symposium on Microarchitecture*, pages 129–139, December 1992.
27. Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. In *University of Virginia Dept of Computer Science Tech Report CS-2003-05*, March 2003.