This is the authors' version. The authoritative version will appear in the Elsevier Journal of Sustainable Computing, Informatics and Systems (SUSCOM)

Thermal Benefit of Multi-core Floorplanning: A Limits Study

Karthik Sankaranarayanan*, Brett H. Meyer*, Mircea R. Stan†, and Kevin Skadron*
Computer Science Department* and Electrical and Computer Engineering Department†
University of Virginia
Charlottesville, VA 22904 USA
ks4kk@virginia.edu, bhm@cs.virginia.edu, mircea@virginia.edu, skadron@cs.virginia.edu

Abstract

As transistors scale, system temperatures are rising, and with them, cooling costs. Faced with such challenges, designers have developed a variety of techniques to reduce temperatures at design-time, through floorplanning, and at run-time, using dynamic thermal management. Multi-core floorplanning, in particular, presents unique opportunities for temperature management: for example, cores can be individually floorplanned to reduce system temperature, and L2 cache banks can be interleaved between cores to separate hot components and take advantage of spatial thermal filtering.

In this paper, we present an evaluation of the potential thermal benefits of multi-core floorplanning. We evaluate techniques from the literature, including manipulating core orientation, L2 cache bank insertion, and hierarchical floorplanning, and introduce two new techniques, core mingling and core scattering, in order to bound the potential benefits of temperature-aware floorplanning. For multi-core architectures up to 16 cores, we observe that simply inserting L2 cache banks between identically floorplanned and oriented cores captures 82% of the possible temperature reduction available to multi-core floorplanning. For many-core architectures, L2 cache occupancy continues to have the most significant effect on temperature. This is good news for designers: while in principle, developing different floorplans for each instance of a core may further reduce temperatures, such techniques require substantial floorplanning effort and introduce new validation requirements and are not in fact necessary.

I. Introduction

As transistors scale to tens of nanometers, low-level physical effects which were previously considered second-order and were largely invisible to computer architects—e.g., leakage, temperature, power delivery and parameter variations—have become primary concerns. Of these, temperature has arguably become one of the hardest obstacles to continued technology scaling. The exponential increase in power density across technology generations translates into a corresponding increase in cooling costs. Furthermore, the exponential impact of temperature on leakage power and lifetime reliability, combined with usability considerations like the energy cost of cooling, fan noise, and wearability, have made high temperature a critical design challenge in microprocessors.

As a result, a variety of temperature management techniques have emerged that control temperatures by distributing computation (and therefore power density) in *time*, in *space*, or both. The canonical approach to distributing computation in time is dynamic voltage and frequency scaling (DVFS), the theoretical basis for a multitude of dynamic thermal management (DTM) techniques [1]–[6]. DVFS can be applied directly to keep temperatures below an emergency threshold. Such techniques, unfortunately, are not a panacea: DTM tends to inherently limit performance, and voltage scaling will become more difficult as scaling continues and the supply voltage nears the threshold voltage. In the context of other, performance-enhancing techniques (*e.g.*, sophisticated caching), DVFS can also be applied to make use of schedule slack to reduce power dissipation (and therefore heat generation); however, such an application of DVFS has little place in high-performance computing, when there is almost always another job that needs to be run.

An alternative is to distribute computation in space. This can be accomplished at run-time by distributing computation across multiple cores or threads, in an effort to evenly heat all resources [7]–[10]. However, the performance of such approaches generally suffers due to workload migration.

Computation can also be distributed in space at design-time, with temperature-aware floorplanning. Temperature-aware floorplanning is attractive because of its predictability (which is important for real-time applications that may not be suitable for DVFS), and because it is orthogonal to run-time management schemes. Temperature-aware microarchitectural floorplanning has been extensively studied for single core processors [11]–[15].

Multi-core temperature-aware floorplanning presents a unique set of challenges and opportunities. For example, tiling processors in a many-core design can result in pathological thermal behavior, with hot components in neighboring processors heating each other to dangerous levels. Multi-core floorplanning can address such concerns in a number of ways: by changing the orientation or internal floorplan of adjacent cores [16], or inserting components with lower power density, e.g., L2 caches, between cores [17], [18].

In this paper, we develop new techniques and implement several from the literature in order to explore the extent to which multi-core temperatures can be controlled at design-time. We first consider multi-core architectures up to 16 cores. In this line of experimentation, we begin by establishing the upper temperature bound using the current approach in industry, which

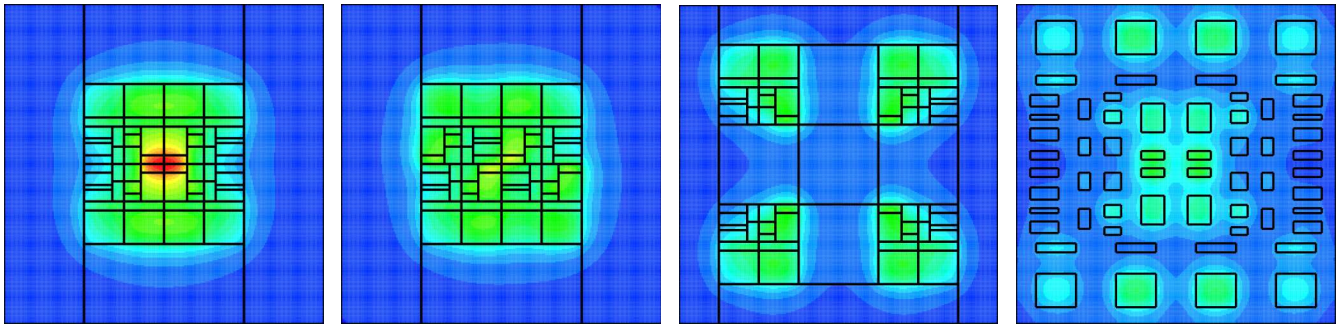


Fig. 1: Heat maps for different core arrangements for a 4-way CMP: (a) a typical floorplan with hot units adjacent to each other; (b) a cooler alternative with different core orientations; (c) a floorplan where L2 cache banks are inserted between cores. (d) an *impractical* floorplan where core components are scattered throughout the die; this floorplan explores the limit of the thermal benefits of floorplanning.

is to tile cores such that there is both vertical and horizontal symmetry. Next, we establish the lower temperature bound using two new (entirely theoretical and wholly impractical) approaches which (a) mingle the components of multiple cores or (b) scatter them throughout the floorplan to minimize temperature, without consideration for what would be considerable performance penalties in each case. We then investigate where other techniques—core orientation, inserting L2 cache banks between cores, and using alternative core floorplans—fit between these extremes. We conclude our experimentation with a limits study examining the thermal benefit of floorplanning in many-core architectures.

Throughout our experiments, we observe that simply inserting L2 cache banks between cores has the most significant effect on system temperature. Considering multi-core architectures, L2 cache insertion between identically floorplanned and oriented cores captures 82% of the possible temperature reduction available to multi-core floorplanning: the extreme (and impractical) floorplanning approaches we introduce provide minimal additional benefit, and are therefore not worth further investigation. This is good news for designers: while in principle, developing different floorplans for each instance of a core may further reduce temperatures, such techniques require substantial floorplanning effort and introduce new, costly validation requirements, and L2 cache bank insertion captures most of the possible thermal benefits of multi-core floorplanning.

II. Temperature-aware Multi-core Floorplanning

There are many levels of design which a multi-core designer can exploit to effectively distribute heat spatially. Homogeneous multi-core floorplanning, which traditionally tiles out cores so that common interfaces to shared resources (e.g., L2 cache) are close to each other, can result in pathological behavior. An example of this is illustrated in Figure 1(a): because the hottest regions of the cores are adjacent to each other, the peak temperature of the system as a whole rises.

To address this problem, new techniques have emerged that attempt to separate, and therefore cool, the hottest components of individual cores. In this paper, we investigate the effect of five such techniques: core orientation manipulation, L2 cache bank insertion, hierarchical floorplanning, core mingling and core scattering.

A. Manipulating Core Orientation

Manipulating core orientation can significantly decrease peak temperature [16]. Theoretically, each core can have eight different orientations: four rotational symmetries and their corresponding four reflections (mirror images). The advantage of manipulating core orientation is that it can often be employed without increasing design costs. For example, standard cell layouts are easily reflected along either the x- or y-axis, limiting the additional design effort required to change core orientation. In fact, current designs already employ core reflection (e.g., in the Niagara 2 [19], Cell [20], and Xeon [21]).

The effect of core orientation is illustrated in Figure 1(b): compared to Figure 1(a), the top-left and bottom-right cores have each been mirrored on their y-axis, and as a result, the cores' hottest components are better spatially distributed. This clearly produces a net reduction in peak temperature.

There are several trade-offs to using core orientation. The first is that cores are traditionally oriented with hot components together for a reason: shorter distances, and therefore faster access, to shared resources. Changing core orientation may therefore negatively affect performance. Furthermore, in the future, opportunity for manipulating core orientation may be limited. In advanced processes where design rules restrict the directions that metal layers may be routed, taking advantage of rotational symmetry may require additional designer effort: in this case, cores may have to be floorplanned for each rotation, multiplicatively increasing the required design validation effort.

B. Inserting Cache Banks

Inserting caches between cores is another way to decrease peak temperature [17], [18]. Silicon has been shown to act as a spatial low-pass filter for temperature [18], [22]: increasing the spatial frequency of the power density distribution reduces system temperature. For tiled multi-core chips, this can be accomplished by interleaving cores (high power density) and banks of memory, *e.g.*, L2 cache (low power density); the larger the banks of L2 separating the cores, the higher the spatial frequency and lower the resulting temperature.

L2 cache banks are particularly convenient for inserting between cores: they are already partitioned into many banks, their power density is quite low because of relatively infrequent accesses, and therefore their temperatures are usually among the lowest on a chip.

The effect of L2 cache bank insertion is illustrated in Figure 1(c). In this case, the L2 cache banks act as buffers between the cores, absorbing heat and reducing system temperature more significantly than changes in core orientation.

C. Hierarchical Floorplanning

Another avenue for temperature reduction is a hierarchical approach: first, floorplan the cores themselves, and then floorplan the system. Total system floorplanning (in which each component of each core is placed simultaneously) is impractical due to the size of a multi-core system. However, recent work has explored deriving different floorplans for different instances of the same core [16]. One obvious trade-off is that, as in the case of rotational symmetries in advanced processes, each individual core plan (a floorplan for a particular core) needs to be validated (for pre- and post-silicon timing, power, lifetime, *etc.*) as if it were an entirely different core. As a result, much of the benefit of tiling cores in a homogeneous system is lost. Another challenge is that by not performing global floorplanning (and instead performing local optimization), global thermal data isn't considered, leading to sub-optimal local decisions.

In this paper, we investigate a subset of hierarchical floorplanning where a core is floorplanned based on its targeted applications (*i.e.,* SPEC2000), and then tiled, oriented, or buffered with L2 caches as described above.

D. Core Mingling and Scattering

We propose for consideration two additional techniques: *core mingling* and *core scattering*. Neither of these technique are expected to be practical from a performance or implementation standpoint. However, these approaches represent the limits of what is possible with core floorplanning and L2 cache bank insertion.

Core mingling applies single-core floorplanning to all of the functional blocks of each core simultaneously, resulting in a medley of functional blocks from different cores surrounded by L2 cache banks. Compared with manipulating core orientation, core mingling represents the potential for temperature reduction if core components can be freely distributed within the boundaries of a single "super-core."

Core scattering dissolves the boundary of this "super-core" and allows L2 cache banks to be inserted between the functional units of cores, resulting in the scattering of the functional blocks throughout the entire chip, as illustrated in Figure 1(d). Core scattering maintains the adjacency of functional blocks from the original core floorplan; this keeps communicating blocks next to one another, and likely results in higher temperatures than would be observed if functional blocks could be placed arbitrarily. Compared with inserting L2 cache banks between cores, core scattering represents the potential for temperature reduction if L2 cache can be inserted between the blocks that make up a core, effectively dissolving the boundaries of the chip; aside from assuming a completely uniform power density distribution, such a floorplan should be among the coolest possible.

III. Experimental Setup

We conducted simulations to evaluate the relative effects of core orientation manipulation, L2 cache bank insertion, and hierarchical floorplanning, and their relevant combinations, in the floorplanning of homogeneous multi-core processors. To establish the effectiveness of these approaches, we also identified and evaluated a worst-case (hot) architecture, and applied our two aggressive (if most likely impractical) techniques for reducing temperature, core mingling and core scattering.

A. Simulation Infrastructure

We use a combination of the HotSpot [5] thermal model, Wattch [23] power model and Simplescalar [24] performance model to perform our thermal simulations. We selected the SPEC2000 [25] benchmark suite for our workload.

We selected a core with a microarchitecture similar to the Alpha 21364 as in [5], but scaled to 45 nm for our baseline system of four cores. As we scale to systems with more cores, we scale both the area of each core and its power consumption such that the power density of each core remains constant. In practice, increasing the number of cores by reducing core complexity may in fact reduce power density. On the other hand, increasing the number of cores through transistor scaling alone is likely to increase power density. To model the thermal behavior of multi-core systems, we assume each core executes the same benchmark at the same time, and therefore simultaneously generates the same power trace.

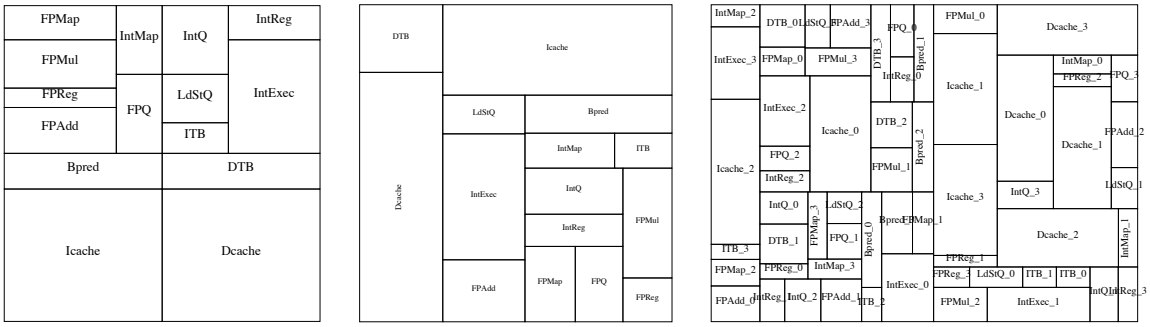


Fig. 2: (a) Original EV6 floorplan, (b) Alternative EV6 floorplan, *altflp*, (c) Floorplan of four *mingled* EV6 cores.

Because our goal is precisely quantifying the thermal effect of floorplanning choices and not exploring the performance trade-offs, we assume a traditional Uniform Cache Access (UCA) L2 cache in order to simplify our simulation framework. The trade-offs in cache bank insertion are complex, and largely dependent on the cache access paradigm. In the case of UCA, interleaving cache banks distributes the cache, potentially increasing access delay. If a Non-Uniform Cache Access (NUCA) paradigm is assumed, *e.g.*, CMP-NuRAPID [26], inserting L2 banks between cores will make some L2 banks closer to some cores, while others will be farther away; this has the potential to actually improve the performance of some fraction of cache accesses. UCA therefore provides a lower bound for the benefits, because it assumes the worst-case delay. Since we model performance and generate power traces from the execution of single-core applications, we do not model the effect of accesses to shared memory when performing thermal simulations for multi-core systems. Furthermore, without significant loss of generality, we do not model L3 cache; in systems such as those we consider, L3 would also likely be distributed, and have similar thermal properties to L2 cache. Detailed analysis of the performance impact of cache bank insertion is beyond the scope of this paper.

In order to model a midrange cooling solution, in HotSpot we assigned convection resistance and TIM thickness values of 0.5 $\frac{K}{W}$ and 30μm respectively. All other thermal model parameters were set to their default values. The default configuration modeled is a 4-core processor with 75% of the die-area occupied by L2 cache.

Our experiments proceed as follows. First, we simulate each benchmark in SPEC2000 using SimpleScalar. We employ Wattch, HotSpot and SimPoint to find the power trace for the most representative 500 million instruction interval; during this simulation run, we use HotSpot's temperature-dependent leakage model to dynamically re-calculate leakage power as temperature changes [5].

The individual power traces collected from each benchmark are then aggregated to derive a single set of average power values for each functional unit in the core. This new average power trace (derived for a single core) is subsequently used to drive thermal optimization (for the entire multi-core system) in HotFloorplan [11] using one or more of the techniques presented in Section II. In this case, HotFloorplan employs HotSpot's block-based model, in the interest of minimizing computational complexity at the cost of some accuracy. This results in a set of thermally fit candidate floorplans.

We then perform a detailed evaluation of the floorplans identified by HotFloorplan. Using the original power traces gathered for each benchmark, we perform simulation again using the candidate floorplans. In this case, HotSpot employs its more accurate grid-based mode. We use a grid resolution of 256 x 256 and perform steady state thermal simulations to obtain the peak temperature of the system. The peak temperature values we obtain here are conservative. Once the peak steady state temperature of a benchmark for a given floorplan is computed, power densities are not re-evaluated to take into account the reduced leakage due to the temperature reduction achieved with floorplan optimization. As a result, the temperature improvement we report is a lower bound, and in practice the improvement is expected to be larger due to the reduction in leakage.

B. Core Orientation

We first extended HotFloorplan to manipulate core orientation. A core can have any of eight different orientations: four rotational symmetries and their corresponding four reflections. Hence, for a k-way CMP, there are a total of 8^k different possible floorplans.

For a small number of cores (*e.g.* $k \leq 6$), the floorplan with the lowest peak temperature is identified with an exhaustive search. Once the an exhaustive search is intractable, we employ simulated annealing [27] to search through the vast solution space. Each floorplan is encoded as a k-digit octal number indicating how each core is oriented. The system is permuted by incrementing (modulo 8) a random digit in the k-digit string, thereby selecting the next numerical orientation.

C. L2 Cache Insertion

We next extended HotFloorplan to insert L2 cache banks between cores. HotFloorplan arranges L2 cache banks to tile the entire silicon real estate in a checkerboard-like fashion to increase the lateral spreading of heat. Banks of L2 cache are

intentionally wrapped around peripheral cores; since the chip boundaries allow lateral heat conduction only in two or three directions (instead of four), keeping cores away from chip boundaries facilitates heat spreading. To determine the positions of the cores and the cache banks, we assume (heuristically) that adjacent cores are equidistant from each other and that the distance between a peripheral core and chip boundary is half the distance between adjacent cores.

We assume that the size and aspect ratio of cache banks are flexible; we do not model the resulting effects on system performance and power dissipation. In practice, changing the number of rows and columns in an SRAM array, while common practice in design, does have an effect on the performance and power dissipation of the array [28]. Carefully modeling this effect is the subject of future work.

Under our assumption of an UCA cache architecture, separating cache banks with cores isn't expected to significantly affect cache performance: cache access delay in UCA by definition is a function of the distance to the farthest cache bank. Because the L1 cache filters most accesses to L2, for the range of L2-area to chip-area ratios we consider (25-85%) we determined that the extra distance any core may move from the farthest bank translates to less than a 2% slowdown for SPEC2000 benchmarks [11], [14].

In real systems, it is likely that some form of distributed Non-uniform Cache Access (NUCA) would be employed, such as CMP-NuRAPID [26]. Such systems would be expected to tolerate interleaving L2 cache banks with cores; in fact, performance may increase in this case, since some L2 banks would be closer to some cores than when cores occupy the center of the chip.

D. Hierarchical Floorplanning

We next extended HotFloorplan to perform hierarchical floorplanning. Given a set of functional blocks and their area and aspect ratio constraints, HotFloorplan first floorplans the core using the classic Wong and Liu [29] simulated annealing algorithm with a cost function that includes area, aspect ratio, delay and temperature. Delay and temperature are balanced by carefully setting the relative importance of architectural wires [14]. L2 cache banks are then inserted between the cores as described in Section III-C. As a final step, the orientation space of the cores is searched using simulated annealing as described in Section III-B.

The original EV6 floorplan is illustrated in Figure 2(a); our temperature-aware floorplanning approach produced the core plan illustrated in Figure 2(b), denoted *altflp*. *altflp* has less than 0.05% dead space, an aspect ratio close to 1, and a better wire length metric when compared to the original floorplan.

Note that the alternative floorplan is computed in isolation, without consideration of the core's location within a multicore chip. As a result, hot units are steered away from the boundaries as much as possible to minimize the peak temperature. As we will observe in the following section, this is not necessarily beneficial in a multicore environment—especially with L2 cache bank insertion—because units near the boundaries of a core are closer to the L2 cache, which is relatively cooler.

E. Core Mingling and Scattering

Finally, we extended HotFloorplan to explore core mingling and scattering. Core mingling is achieved using the same temperature-aware floorplanning described in Section III-D, except that components from all four cores are placed at the same time. We assume that all blocks have flexible, if constrained, aspect ratios. The resulting floorplan is illustrated in Figure 2(c). Dead space contributes less than 0.1% of the total area.

Core scattering is achieved by taking an input floorplan for all cores, like that in Figure 1(a) and pulling each component toward the edges of the chip. L2 cache is abstracted into a sea of memory, and components are placed arbitrarily within it. Component adjacency is preserved; components that were next to each other in the original floorplan will be separated only by L2 in the scattered floorplan. Hot components are given a wider margin of L2 than cooler components, as illustrated in Figure 1(d).

IV. Results

We conducted a variety of floorplanning experiments to evaluate each technique in Section III as well as several combinations. We first assume the floorplan in Figure 1(a) as the worst-case floorplan, denoted *hot*. The cores are oriented in such a manner that their hottest units, the integer register files, are touching each other, resulting in the highest observed peak temperature in our experiments.

For the baseline, denoted *base*, we select a floorplan similar to *hot*, but in which all cores have the same orientation, as illustrated in Figure 2(a). We performed L2 cache bank insertion on *base*, producing *base+l2*.

We next found floorplans using core orientation manipulation. For four cores, we performed an exhaustive search for the floorplan with the lowest peak temperature. The result is denoted *orient*. We performed L2 cache bank insertion on *orient* as well, producing *orient+l2*.

We next performed hierarchical floorplanning, generating the core plan illustrated in Figure 2(b), and using it in several derivative system-level floorplans. *altflp* denotes a simple system-level floorplan, in which four cores are surrounded by L2, as in *base*. *altflp+orient* is produced by permuting the orientation of each core; *altflp+l2* is produced using L2 cache bank insertion; and, *altflp+orient+l2* by combining the two techniques.

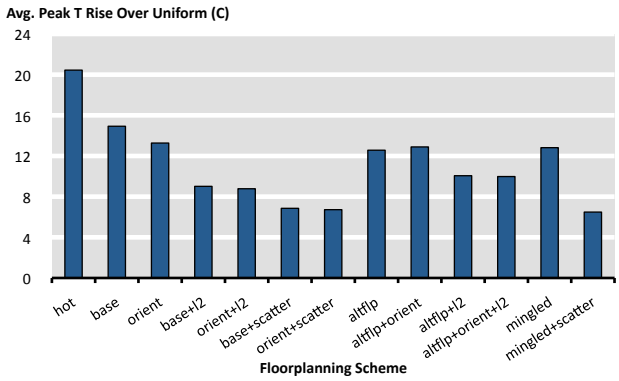


Fig. 3: The peak temperature rise of the different floorplan configurations averaged across all the SPEC2000 benchmarks.

Finally, we examine the potential limits of temperature reduction through multi-core floorplanning by appling core mingling and core scattering. We applied core scattering to *base* and *orient*, producing *base+scatter* and *orient+scatter*. We subsequently produced *mingled* by surrounding the core plan in Figure 2(c) with L2 cache, and further applied scattering to produce *mingled+scatter*.

Figure 3 illustrates the results of our evaluation. Given the total power consumption on a die, the absolute maximum thermal benefit achievable with floorplanning is bounded by the case where power is dissipated uniformly across the entire die. Figure 3 plots the difference between the peak tempearture of the floorplanning schemes and the peak temperature of such a uniform power distribution, averaged across all benchmarks.

As expected, *hot* has the highest average peak temperature rise over that of a uniform power density distribution. Exploiting the orientation of the cores is beneficial when the cores are adjacent to one another, as demonstrated by *base* and *orient*. *base* reduces the temperature rise by 27% relative to *hot*; *orient* reduces the temperature rise by 35%. The insertion of L2 cache banks between the cores reduces peak temperatures even more significantly. *base+l2* reduces the temperature rise by 11.4°C, or 56%.

base+scatter, *orient+scatter* and *mingled+scatter* illustrate the thermal spreading potential available in multicore floorplanning, and reduce the temperature rise relative to *hot* by up to 68%. Comparing *base+l2* and *mingled+scatter*, we can see that L2 cache insertion alone is able to capture 82% this potential reduction relative to *hot*; while mingling cores and scattering components does reduce system temperature, the benefit is not significant enough to justify exploring more practical versions of such extreme floorplanning techniques.

Although the alternative floorplan *altflp* and the mingled floorplan *mingled* are able to achieve significant temperature reduction, much of that reduction can be achieved by a simple orientation space search (*orient*). Furthermore, since *altflp* has the hot functional blocks towards its center to assist internal lateral heat transfer, L2 cache bank insertion provides less benefit than for *base* or *orient*.

A. Sensitivity Studies

We conducted a set of sensitivity studies in order to investigate how our previous observations are affected by our assumptions about core size, core occupancy and L2 power density respectively.

1) Effect of Core Size: For the practical approaches previously described (*i.e.*, excluding *mingled* and *scattered* variants), Figure 4 plots the effect of varying the number of cores in a chip. As the number of cores increases, the size and power of each core is scaled so that their power densities remain constant. Each series decreases due to the fact that silicon acts as a spatial low-pass filter for temperature [18], [22]: for the same power density, smaller cores (high frequency) are cooler than larger cores (low frequency). We observe that the trends observed in the previous section still hold: much of the thermal benefit comes from L2 cache insertion. The only difference is that *altflp* performs worse than *orient* for higher number of cores.

2) Effect of Core Occupancy: For the practical approaches previously described, Figure 5 plots the result of an experiment varying the core occupancy—the ratio of core area to the total area of the chip—from 85% to 25%. In this experiment, we hold core and L2 power density constant. Using a system with four cores, we decrease core occupancy by adding L2, thereby increasing the size of the chip. This increases the total power dissipated in the system. However, as core occupancy decreases, the amount of L2 available to act as a thermal buffer also increases. For this reason, even though total power dissipated in the system increases with decreasing occupancy, the peak temperature decreases.

We also observe that as the occupancy decreases, the importance of core orientation increases; out of all techniques, *orient* is the most sensitive to core occupancy. At 25% core occupancy, *orient* even performs marginally better than the L2 cache insertion techniques.

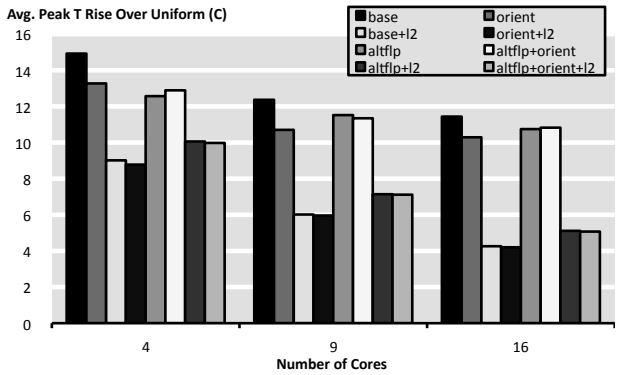


Fig. 4: As the number of cores increases, L2 bank insertion remains the dominant factor in temperature reduction.

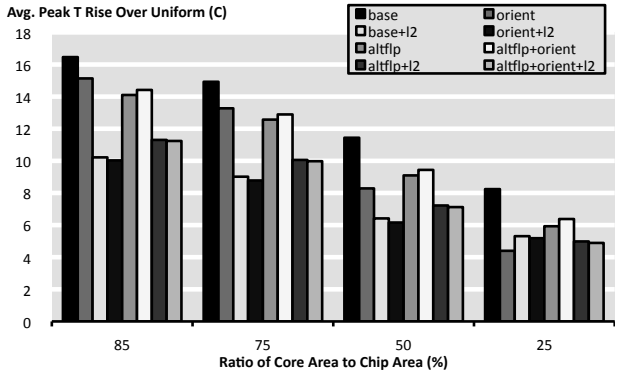


Fig. 5: As core occupancy increases, the relative reduction attributable to L2 cache bank insertion decreases.

3) Effect of L2 Power Density: We also considered the effect of L2 cache bank power density, since the efficacy of several of previously described depends on the capacity of L2 to facilitate lateral heat transfer. Figure 6 plots the average peak temperature for each floorplanning scheme. When we double the power density of L2 caches, we observe that the trends remain the same, with peak temperatures increasing by less than two degrees on average.

V. Scaling to Many-Core Processors

When systems contain 10s or 100s of cores, it is impractical to optimize individual core plans for thermal purposes: verifying different core plans is prohibitive as the number of different core plans grows. Furthermore, existing research suggests that the benefits may be limited: a phenomena called *spatial thermal filtering* indicates that as the size of power dissipators shrink relative to the size of the system, silicon increasingly acts as a low-pass filter, dampening the thermal impact of the dissipators [18], [22], [30], [31]. A consequence of spatial filtering is that, when considering many-core systems where the size of individual components is relatively small compared to the system, local changes in power dissipation are likely to be filtered (and therefore have little or no effect). Systemic changes in global power dissipation, such as system-level organization (relative core and L2 cache placement) or DTM (power down whole cores or groups of cores) may be the only way to affect the thermal profile of the system.

We have conducted additional experiments to explore the thermal benefits of floorplanning as the number of cores in the system increases as a result of manufacturing process scaling. We have observed above that inserting L2 between cores significantly reduces system temperatures. Our experimentation therefore investigates how different ways of inserting L2 between cores affects the temperature of tiled many-core architectures.

A. Tiled Many-core Floorplans

We evaluated the temperature of systems with a growing number of cores in architectures where cores and L2 are tiled in various ways, as illustrated in Figure 7. Darkly shaded blocks are processors while the lightly shaded blocks are caches. The different layouts tile cores and L2 in different ways, changing the distance between power dissipators (cores) by manipulating the placement and orientation of blocks of L2 cache. In the figure, "yyyy-xx" indicates floorplan template "yyyy," *e.g.* regular, with "xx" % core occupancy.

We select *regular-50* as our baseline, as it represents a typical assumption in many architectural studies. This organization assumes square cores and square blocks of L2, tiled in a checkerboard fashion.

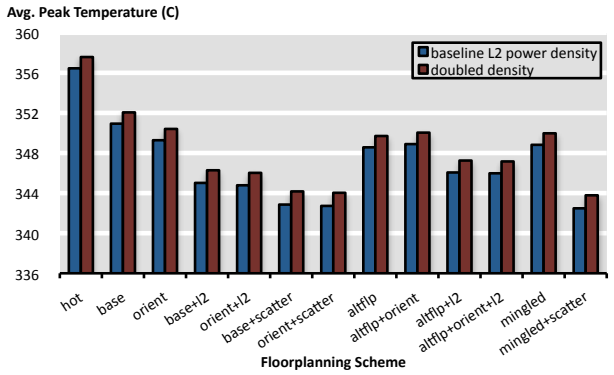
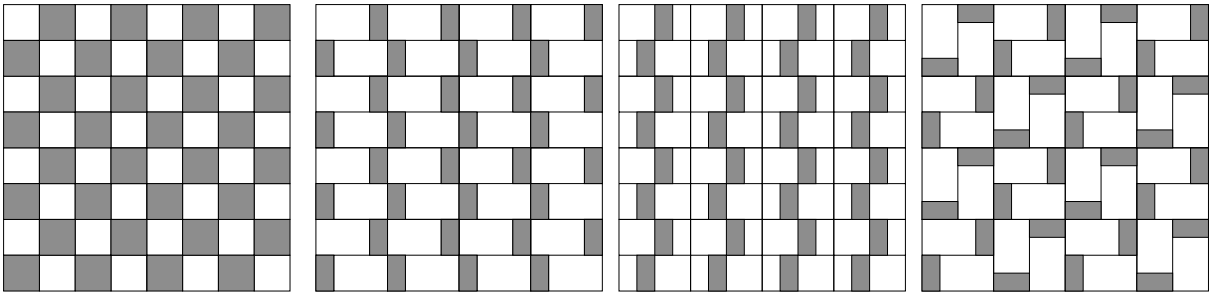


Fig. 6: When L2 cache power density is doubled, average peak temperature rises only modestly.



(a) regular-50 (b) regular-25 (c) alt-25 (d) rotated-25

Fig. 7: Illustration of the several checkerboard configurations studied. Darkly shaded areas are cores and the unshaded areas are lower level cache banks. "-xx" indicates the percentage of die area occupied by the cores. Each figure above shows a many-core die with 32 cores.

As core occupancy increases or decreases, alternative organizations of cores and L2 are possible. At 25% occupancy, *regular-25* simply replaces half of each core with additional L2. *alt-25* moves cores away from the edges of the die, all the better to take advantage of the thermal capacitance of more neighboring L2 blocks. *rotated-25* changes the orientation of some cores in an effort to increase the average distance between cores.

We also examine the dual of the 25% occupancy cases, when core occupancy is 75%. *regular-75* is the complement of *regular-25*. Likewise, *rotated-75* is the complement of *rotated-25*. We do not examine an *alt-75* configuration, as this would require dividing some cores and placing pieces of them on either side of a cache bank.

B. Experimental Setup

To evaluate the floorplanning templates in Figure 7, we performed steady state analysis using the ANSYS 11.0 Finite Element Method (FEM) solver. We assume the silicon die is 16 mm x 16 mm x 0.15 mm. Since the thickness of the die does not scale with the feature size for reasons of mechanical strength, it is assumed to be constant. Each system also includes (1) a layer of thermal interface material (TIM) $20\mu m$ thick and of the same lateral dimensions as the die, (2) a 3 cm x 3 cm x 1 mm copper heat spreader, and (3) a 6 cm x 6 cm x 6.9 mm copper heat sink. The heat sink is cooled by convection; we assume a co-efficient of heat transfer equivalent to a 0.1 $\frac{K}{W}$ thermal resistance. The thermal conductivities for each layer are 100 $\frac{W}{mK}$ for silicon, 4 $\frac{W}{mK}$ for the TIM, 400 $\frac{W}{mK}$ for the heat spreader and 400 $\frac{W}{mK}$ for the heat sink.

We model the system using a lateral grid size of 48 x 48. The lateral dimensions of the grid cells in the other layers are the same as in silicon. Vertically, the silicon die is divided into three layers, spreader and sink are divided into four layers each and the TIM is modeled as a single layer.

We assume that the power density of the cores is uniform, and set to 1 $\frac{W}{mm^2}$, while that of the L2 cache banks is set to be 0.1 $\frac{W}{mm^2}$. We assume that as the number of cores increases, the power density remains constant. In practice, increasing the number of cores by reducing core complexity may in fact reduce power density. On the other hand, increasing the number of cores through transistor scaling alone is likely to increase power density.

We also explore the case of infinite cores as a limit study. When the number of cores is infinity, the power density on the entire die becomes uniform with a value equal to the average of the power density of the cores and that of the cache banks, weighted by their respective area occupancies.

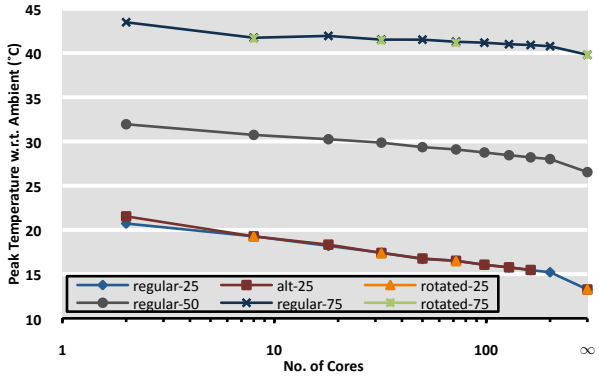


Fig. 8: Occupancy clearly has the greatest effect on peak temperature. Lower occupancy, however, does yield greater temperature reductions with increasing core count. Core orientation has the least significant effect.

C. Results

We first investigated the extent to which scaling by itself improves the thermal characteristics of silicon systems. Figure 8 plots the difference between the peak temperature on the die and the ambient temperature for three different occupancy levels (25%, 50%, and 75%), and each possible template from Figure 7 in each case. Both the axes are logarithmic (except for the infinity point on the x-axis). Consider the base case, *regular-50*. Moving from 2 to 16 cores reduces the peak temperature 5% in this case; in the limit, peak temperature can be reduced by 17% by spreading power dissipation (and thus heat generation) across the whole chip. As the number of cores increases, the spatial filtering effect contributes to cooling when total power dissipation is held constant.

We next conducted experiments to explore the effect of spatial filtering on systems with different core occupancy, since different systems may have different ratios of area devoted to core and cache. Just as we observed above, we see in Figure 8 that as the number of cores scales, die occupancy influences the peak temperature more strongly than core orientation or the number of cores alone. On average, increasing occupancy from 50% to 75% increases peak temperature by 43% under the *regular* floorplan. Likewise, decreasing occupancy from 50% to 25% decreases peak temperature by 42% under the *regular* floorplan. In comparison, recall that at most, increasing core count reduces peak temperature by 17% when half the die is cores.

Occupancy also influences the extent to which core count can reduce peak temperature. In this experiment, the die is kept constant as occupancy increases. Occupancy is increased by increasing the area used by cores, while occupancy is decreased by decreasing the area used by cores (in Figure 7, compare *regular-25*, *regular-50*, and the dual of *regular-25*, *regular-75*).

When core occupancy increases, power dissipation and therefore peak temperature also increase (since cores dissipate more power than L2 cache). Moreover, dies are separated by less L2 when occupancy is higher, reducing the effect of spatial filtering. In this case, scaling from 2 to 16 cores reduces peak temperatures by 4%; in the limit, the peak temperature is reduced just 8%.

On the other hand, when the die occupancy is lower, the effect of spatial filtering is greater: scaling from 2 to 16 cores reduces peak temperatures by 12%, and the peak temperature is reduced 36% in the limit.

The different ways of orienting cores, however, plays only a marginal role as technology scaling produces systems with more cores. When occupancy is fixed at either 25 or 75%, peak temperature changes on average less than 1% when the floorplan is changed. Clearly, placing cores in the corner of the chip does not matter in the presence of a heat spreader and sink, and rotation provides no significant benefit.

VI. Conclusion

As the importance of thermal management increases with transistor scaling and rising cooling costs, it is imperative that designers consider all available options for reducing system temperature. In this paper, we have investigated the potential for temperature reduction when performing temperature-aware multi-core floorplanning. Orthogonal to dynamic thermal management, multi-core floorplanning presents important opportunities to reduce system temperature, and in some cases with minimal designer effort.

We evaluated a variety of new and existing floorplanning techniques for reducing temperature, including manipulating core orientation, inserting L2 cache banks between cores, hierarchical floorplanning, as well as core mingling and core scattering, and relevant combinations. Many of these techniques have significant implications for designer effort and system performance. However, we observed that leveraging spatial thermal filtering by inserting L2 cache banks between unmodified cores captures the overwhelming majority of the opportunity for temperature reduction. Compared to the hottest floorplan, L2 cache bank insertion reduced the temperature rise over that resulting from uniform power density by 68%, achieving 82% of the reduction

possible when core components are placed individually in a sea of L2 cache. While the extreme floorplanning techniques we introduced further reduce temperatures, the benefits are minimal, while the costs are significant; such techniques therefore do not warrant further investigation. We observe that the same phenomena dominates in many-core architectures: changing the amount of L2 cache has far and away the most significant effect on temperature.

While we observed that developing different floorplans for each instance of a core reduces temperatures, such techniques require substantial floorplanning effort and multiplicatively increase validation requirements. Instead, design-time thermal management can be largely achieved with careful L2 cache bank placement, a natural fit for emerging distributed L2 cache architectures.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant CRI-0551630, and research gifts from Intel and NVIDIA.

REFERENCES

[1] S. Gunther, *et al.*, "Managing the impact of increasing microprocessor power consumption," in *Intel Technology Journal*, Q1 2001.
[2] W. Huang, *et al.*, "A framework for dynamic energy efficiency and temperature management," in *MICRO-33*, Dec. 2000.
[3] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *HPCA-7*, Jan. 2001.
[4] C.-H. Lim, W. Daasch, and G. Cai, "A thermal-aware superscalar microprocessor," in *ISQED'02*, Mar. 2002.
[5] K. Skadron, *et al.*, "Temperature-aware microarchitecture," in *ISCA-30*, April 2003.
[6] S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," in *ISLPED'03*, Aug. 2003.
[7] E. Rohou and M. Smith, "Dynamically managing processor temperature and power," in *FDDO-2*, Nov. 1999.
[8] M. D. Powell, *et al.*, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system," in *ASPLOS-XI*, Oct. 2004.
[9] P. Chaparro, J. González, and A. González, "Thermal-aware clustered microarchitectures," in *ICCD'04*, Oct. 2004.
[10] J. Choi, *et al.*, "Thermal-aware task scheduling at the system software level," in *ISLPED'07*, 2007.
[11] K. Sankaranarayanan, *et al.*, "A case for thermal-aware floorplanning at the microarchitectural level," *J. Instr.-Level Parallelism*, vol. 7, 2005.
[12] Y. Han, *et al.*, "Temperature aware floorplanning," in *TACS-2*, June 2005.
[13] Y-W. Wu, *et al.*, "Joint exploration of architectural and physical design spaces with thermal consideration," in *ISLPED'05*, August 2005.
[14] A. Chakravorty, *et al.*, "Re-visiting the performance impact of microarchitectural floorplanning," in *TACS-3*, June 2006.
[15] V. Nookala, *et al.*, "Temperature-aware floorplanning of microarchitecture blocks with IPC-power dependence modeling and transient analysis," in *ISLPED'06*, 2006.
[16] M. B. Healy, *et al.*, "Thermal optimization in multi-granularity multi-core floorplanning," in *ASP-DAC'09*, 2009.
[17] J. Donald and M. Martonosi, "Temperature-aware design issues for SMT and CMP architectures," in *WCED'04*, Jun. 2004.
[18] W. Huang, *et al.*, "Many-core design from a thermal perspective," in *DAC'08*, June 2008.
[19] U. G. Nawathe, M. Hassan, L. Warriner, K. Yen, B. Upputuri, D. Greenhill, A. Kumar, and H. Park, "An 8-core 64-thread 64b power-efficient SPARC SoC," in *the Proceedings of the 2007 International Solid-State Circuits Conference, ISSCC'07*, February 2007.
[20] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, "Introduction to the Cell multiprocessor," *IBM Journal of Research and Development*, vol. 49, 2005.
[21] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Kottapalli, "A 45nm 8-Core Enterprise Xeon©Processor," in *the Proceedings of the 2009 International Solid-State Circuits Conference, ISSCC'09*, February 2009.
[22] K. Etessam-Yazdani, *et al.*, "Investigation of the impact of power granularity on chip thermal modeling using white noise analasis," *IEEE Trans. Compon., Packag., Technol.*, vol. 31, no. 1, March 2008.
[23] D. Brooks, *et al.*, "Wattch: A framework for architectural-level power analysis and optimizations," in *ISCA-27*, June 2000.
[24] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," *IEEE Computer*, vol. 35, no. 4, February 2002.
[25] Standard Performance Evaluation Corporation, "SPEC CPU2000 Benchmarks," http://www.specbench.org/osg/cpu2000.
[26] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Optimizing replication, communication, and capacity allocation in cmps," in *ISCA'05*, 2005.
[27] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.
[28] S. Nalam, M. Bhargava, K. Mai, and B. H. Calhoun, "Virtual prototyper (ViPro): An early design space exploration and optimization tool for SRAM designers," in *the Proceedings of the 2010 Design Automation Conference, DAC'10*, June 2010.
[29] D. F. Wong and D. L. Liu, "A new algorithm for floorplan design," in *DAC'86*, Jun. 1986, pp. 101–107.
[30] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, and S. Ghosh, "Hotspot: A compact thermal modeling method for CMOS VLSI systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, May 2006.
[31] W. Huang, K. Sankaranarayanan, K. Skadron, R. J. Ribando, and M. R. Stan, "Accurate, pre-rtl temperature-aware design using a parameterized, geometric thermal model," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1277–1288, September 2008.