Resource Area Dilation to Reduce Power Density in Throughput Servers

Michael D. Powell[1]
Fault Aware Computing Technology Group
Intel Massachusetts, Inc.
michael.d.powell@intel.com

T. N. Vijaykumar
School of Electrical and Computer Engineering
Purdue University, W Lafayette, IN
vijay@purdue.edu

ABSTRACT

Throughput servers using simultaneous multithreaded (SMT) processors are becoming an important paradigm with products such as Sun's Niagara and IBM Power5. Unfortunately, throughput-computing via SMT aggravates power-density problems because SMT increases utilization, decreasing cooling opportunities for overheated resources. Existing power density techniques are: slowing computation and lowering supply voltage, which is likely infeasible in future technologies; stopping computation to reduce heating, which substantially degrades performance; or migrating computation to spare resources, which adds complexity; or requiring underutilized resources, which may not be available in an SMT-based throughput server.

An alternative is to increase the area of heat-prone resources at design time. We propose the concept of dilation where a resource's circuit components are spread over an area larger than required for correct logic. Increasing area allows the resource to be utilized more without violating power-density constraints. This paper is the first to consider increasing *CPU resource area for improving throughput in a thermally-constrained processor. Dilating area to improve performance seems counterintuitive because it also increases the latency of the components. However this technique is uniquely effective in SMT-based throughput computing because having multiple threads from which to choose instructions makes SMTs more tolerant of added latency than superscalars. We propose two implementations. Our first implementation, Simple Resource Area Dilation (S-RAD), increases the area of heat-prone resources and scales the CPU clock frequency accordingly. Our second implementation, Pipelined Resource Area Dilation (P-RAD), pipelines the dilated resources to maintain clock frequency.*

Categories and Subject Descriptors

C.1.0 [**Processor Architectures**]: General

General Terms

Performance, Reliability.

Keywords

Power Density, Throughput Servers, SMT

1 INTRODUCTION

Power density is a growing problem in high-performance processors in which small, high-activity resources such as functional units overheat. Power density increases with technology generations as scaling of clock speed, processor current, and device area further exceeds the ability of affordable packages to dissipate heat away from the hot spots.

Throughput servers using Simultaneous multithreaded (SMT) processors [4] are becoming an important paradigm with products such as the Sun Niagara [14] and IBM Power5 [5]. However, SMT aggravates power-density problems because SMT increases utilization, decreasing cooling opportunities for overheated resources by overlapping natural idle periods in one thread (e.g., cache misses) with the running of other threads [15]. A recent study showed that adding threads in a thermally-constrained SMT can actually *decrease* throughput due to increased overheating [9]. Another study showed that SMT was prone to hot spots [8].

Existing power density techniques are: slowing computation and lowering supply voltage, which is likely infeasible in future technologies; stopping computation to reduce heating, which substantially degrades performance; or migrating computation to spare resources, which adds complexity; or requiring underutilized resources, which may not be available in an SMT-based throughput server. We discuss details of these approaches in Section 2.

An alternative to slowing, stopping, or migrating computation is to increase the area of heat-prone resources at design time. To that end we propose the concept of resource area dilation (RAD) where a resource's circuit components are spread over a die area larger than that required for correct logic; gates are not increased in size but local wires between them are lengthened, spreading the resource's heat over a larger area. RAD allows the resource to be utilized more (and to consume more power) without overheating. This paper is the first to consider increasing CPU resource area for improving throughput in a thermally-constrained processor.

Increasing the area of processor resources seems counterintuitive from a conventional processor-design standpoint because larger resources means longer wires and thus increased resource latency. However, SMT changes the "faster-is-better" mentality associated with superscalars because SMT is more latency-tolerant than superscalars. Indeed, the key result of this paper is that in throughput-oriented environments such as servers, the throughput degradation from longer latency is more than offset by the throughput increase from reduced overheating. Of course, RAD's latency increase will degrade single-thread performance. However, single-thread performance is not the goal of products such as Sun's Niagara [14], which already sacrifices individual core performance for reduced design complexity and high throughput over many threads. Furthermore, throughput-oriented server processors such as the Intel Xeon tend to be separate product lines manufactured on dies different than single-thread-oriented processors (e.g. desktop or mobile) such as the Intel Centrino. Therefore RAD can be utilized on server processors without impacting single-thread processors.

1. This work was completed while Michael D. Powell was a graduate student at Purdue University.

RAD has several advantages over conventional frequency scaling (FS). An increase in area alone has a linear effect on power density. However, if frequency is scaled linearly to compensate for the increased latency associated with increased area, the combined reduction in power density is quadratic. Conventional FS incurs this increased latency but does not exploit the opportunity to increase area. We are the first to exploit the fact that adding RAD to frequency scaling increases power-density improvements without further degrading performance. While RAD achieves only a quadratic reduction compared to DVFS's cubic reduction, RAD is feasible in future technologies whereas DVFS is likely not.

A concern for RAD is increased die area. However we show that the area increase is small (less than 2%) because RAD applies only to heat-prone core resources and not to the caches which consume the vast majority of the die.

We propose two implementations of RAD. Our first implementation, called Simple Resource Area Dilation (S-RAD), increases the area of heat-prone resources and scales clock frequency of the entire CPU accordingly, resulting in a quadratic reduction in power density in the heat-prone resources. This implementation is simple because it amounts to a reverse die-shrink on existing resources without regressing in process technology (feature size).

We alleviate the latency penalties of RAD by pipelining the dilated resources in our second implementation, called Pipelined Resource Area Dilation (P-RAD), instead of decreasing clock frequency as done in S-RAD. Pipelining critical resources such as instruction issue increases misspeculation penalties and makes it hard to issue consumer instructions immediately after their producer. In SMT, however, the misprediction and dependence issues are mitigated by the presence of independent instructions from multiple threads. The throughput improvements from reducing power density outweigh the throughput loss from increased penalties. Pipelining also slightly increases power due to additional latches; but again this increase is small compared to the reduction in power density.

The main results of this paper are:

- In a 2-context SMT, S-RAD improves throughput for thermally-constrained workloads by an average of 42% over an SMT using stop-go, and by an average of 6% (31% for severely thermally-constrained workloads) over an SMT using dynamic frequency scaling.
- In a 2-context SMT, P-RAD improves throughput in thermally constrained workloads by an average of 41% over an SMT using stop-go, and by an average of 5% (56% for severely thermally-constrained workloads) over an SMT using dynamic frequency scaling.

The rest of this paper is organized as follows. Section 2 discusses related work on power density. Section 3 describes S-RAD, and Section 4 describes P-RAD. We explain our experimental methodology in Section 5 and present experimental results in Section 6. Finally, we conclude in Section 7.

2 RELATED WORK

2.1 Superscalar

Existing power-density mitigation techniques fall into two categories: temporal and spatial migration. Temporal proposals which trade off latency for power density include balancing chip-level heat production rate to heat dissipation [1] and fetch-throttling and use of PID controllers [12]. Heating can be reduced at coarse granularity by temporarily stopping computation to allow cooling, called *stop-go* [6] (same as Intel Pentium 4's thermal throttling), or at fine granularity by slowing computation through dynamic frequency scaling (DFS) or dynamic frequency and voltage scaling (DVFS) [13]. The slowing or stopping results in performance degradation. DVFS achieves a quadratic reduction in power from voltage scaling and a linear reduction from dynamic frequency scaling for a combined cubic reduction. However, voltage scaling will be difficult or impossible in future technologies. As the per-generation decrease in nominal supply voltages slows or stops because of circuit reliability concerns, there is little headroom for further supply-voltage scaling. (The ITRS shows nominal supply voltage dropping by only 100 mV between 2005 and 2011 [11].) If voltage scaling is not possible, the remaining temporal techniques, stop-go and frequency-scaling, achieve only linear power reduction.

Spatial migration solutions reduce heat by moving computation from a hot resource to an alternate resource copy (e.g., a spare ALU). Migration has a small impact on performance, but requires availability of spatial slack in the form of spare or underutilized resource copies. Unfortunately, SMTs are much less likely than superscalars to have spare or underutilized resources. One paper [13] proposes duplicating the integer register file, which would add substantial wiring and select-logic complexity. Another paper [7] proposes "ping-ponging" resource activity for various pipeline resources between duplicate resource copies within a superscalar core but does not address the scheduling-logic or wiring complexity of providing these duplicates. While we also trade off area for power density like [13] and [7], we only dilate the area of existing resources, which is substantially simpler to implement.

2.2 SMT and CMP

Three previous proposals have considered power density in SMT and chip multiprocessor (CMP) architectures. [9] proposes a spatial technique for migrating and assigning threads on SMT CMPs but does not address power density within a single SMT core. [8] evaluates power density tradeoffs between SMTs and CMPs and concludes that SMTs have more hotspots than single-thread CMP cores because they concentrate more execution in a smaller area. However, replacing SMTs with many single-thread-cores in CMPs is not desirable because SMT is needed to maintain high throughput as the gap between processor clock speed and off-chip latency increases. In future technologies where off-chip latencies are many-hundreds to a few thousand cycles, even narrow-issue CMP cores will need SMT to keep execution resources busy during cache misses. Consequently, addressing the power-density challenges of SMT is critical.

Finally, [3] discusses tradeoffs between spatial and temporal techniques in a CMP environment. [3] combines DVFS and thread migration and evaluates the tradeoffs among these techniques. [3] does not consider changing core area to mitigate power density.

3 SIMPLE RESOURCE AREA DILATION

Simple Resource Area Dilation (S-RAD) dilates the area of heat-prone processor resources and slows the frequency of the entire processor to compensate for the increased latency of those components. This section describes how much the frequency needs to be reduced for S-RAD and the impact of S-RAD on power and die area.

S-RAD is implemented by dilating the wires between logic gates (or sub-blocks of logic gates within resources) to increase the area of heat-prone resources such as issue queues, register files, and ALUs. Lengthening local wires within a resource is adequate to spread the resource's heat; resizing the logic gates themselves is undesirable because of latency and power.

Table 1: Impact of S-RAD assuming a dilation factor of *A*.

Value	Original	Dilated
Resource length	X	$\sqrt{A}X$
Resource height	Y	$\sqrt{A}Y$
Resource area	XY	AXY
Dilated wire resistance	R	$\sqrt{A}R$
Dilated wire capacitance	C	$\sqrt{A}C$
Dilated wire latency	T	AT
Processor clock frequency	F	F/A

Because S-RAD does not guarantee that no thermal violations will occur, we use stop-go on top of S-RAD as a safety net. It would also be possible to use other techniques such as frequency scaling, but doing so would add complexity.

3.1 Frequency and Latency

The impact of S-RAD on latency can be expressed in terms of the resource dimensions and the RC (resistive-capacitive) delay of the wires. Because dilating by a factor of A increases resistance and capacitance each by a factor of square-root A, latency increases by a factor of A, and processor frequency must be divided by A. Our calculations are summarized in Table 1. Of course, real designs may have some slack which would allow some dilation without any clock-period increase. However, in this paper we assume the worst case that there is no slack.

3.2 Impact on Power

The power consumed by a wire is proportional to its capacitance, meaning the maximum power increase from S-RAD would be a factor of the square root of the dilation. Even for this worst-case increase, S-RAD would still reduce power density because the increase in area exceeds the increase in power. Fortunately, the power of a dilated resource does not increase that much for two reasons. First, not all wires in a dilated resource are lengthened; only wires between large subblocks of gates which are a small fraction of wires. Second, not all power in a resource comes from wires; much of it comes from logic gates which are not dilated. (Some logic gates might need to be increased slightly in size to drive longer wires, but the number of these gates is small compared to the total number of gates in a resource.)

3.3 Impact on Area

S-RAD reduces power density by increasing the area of critical components. However, the net impact of S-RAD on processor die area is small because the dilated resources represent a small fraction of die area. Only certain resources, such as issue queues, register files, and ALUs, are prone to overheating and require dilation. These resources represent around 25% of core area (excluding L2 caches) (based on [13]), and the core is around 15% of the die area (including L2 caches). Dilating 25% of 15% of the die by even a large dilation factor of 1.4 causes less than 2% net increase. One could offset this increase by slightly reducing the cache size (SMT can absorb the increased misses).

4 PIPELINED RESOURCE AREA DILATION

Pipelined Resource Area Dilation (P-RAD) avoids the clock-frequency degradation of S-RAD by pipelining the dilated resources. P-RAD maintains the same clock frequency as a conventional system while pipelining dilated resources which are prone to overheating: instruction issue, register files, and ALUs.

P-RAD also uses stop-go as a safety net just like S-RAD as discussed in Section 3.

4.1 Pipelining and Area

Pipelining dilated resources allows a substantial area dilation while maintaining clock frequency because of the discrete nature of pipeline-stage latencies in whole clock cycles. Because the latency increase due to dilation must be at least a whole cycle (and not fractional cycles), a large area dilation is possible to fully utilize the extra latency. Using the area-latency relationship established in Section 3.1, we find that if we double a resource's latency from one cycle to two cycles, we can double its area. If we dilate a 2-cycle resource to 3 cycles, we can increase its area by up to half. Of course, the overhead of pipeline latches reduces the possible area dilation somewhat, but area dilations large enough to alleviate serious power density problems are still feasible.

4.2 Pipelining and Throughput

Pipelining resources in an out-of-order processor increases misspeculation penalties and lengthens some dependence loops. Pipelining each of the issue queue, register file, and ALUs directly increases the branch misprediction penalty and d-cache miss replay penalty by the number of stages added to the pipeline. The architectural pipeline stages (many of which already consist of several circuit stages in modern designs) and our extra circuit stages for dilation are shown in Figure 1.

Pipelining the instruction issue stage also creates a problem with completing operand wakeup and instruction select in a single cycle. If these operations are split across two pipeline stages, it is not possible to issue a dependent instruction back-to-back with its producer. Pipelining execution resources creates a similar concern in that a consumer can not immediately follow a producer in the pipeline. Fortunately, this penalty need be paid only once even if multiple resources are pipelined.

In a single-thread superscalar environment, this back-to-back problem can cause performance degradation. However, in a throughput-oriented SMT environment, independent instructions from different threads should be available to issue, minimizing throughput degradation from disallowing back-to-back issue. It is important to note that single-thread performance is not sacred for *all* markets. Conventional SMTs may already deepen pipelines to accommodate larger resources (such as register files) and incur performance penalties relative to the equivalent superscalar, single-

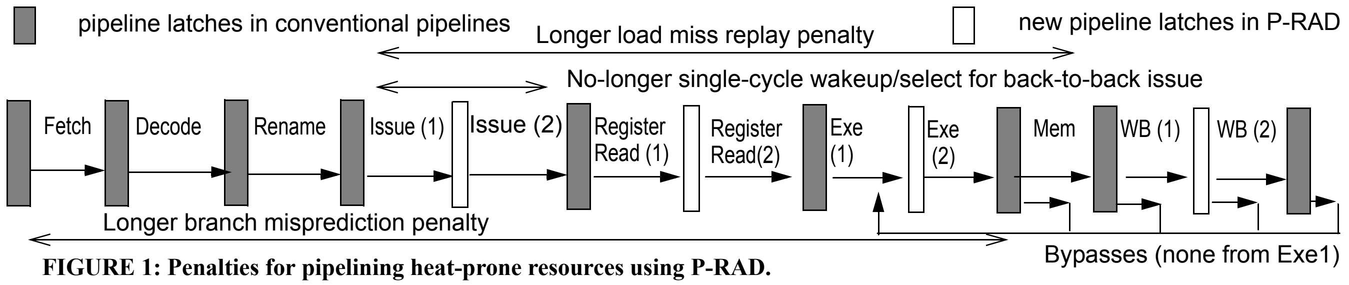
pipeline latches in conventional pipelines
Longer load miss replay penalty
new pipeline latches in P-RAD
No-longer single-cycle wakeup/select for back-to-back issue
Fetch
Decode
Rename
Issue (1)
Issue (2)
Register Read (1)
Register Read(2)
Exe (1)
Exe (2)
Mem
WB (1)
WB (2)
Longer branch misprediction penalty
Bypasses (none from Exe1)

FIGURE 1: Penalties for pipelining heat-prone resources using P-RAD.

Table 2: Base Processor Parameters

Out-of-order issue	6 instructions/cycle
Active list	128 entries (64-entry LSQ)
Issue queue	32-entries each Int and FP
Caches	64KB 4-way 2-cycle L1s (2 ports); 2M 8-way unified L2
Memory latency; L2 latency	72 ns; 2.9 ns
Branch misprediction penalty	15 cycles
Heatsink thickness	6.9 mm
Convection resistance	0.8 K/W
Thermal cooling time	10 ms
Maximum temperature	358 K
Circuit/process parameters	4.2 GHz; 1.2V; 90nm

thread pipeline. P-RAD may be thought of as a design point within this continuum of design choices which favor throughput-oriented markets over single-thread markets. Recall from Section 1 that S-RAD and P-RAD do not target single-thread workloads because single-thread workloads do not have either the power-density problems or latency insensitivity of SMT workloads. Also recall from Section 1 that because manufacturers produce different processors on different dies for the server (multi-thread) market and desktop (single-thread) market (e.g., Intel Xeon vs. Intel Centrino), server processors may gain the benefits of RAD without penalizing processors targeted at single-thread workloads.

There are techniques to maintain back-to-back instruction issue despite pipelining issue [10]. Using these techniques would reduce P-RAD's performance degradation. However, because these techniques bring their own complexity concerns, we *do not* assume any of these techniques and instead charge the full penalties described in this section. In that sense, our results are conservative.

4.3 Pipelining and Power

In addition to the power increases from S-RAD described in Section 3.2, pipelining adds new latches between stages. Although these latches slightly increase the power of the dilated resource, the power density impact is more than offset because of the large area increases possible with P-RAD.

5 METHODOLOGY

We model power density by using HotSpot [13] which is built over Wattch [2]. We extend Wattch to include SMT with Icount [16]. We sense temperature at 100,000 cycle intervals (well under the thermal RC time constant of any resource). Architectural configuration and circuit and packaging parameters are shown in Table 2. The parameters are consistent with ITRS estimates for high-performance air-cooled designs in the next 5 years [11].

We use the superscalar floorplan provided in [13] as our baseline, and then scale it to 90nm while adding 10% to the area for SMT's overhead over superscalar. For RAD, we maintain the same floorplan and dilate the area of the heat-prone resources: integer issue queue, integer and floating-point register file.

We simulate S-RAD with 10%, 20% and 30% dilation of the integer issue queue, integer execution units, integer register file, floating-point adders, and floating-point multiplier, which are the components that overheat in our simulations. These S-RAD configurations reduce the clock frequency to 3.8 GHz, 3.5 GHz, and 3.2 GHz respectively. We also model the power overheads described in Section 3.2. Net die area increase ranges from 0.4% with 10% dilation to 1.3% with 30% dilation.

We simulate P-RAD with 60% dilation where integer issue queue, execution units, and register file are pipelined from one to two cycles. The floating-point adders and multipliers are pipelined from 2 and 4 cycles to 4 and 8 cycles, respectively. We account for the performance and power penalties discussed in Section 4.2 and Section 4.3. Though this pipelining would allow latency for 100% dilation, we conservatively dilate only 60%. The net die area increase is 2.6%.

We also compare to dynamic frequency scaling (DFS) and dynamic voltage-frequency scaling (DVFS). We implement a PI-controller as described in [13] with a gain of 10 and setpoint of 82 degrees, and allow configuration changes no more frequently than every 400000 cycles to avoid thrashing. We model a 10 ms overhead for frequency changes, which is typical of current systems. For DFS and DVFS, we allow five equally-spaced frequency and voltage steps between 4.2 and 2.1 GHz and 1.2 and 1.0 V.

We model throughput by running two-thread groupings of SPEC2000. We fast-forward each thread five billion instructions, warm up the caches (cache state, not temperature) for the last one billion instructions of warmup on each thread, and run until one thread completes 500 million instructions. Because S-RAD changes frequency, we measure throughput in instructions per second (IPS) and not instructions per cycle (IPC).

We choose to show 16 pairs of SPEC2K applications that are thermally constrained (i.e., lose performance due to power density) and 5 that are not. The pairings shown within each category are chosen to maximize the number of SPEC applications represented in our results. To set the initial temperatures correctly, we run customized trial runs for each workload using each specific technique under evaluation.

6 RESULTS

In this section we present our experimental results. Section 6.1 shows throughput for various S-RAD and P-RAD configurations that outperform stop-go for thermally constrained workloads. Section 6.2 compares S-RAD and P-RAD to dynamic frequency scaling (DFS) and dynamic voltage-frequency scaling (DVFS) which are other power density techniques.

6.1 S-RAD and P-RAD

Figure 2 shows our results for S-RAD and P-RAD for two-thread workloads. Our workloads are shown along the x-axis along with their base throughput in billions of instructions per second (GIPS) using the stop-go power density technique at 4.2 GHz with no area dilation. Thermally-constrained workloads are on the left; unconstrained workloads are on the right. The S-RAD and P-RAD simulations also use stop-go if they overheat. The bottom graph shows the *duty cycle* of the simulations, which represents the fraction of the time the processor is running and not stopped due to overheating. The top graph shows throughput relative to the base. The bars for each configuration, a (the base) through e, show the different configurations.

From the bottom graph, we see the quadratic power density reduction from frequency and area of S-RAD is effective at raising duty cycles in constrained workloads. 30% dilation achieves a 100% duty cycle (i.e., it eliminates power density problems) in all but one workload. P-RAD also raises all duty cycles to 100%, not because of reduced frequency but because of large area increase. (We ran additional simulations, not shown, with P-RAD and a 40% area dilation, and not all duty cycles were 100%.)

The top graph shows throughput for S-RAD and P-RAD. Workloads that originally had low duty cycles have the largest improvements. However, once a workload reaches a 100% duty cycle,

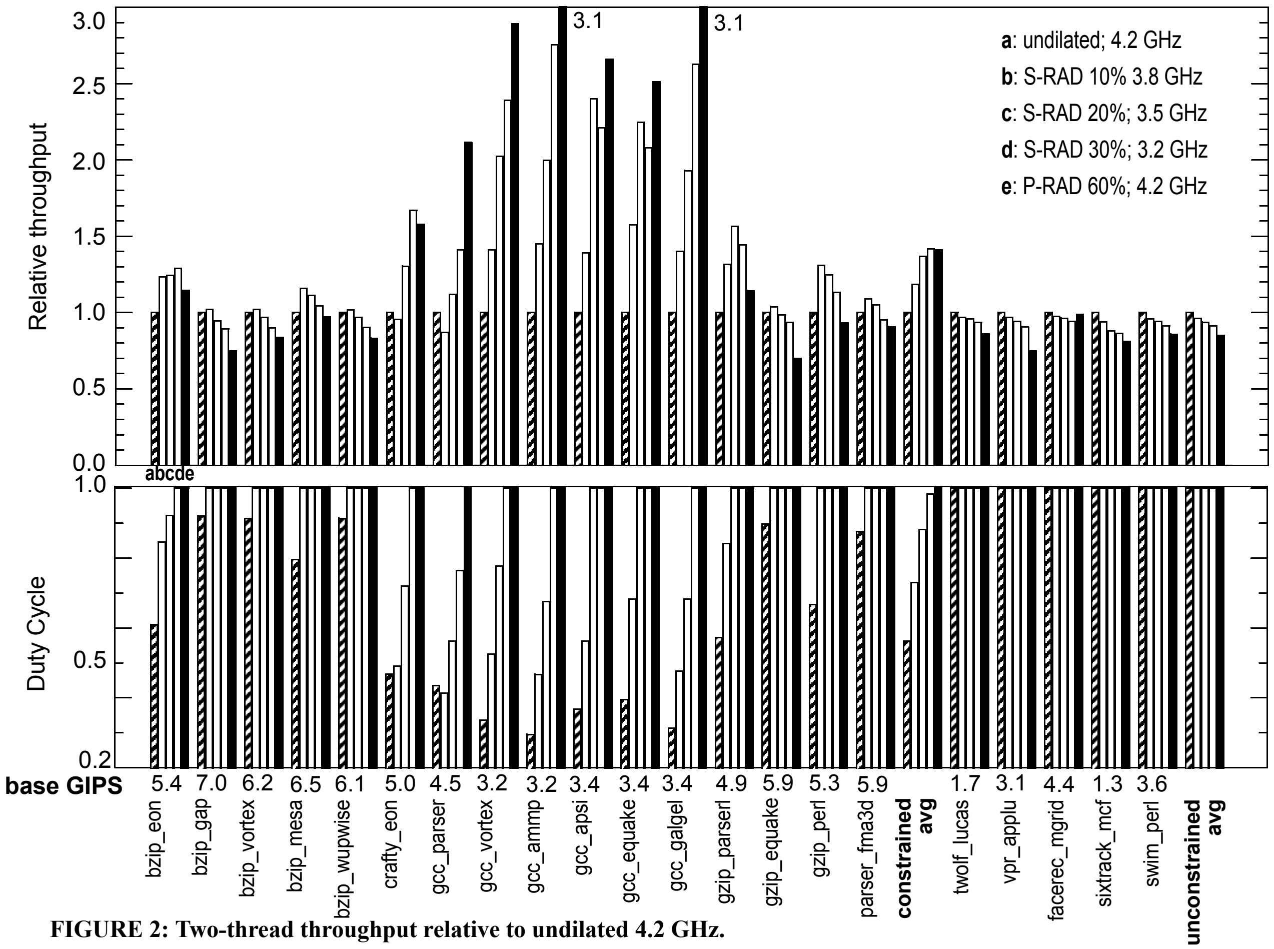

FIGURE 2: Two-thread throughput relative to undilated 4.2 GHz.

additional dilation degrades throughput because clock frequency is reduced with no additional benefit as seen in workloads such as *bzip_gap*. On average for thermally-constrained benchmarks, S-RAD with 10%, 20%, and 30% dilation achieves 19%, 37%, and 41% better throughput than stop-go alone. P-RAD achieves 41% better throughput on average for constrained workloads and the best individual speedups. P-RAD experiences dramatic speedups for frequency-sensitive thermally-constrained workloads like *gcc_galgel* but loses throughput in less frequency-sensitive workloads due to penalties discussed in Section 4.2.

Thermally-unconstrained workloads initially have a 100% duty cycle and thus do not benefit from S-RAD or P-RAD. On average 10% dilation, 20% dilation, 30% dilation, and P-RAD degrade throughput for unconstrained workloads by 4%, 6%, 9%, and 15%.

Using 20% dilation achieves good overall throughput improvement while avoiding both problems with 30% dilation in S-RAD and P-RAD cases like *bzip_gap, gzip_equake,* and *vpr_applu*.

While we show only two-thread results, larger numbers of threads will only aggravate power density problems more, improving our results. Therefore our two-thread choice is conservative.

6.2 Comparison to other power-density techniques

In this section we compare S-RAD and P-RAD to dynamic frequency scaling (DFS) and dynamic voltage-frequency scaling (DVFS). We compare to both DFS and DVFS because voltage-frequency scaling and its cubic reduction in power density may not be feasible in future technologies. While we could have improved the S-RAD technique by scaling voltage along with frequency reductions associated with dilation, we scale only frequency to show that S-RAD is effective *without* voltage scaling and thus applicable to future technologies where voltage scaling may be unavailable. We expect S-RAD and P-RAD to outperform DFS and DVFS for thermally-constrained workloads but to underperform for unconstrained workloads.

One key to comparing RAD to conventional temporal schemes is understanding high resource utilization in SMT compared to single thread. Table 3 shows average IPC for all of our 1-thread and 2-thread workloads using the stop-go technique. The IPC is divided by average duty cycle (1.00 for all of the 1-thread SPEC2K workloads) to compute the “active IPC,” which is the IPC excluding the periods when the processor was stopped. 2-thread workloads have an active IPC 29% above 1-thread workloads, demonstrating the higher resource utilization in SMT. As mentioned in Section 1, conventional temporal power-density techniques such as stop-go and frequency scaling reduce power density by reducing utilization at either coarse grain (stop-go) or fine granularity (frequency scaling). In high-utilization workloads, there is little opportunity for these techniques to reduce power density without substantially reducing utilization and thus throughput.

Table 4 shows our results for the power-density comparison relative to the 4.2 GHz undilated base configuration. Results are averaged across classes of workloads based on thermal constraint. The

Table 3: Single-thread vs. SMT utilization

Workloads	IPC	Duty cycle	Active IPC
1-thread	1.10	1.00	1.10
2-thread	0.95	0.67	1.42
Active IPC ratio of 2-thread:1-thread = 1.29			

Table 4: Avg. 2-thread throughput relative to undilated 4.2GHz.

Configuration	Con-strained	Severely constrained	Uncon-strained
S-RAD 20%; 3.5 GHz	1.37	1.86	0.94
S-RAD 30%; 3.2 GHz	1.42	2.16	0.91
P-RAD 60%; 4.2 GHz	1.41	2.57	0.85
DFS (4.2-2.1 GHz)	1.34	1.62	1.00
DVFS (4.2-2.1 GHz, 1.2-1.0 V)	1.36	1.58	1.00

classes constrained and unconstrained are as defined before, and *severely thermally-constrained* workloads are those with duty cycles less than 50% in Figure 2 (*crafty_eon* through *gcc_galgel*).

RAD has higher throughput than stop-go, DFS, or DVFS in high-utilization thermally-constrained workloads because resource dilation mitigates power density with less utilization impact than these techniques. On average, S-RAD with 30% dilation has 6% higher throughput than DFS and 4% higher throughput than DVFS for thermally-constrained workloads. For the seven severely thermally-constrained workloads, S-RAD with 30% dilation has 31% better average throughput than DFS (and is 27% better than DVFS). P-RAD has 56% better average throughput than DVFS (and is 51% better than DVFS).

For the unconstrained workloads, DFS and DVFS do not suffer throughput degradation because of their dynamic nature. For similar reasons DFS and DVFS outperform RAD in some workloads with small thermal constraints. (e.g., high duty cycles in Section 6.1) However, more and more multi-threaded workloads in future SMT processors are likely to be thermally constrained, and S-RAD and P-RAD will improve performance in these workloads. Workloads with more than two threads in future SMTs will also behave more like the severely thermally-constrained workloads where RAD substantially outperforms DVS and DVFS.

7 CONCLUSIONS

Throughput servers using simultaneous multithreaded (SMT) processors are becoming important (e.g., the Sun Niagara and IBM Power5). Unfortunately, throughput-computing via SMT aggravates power-density problems because SMT increases utilization, decreasing cooling opportunities

We target the density component of power density by increasing the area of heat-prone resources at design time. We propose the concept of dilation where a resource's circuit components are spread over a die area larger than required for correct logic. Increasing area allows the resource to be utilized more (and to consume more power) without violating power-density constraints and overheating. Though dilating area increases the latency of the resources, this technique is uniquely effective in SMT-based throughput computing because SMT is more latency-tolerant of than superscalars.

We propose two implementations of dilation. Our first implementation, Simple Resource Area Dilation (S-RAD), increases the area of heat-prone resources and scales clock frequency of the entire CPU accordingly. In a 2-context SMT, S-RAD improves throughput for thermally-constrained workloads by an average of 42% over an SMT using stop-go, and by an average of 6% (31% for severely thermally-constrained workloads) over an SMT using dynamic frequency scaling (DFS). Our second implementation, Pipelined Resource Area Dilation (P-RAD), pipelines dilated resources to maintain clock frequency. In a 2-context SMT, P-RAD improves throughput in thermally constrained workloads by an average of 41% over an SMT using stop-go, and by an average of 5% (56% for severely thermally-constrained workloads) over an SMT using DFS.

References

[1] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Seventh International Symposium on High Performance Computer Architecture (HPCA)*, pages 171–182, Jan. 2001.

[2] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 83–94, June 2000.

[3] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA 33)*, pages 78 – 88, June 2006.

[4] S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, R. L. Stamm, and D. M. Tullsen. Simultaneous multithreading: A platform for next-generation processors. *IEEE Micro*, 17(5):12–19, Sept. 1997.

[5] S. Finnes. *iseries.myseries*. http://www-1.ibm.com/servers/uk/media/iseries_skillbuilder/POWER5DeliverWith% outDisruption1.pdf, 2004.

[6] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall. Managing the impact of increasing microprocessor power consumption. In *Intel Technology Journal Q1 2001*, Q1 2001.

[7] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 217–222, Aug. 2003.

[8] Y. Li, D. Brooks, Z. Hu, and K. Skadron. Performance, energy, and thermal considerations for smt and cmp architectures. In *Eleventh International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2005.

[9] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat and run: Leveraging smt and cmp to manage power density through the operating system. In *Proceedings of the Eleventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XI)*, pages 260–270, Oct. 2004.

[10] S. Raasch, N. Binkeri, and S. Reinhardt. A scalable instruction queue design using dependence chains. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 318–329, June 2002.

[11] SIA. *International Technology Roadmap for Semiconductors (ITRS)*. http://www.itrs.net/Links/2005ITRS/PIDS2005.pdf, 2005.

[12] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *Eighth International Symposium on High Performance Computer Architecture (HPCA)*, pages 17–28, Feb. 2002.

[13] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the 30th International Symposium on Computer Architecture (ISCA 30)*, pages 2–13, June 2003.

[14] Sun Microsystems. Throughput computing. http://www.sun.com/processors/throughput.

[15] D. M. Tullsen and J. A. Brown. Handling long-latency loads in a simultaneous multithreading processor. In *Proceedings of the 34th International Symposium on Microarchitecture (MICRO 34)*, pages 318–327, Dec. 2001.

[16] D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm. Exploiting choice: instruction fetch and issue on an implementable simultaneous multithreading processor. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 191–202, June 1996.