A Case for Thermal-Aware Floorplanning at the Microarchitectural Level

Karthik Sankaranarayanan, *Sivakumar Velusamy and Kevin Skadron
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
E-mail: {karthick,siva,skadron}@cs.virginia.edu

Preliminary draft, to appear, Journal of Instruction Level Parallelism, 2005.
Please do not distribute.

Abstract

In current day microprocessors, exponentially increasing power densities, leakage, cooling costs, and reliability concerns have resulted in temperature becoming a first class design constraint like performance and power. Hence, virtually every high performance microprocessor uses a combination of an elaborate thermal package and some form of Dynamic Thermal Management (DTM) scheme that adaptively controls its temperature. While DTM schemes exploit the important variable of power density to control temperature, this paper attempts to show that there is a significant peak temperature reduction potential in managing lateral heat spreading through floorplanning. It argues that this potential warrants consideration of the temperature-performance trade-off early in the design stage at the microarchitectural level using floorplanning. As a demonstration, it uses previously proposed wire delay model and floorplanning algorithm based on simulated annealing to present a profile-driven, thermal-aware floorplanning scheme that significantly reduces peak processor temperature with minimal performance impact that is quite competitive with DTM.

1. Introduction

As process technology scales into the nanometer region, the exponential increase of power densities across process generations results in higher die temperatures and even higher temperatures in the wires of today's microprocessor chips. The exponential dependence of leakage on temperature aggravates this problem even further. Such high temperatures, when left unmanaged, could potentially affect the processor's correctness of operation. They could also result in its accelerated aging and reduce its operating speed and lifetime. In microprocessors, this has invariably resulted in some form of cooling solutions. Traditional ones among them have been designed for the worst-case power dissipation and have focused mainly on the thermal package (heat sink, fan etc.). However, more recent solutions involve managing the application's behaviour adaptively in response to the on-chip temperature. These run-time feedback driven mechanisms are called *Dynamic Thermal Management (DTM)* schemes. They slow down the execution of the microprocessor in response to the temperature sensed, resulting in the reduction of the power dissipated and hence in the reduction of the on-chip temperature.

*. Sivakumar Velusamy is currently with Xilinx, Inc., California. A part of this work was conducted when he was a graduate student in UVA.

Since most DTM schemes involve stopping the processor clock or reducing its supply voltage, they have certain implications for a high-performance microprocesor. Firstly, in multi-processor server-based systems, this results in problems with clock synchronization and accurate timekeeping. Secondly, high performance, power-hungry, hot applications causing the DTM to be enabled are slowed down. This impacts systems offering real time guarantees negatively as the slowdowns caused are unpredictable and could potentially lead to failures in meeting the computational deadlines. DTM schemes are designed as solutions to deal with the worst-case applications where the thermal package deals with the average case. However, as processors become hotter across technology generations, this average-case application behaviour itself tends to grow hotter causing reliability lapses and higher leakage. Hence, static microarchitectural techniques for managing temperature can complement what DTM is trying to achieve.

Orthogonal to the power density of the functional blocks, another important factor that affects the temperature distribution of a chip is the lateral spreading of heat in silicon. This depends on the functional unit adjacency determined by the floorplan of the microprocessor. Traditionally, floorplanning has been dealt with at a level closer to circuits than to microarchitecture. One of the reasons for this is the level of detailed information floorplanning depends on, which is only available at the circuit level. However, with wire delays dominating logic delays and temperature becoming a first class design constraint, floorplanning has started to be looked at even at the microarchitecture level. In this work, we investigate the question of whether floorplanning at the microarchitectural level can be applied viably towards thermal management. The question and the associated trade-off between performance and temperature are examined at a fairly higher level of abstraction. In spite of using models that are not necessarily very detailed, this paper hopes to at least point out the potential of microarchitectural floorplanning in reducing peak processor temperature and the possibility of its complementing DTM schemes. It should be noted that floorplanning does not reduce the average temperature of the entire chip very much. It just evens out the temperatures of the functional units through better spreading. Therefore, the hottest units become cooler while the temperature of a few of the colder blocks increases accordingly.

Contributions This paper specifically makes the following contributions:

1. It presents a microarchitecture level thermal-aware floorplanning tool, HotFloorplan, that extends the classic simulated annealing algorithm for slicing floorplans [1], to account for temperature in its cost function using HotSpot [2]—a fast and accurate model for processor temperature at the microarchitecture level. HotFloorplan will be released along with the next version of HotSpot and can be downloaded from the HotSpot download site. The URL is: http://lava.cs.virginia.edu/HotSpot.

2. It makes a case for managing the trade-off between performance and temperature at the microarchitectural level. It does so by employing a profile-driven approach of evaluating temperature and performance respectively by using previously proposed thermal [2] and wire delay [3, 4, 5] models.

3. It finds that thermal-aware floorplanning reduces the hottest temperatures on the chip by a significant amount (about 20 degrees on the average and up to 35 degrees) with minimal performance loss. In fact, floorplanning is so effective that it eliminates all the thermal emergencies (the periods of thermal stress where temperature rises above a safety threshold) in the applications without the engagement of DTM.

The remainder of this paper is organized as follows: Section 2 discusses the previous work in the area closely related to this paper. Section 3 investigates the cooling potential of lateral spreading and presents it as the motivation for this work. Section 4 describes the thermal-aware floorplanning algorithm, the microarchitectural performance model used to study the delay impact of floorplanning and the simulation setup used in the evaluation of this work. Section 5 presents the findings of our research. Section 6 concludes the paper and discusses possible future work.

2. Related Work

Previous work related to this paper falls into three broad categories—first is the wealth of classical algorithms available for floorplanning, second is the addressing of floorplanning at the architecture level for performance and the third is floorplanning for even chip-wide thermal distribution.

Since the research in classical floorplanning is vast and it is impossible to provide an exhaustive overview of the contributions in the field, we only mention a very small sample of the work related to our thermal-aware floorplanning algorithm. A more thorough listing can be found in VLSI CAD texts like [6, 7]. Many floorplan-related problems for general floorplans have been shown to be intractable. Even when the modules are rectangular, the general floorplanning problem is shown to be NP-complete [8, 9]. Hence, 'sliceable floorplans' or floorplans that can be obtained by recursively sub-dividing the chip into two rectangles, are most popular. Most problems related to them have exact solutions without resorting to heuristics. While more general and complex floorplan algorithms are available, this paper restricts itself to sliceable floorplans because of their simplicity. For our work which is at the architectural level, since the number of functional blocks is quite small, sliceable floorplans are almost as good as other complex floorplans. The most widely used technique in handling sliceable floorplans is Wong et al.'s simulated annealing [1]. It is easy to implement, is versatile in handling any arbitrary objective function and has been implemented in commercial design automation tools.

In the area of floorplanning at the architectural level for performance, Ekpanyapong et al.'s work on profile-guided floorplanning [10] and Reinman et al.'s MEVA [11], are works that we are aware of. Profile-guided floorplanning uses microarchitectural profile information about the communication patterns between the functional blocks of a microprocessor to optimize the floorplan for better performance. MEVA evaluates various user-specified microarchitectural alternatives on the basis of their IPC vs. cycle time trade-off and performs floorplanning to optimize the performance. In spite of dealing with architectural issues, it does so at a level close the circuit by specifying architectural template in structural verilog and architectural alternatives in a Synopsis-like '.lib' format. Both of these do not deal with temperature.

Thermal placement for standard cell ASIC designs is also a well researched area in the VLSI CAD community. [12, 13] is a sample of the work from that area. Hung et al.'s work on thermal-aware placement using genetic algorithms [14] and Ekpanyapong et al.'s work on microarchitecural floorplanning for 3-D chips [15] are also close to the area of our work.

Parallel to our work [16], recently, Han et al. also published research on thermal-aware floorplanning at the microarchitectural level [17]. While their work also deals with the same topic as ours, it does not consider performance directly and uses the total wire-length of a chip as the proxy for performance. As we show in our results, this proxy could be misleading and could potentially result in a floorplan with shorter wire-length but worse performance. Also, as [17] does not compare the performance impact of floorplanning with that of DTM, it does not answer the crucial question

of why it is necessary to address temperature at the floorplanning stage even in the presence of DTM. Furthermore, this work is different in that it built the floorplanner used for thermal-aware floorplanning while [17] uses the previously published Parquet floorplanner [18].

Apart from the above-mentioned research, we would also like to mention the wire delay model and parameters from Brayton et al. [5] and Banerjee et al. [4] and the wire capacitance values from Burger et al [3]'s work exploring the effect of technology scaling on the access times of microarchitectural structures. We use these models and parameters in the evaluation of our floorplanning algorithm for calculating the wire delay between functional blocks.

3. Potential in Lateral Spreading

Before the description of the thermal-aware floorplanner, it is important to perform a potential study that gives an idea about the gains one can expect due to floorplanning. Since the cooling due to floorplanning arises due to lateral spreading of heat, we study the maximum level of lateral heat spreading possible. This is done using the HotSpot thermal model which models heat transfer through an equivalent circuit made of thermal resistances and capacitances corresponding to the package characteristics and to the functional blocks of the floorplan. In the terminology of the thermal model, maximum heat spreading occurs when all the lateral thermal resistances of the floorplan are shorted. This is equivalent to averaging out the power densities of the individual functional blocks. That is, instead of the default floorplan and non-uniform power densities, we use a floorplan with a single functional block that equals the size of the entire chip and has a uniform power density equal to the average power density of the default case. On the other extreme, we also make the thermal resistances corresponding to the lateral heat spreading to be equal to infinity. This gives us an idea of the extent of temperature rise possible just due to the insulation of lateral heat flow. The table below presents the results of the study for a subset of SPEC2000 benchmarks [19]. The 'Min' and 'Max' columns correspond to the case when the lateral thermal resistances are zero and infinity respectively, while the 'Norm' column shows the peak steady-state temperature of the chip when the thermal resistances have the normal correct values.

Table 1: Peak steady-state temperature for different levels of lateral heat spreading (^{o}C)

Bench	Min	Norm	Max
bzip2	56	123	222
gcc	55	120	220
crafty	54	120	217
gzip	54	120	215
perlbmk	54	114	201
mesa	54	114	203
eon	54	113	201
art	55	109	188
facerec	52	104	183
twolf	51	98	168
mgrid	47	75	126
swim	44	59	84

Clearly, lateral heat spreading has a large impact on processor temperature. Though the ideal spreading forms an upper bound on the amount of achievable thermal gain, realistic spreading due to floorplanning might only have a much lesser impact. This is so because the functional blocks of a processor have a sizeable, finite area and cannot be broken down into arbitrarily small sub-blocks that can be moved around independently. Hence the maximum attainable thermal gain is constrained by the functional unit granularity of the floorplan. In spite of the impracticality of implementation, this experiment gauges the potential available to be tapped. Conversely, if this experiment indicated very little impact on temperature, then the rest of our paper would be obviated.

4. Methodology

4.1 HotFloorplan Scheme

The broad approach we take in this work is to use the classic simulated annealing based floorplanning algorithm [1]. The only difference is that the cost function here involves peak steady-state temperature, which comes from a previously proposed microarchitectural thermal model, HotSpot [2]. Just like [1], HotFloorplan uses Normalized Polish Expressions (NPE) to represent the solution space of sliceable floorplans and uses three different types of random perturbance *moves* to navigate through them. The aspect ratio constraints for each functional block are represented as piecewise linear shape curves. For each slicing structure corresponding to an NPE, the minimum-area sizing of the individual blocks is done by a bottom-up, polynomial-time addition of the shape curves at each level of the slicing tree [7]. Once the sizing is done, the placement is then passed onto HotSpot for steady-state temperature calculations. It uses the profile-generated power dissipation values of each functional block and the placement generated by the current step of HotFloorplan to return the corresponding peak steady-state temperature. HotFloorplan then continues through the use of simulated annealing as the search scheme through the solution space.

This work uses a cost function of the form $(A+\lambda W)T$ where A is the area corresponding to the minimum-area sizing of the current slicing structure, T is the peak steady-state temperature, W is the wire-length metric given by $\sum c_{ij}d_{ij}, 1 \leq i, j \leq n$, where n is the number of functional blocks, c_{ij} is the wire density of the interconnection between blocks i and j, and d_{ij} is the manhattan distance between their centers. λ is a contol parameter that controls the relative importance of A and W. As the units of measurement of A and W differ, λ is also used to match up their magnitudes to the same order.

There are two floorplanning schemes that we evaluate. The first, called as *flp-basic*, is a scheme where all the microarchitectural wires modeled are given equal weightage, i.e., $c_{ij} = 1, \forall i, j$. In the second, called as *flp-advanced*, the weights c_{ij} are computed in such a way that $W = \sum c_{ij}d_{ij}$ becomes an estimate of the slowdown in the execution time of the application when run on the floorplan being evaluated, in comparison to one with a default floorplan.

For the simulated annealing, we use a fixed ratio temperature schedule such that the annealing temperature of a successive iteration is 99% of the previous one. Initial annealing temperature is set such that the initial move acceptance probability is 0.99. The annealing process is terminated after 1000 iterations or after the annealing temperature becomes lesser than a threshold, whichever is earlier. The threshold is computed such that the move rejection probability at that temperature is 99%.

4.2 Wire Delay Model

Thermal-aware floorplanning algorithms are faced with an interesting temperature-performance trade-off. While separating two hot functional blocks is good for thermal gradient, it is bad for performance. To manage this trade-off during the design stage at the microarchitectural level, it is essential to have a wire delay model that is detailed enough to accurately indicate the trend of important effects and at the same time, simple enough to be handled at the architecture level. In our work, such a model is essential for evaluating the performance trade-off of the thermal-aware floorplans generated. In the *flp-advanced* scheme mentioned above, such model is also necessary during the profile-driven floorplanning phase to convert the critical wire-lengths of the floorplan into actual performance estimates. Hence, we use a previously proposed, simple, first-order model for wire delay [4, 5]. We assume optimal repeater placement and hence, wire delay becomes a linear function of wire-length. The equation from [5] that gives the wire delay for an interconnect of length l segmented optimally into segments each of size l_{opt}, is given by

$$T(l) = 2l\sqrt{rcr_oc_o}\left(b + \sqrt{ab\left(1+\frac{c_p}{c_o}\right)}\right) \quad (1)$$

where r_o, c_o and c_p are the resistance, input and parasitic output capacitances of a minimum-sized inverter respectively. $a = 0.7$, $b = 0.4$ and r and c are the resistance and capacitance of the wire per unit length respectively. We use the equation for l_{opt} and its measured values for a global 130 nm wire (2.4 mm) from [4] and also assume that $c_o = c_p$. We then obtain the values of r and c for global and intermediate level wires at the 130 nm technology node from [3]. Using these and the equation for l_{opt}, we obtain the l_{opt} value for intermediate level wires also (since l_{opt} only depends on $\sqrt{rc}$ of that metal layer), which is found to be 1.41 mm. Using the above mentioned equation and constants derived from previously published works, we compute the delay of a global or intermediate level wire, given its length for the 130 nm technology node. Assuming a clock frequency of 3 GHz, using this model, the delay of a 5 mm wire amounts to 1.69 cycles at the global layer and 2.88 cycles at the intermediate layer.

4.3 Simulation Setup and Evaluation

The microarchitectural performance model we use is a derivative of the SimpleScalar [20] simulator, the power model is a derivative of Wattch [21] and the thermal model used is HotSpot version 2.0 [2]. The basic processor architecture and floorplan modeled is similar to [22], i.e., closely resembling the Alpha 21364 processor. The leakage power model is also similar to [22] which uses ITRS [23] projections to derive the empirical constants. The differences are mentioned here. This paper uses a later version of HotSpot which additionally models an interface material of thickness 75μ between the die and the heat spreader. Further, the package thermal resistance is 0.1 K/W and the ambient temperature is at 40^o C. The threshold at which the thermal sensor of the processor engages DTM (called the trigger threshold) is 111.8^o C while the absolute maximum junction temperature that the processor is allowed to reach with DTM (called the emergency threshold) is 115^o C. The floorplan similar to Alpha 21364 processor core is scaled to 130 nm and is located in the middle of one edge of the die. Figure 1 shows this base processor floorplan. The entire die size is 15.9 mm x 15.9 mm while the core size is 6.2 mm x 6.2 mm. In other words, the manhattan distance between diagonally opposite corners of the core is 4.21 cycles if a signal travels by a global wire while 7.16 cycles

when it travels by an intermediate level wire. The floorplanning schemes mentioned above operate on the set of blocks shown in Figure 1. For the purposes of the floorplanning algorithm, all the core blocks are allowed to be rotated and the maximum allowed aspect ratio is 1:3 except when the aspect ratio of a block in the base processor floorplan is itself greater than that. In that case, the aspect ratio of the block in the basic floorplan, rounded to the nearest integer, forms the upper limit on the allowable aspect ratio. Moreover, for this paper, HotFloorplan operates only upon the core functional blocks. Once the core floorplan has been computed, the L2 cache is just wrapped around it so as to make the entire die a square.

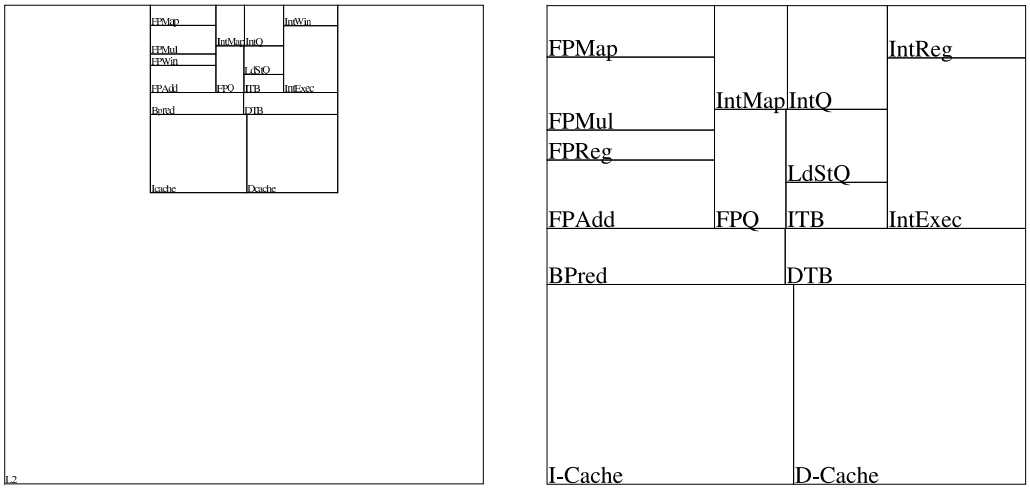

Figure 1: (a)The basic floorplan corresponding to the 21364 that is used in our experiments. (b) Close-up of the core area.

The first step of our thermal-aware floorplanning is profiling to obtain the average power dissipation values for each of the functional blocks. Hence, we use a set of 12 benchmarks from the SPEC2000 [19] benchmark suite for this purpose. These benchmarks are also used in the evaluation of our schemes. During the profiling phase, the benchmarks are run with *train* inputs while the evaluation runs use the *reference* input set. A subset of the benchmarks was chosen so as to minimize the simulation time. However, the choice was made carefully to exclude any bias. Of the 12 benchmarks, 7 are integer benchmarks and 5 are from the floating point suite. They form a mixture of hot and cold, power hungry and idle benchmarks. The list of benchmarks and their characteristics is shown in Table 2. Whether it is from the integer or floating point suite is indicated alongside the benchmark name. The temperatures shown are transient values across the entire run of the benchmark. All the benchmarks are simulated for 500 Million instructions after an architectural warm-up of 100 Million instructions and a thermal warmup of 200 Million instructions. Like [22], the simulation points for the reference runs are chosen using the SimPoint [24] tool, while the profile runs are just simulated after fast-forwarding for 2 Billion instructions to remove unrepresentative startup behaviour.

In order to model the performance impact of floorplanning, this work models in Simplescalar, the delay impact of 13 major architecture level wires that connect the blocks of the floorplan shown in Figure 1. These are by no means exhaustive but attempt to capture the most important wire delay

Table 2: Benchmark characteristics

Bench.	IPC	Average Power (W)	Average Temp. (^{o}C)	Peak Temp. (^{o}C)
bzip2 (I)	2.6	42.2	81.7	127.1
gcc (I)	2.2	39.8	79.3	121.4
gzip (I)	2.3	39.3	79.1	122.1
crafty(I)	2.5	39.3	79.0	120.0
eon(I)	2.3	38.6	79.0	113.5
art(F)	2.4	41.9	78.7	109.9
mesa(F)	2.7	37.4	78.2	114.6
perlbmk(I)	2.3	37.1	76.9	117.3
facerec(F)	2.5	33.6	74.4	107.5
twolf(I)	1.7	28.8	68.6	98.6
mgrid(F)	1.3	19.6	61.2	77.6
swim(F)	0.7	11.2	51.6	59.8

effects, especially with very little connectivity information available at the architectural level. In addition to the profile information gathered about the power dissipation of the functional blocks, the *flp-advanced* scheme also makes use of summary information collected during the profiling phase about the performance impact of each of the 13 wires. This data is presented in Figure 2. x-axis of the figure shows extra delay incurred due to a particular wire and the y-axis shows the resulting performance slowdown. The slowdown is computed in comparison to the base Alpha 21364-like microarchitectural model.

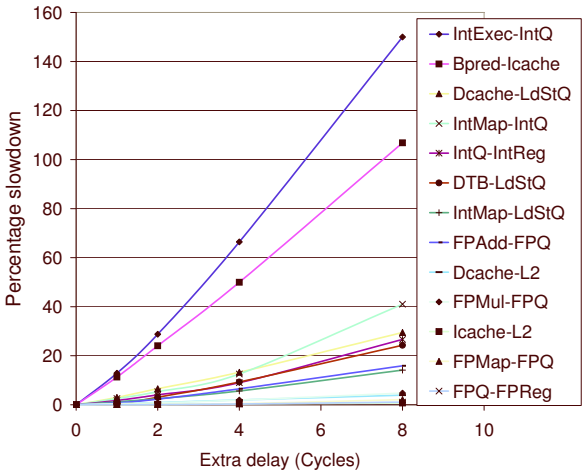

Figure 2: Performance impact of varying the delay on critical wires.

This clearly shows that some wires are more critical than others for performance. The *flp-advanced* scheme is designed to exploit this. Given a floorplan, its cost function tries to estimate its

performance in comparison with the base Alpha-like floorplan. We do this normalization as a sanity check for our wire delay model. While the wire delay model we use might be accurate in terms of the time it takes for a signal to propagate through a wire of given length, it ignores routing information and metal layer assignment because of such details not being available at the architectural level. So, we use the microarchitectural model itself as a sanity check for the wire delay model. For instance, in the base floorplan, assuming global wires, the wire delay model indicates that the delay between the FPMap and FPQ units is 1.13 cycles (2 cycles when rounded to the next higher integer). However, these units microarchitecturally correspond to the dispatch stage and it just takes 1 cycle in the Alpha 21364 pipeline for dispatch. Hence, when comparing performance of a floorplan with the base floorplan, for each of the 13 wires, we find the difference in the corresponding wire delays between the given floorplan and the base case. Only this difference, and not the actual wire delay indicated by the model, is rounded to the nearest higher integer cycle boundary and used in our performance model as the extra delay incurred. If the new floorplan has a wire shorter than the base case, it is ignored. This style of performance modeling is advantageous to the base floorplan but is also justifiably so because the base floorplan is derived from an actual processor and hence is most likely to be optimized in the best possible manner for performance. If a wire is longer in the base floorplan, it is still probably the optimal point for performance. Hence, we do not count the wires shorter in the floorplans generated by our schemes. Also, in order to deal with the issue of metal layer assignment, we take the approach of doing a sensitivity study with two extremes—all wires being global vs. all wires being intermediate. Studying these two extremes will show the best- and worst-case gains achievable by floorplanning.

The *flp-advanced* scheme uses the data from Figure 2 for its cost function. A simple linear regression analysis is performed on the data and a straight line fit is made between the extra wire delay (in cycles) and the performance slowdown for each wire. The slope of this line gives the performance slowdown per cycle of extra delay. Assuming that the performance impact of different wires add up, *flp-advanced* uses a summation of these individual slowdowns to obtain an estimate of the overall slowdown due to a particular floorplan. Effectively, this can be factored in its cost function in the wire-length term $W = \sum c_{ij} d_{ij}$ itself. As the wire delay value depends linearly on the d_{ij} term, the slowdowns can be incorporated in the c_{ij} term after proper scaling by the delay model constants to convert the d_{ij} term, which is in meters, to percentage slowdown, which is dimensionless.

In evaluating the performance impact of the floorplanning schemes, this paper compares them with control theoretic DVS [22] as the DTM technique where the voltage is varied from 100% to 50% in ten discrete steps. The frequency is also changed accordingly. The time taken for changing the voltage and resynchronizing the PLL is assumed to be 10 μs. Two versions of DVS are modeled. In the first, called *dvs*, the processor stalls during the 10 μs interval when the voltage is being changed. In the second, called *dvs-i* (for 'ideal dvs'), the processor continues to execute albeit the new voltage becomes effective only after the 10 μs period. Finally, we would like to mention that while the thermal-aware floorplanning is designed to reduce temperature, still a DTM scheme is required as a fallback in case the temperature rises beyond the threshold even with floorplanning. This is also a reason why floorplanning is not a replacement for DTM but a complement to it.

5. Results

First, we present the floorplans selected by the *flp-basic* and *flp-advanced* schemes. Figure 3 shows the core floorplans. The dead space in the floorplans is 1.14% for *flp-basic* and 5.24% for *flp-advanced*, computed as ratios to the base core area. In case of the latter, the extra space is chosen by the floorplanner as a trade-off for maintaining both good thermal behaviour and performance. With current day microprocessors being more limited by thermal considerations than by area, we feel that a 5% overhead could be tolerated. There is a possibility that this extra area could be used for better performance. However, due to diminishing ILP, as processors are moving away from increasing single processor resources to more throughput-oriented designs, area is becoming less critical than the other variables. Also, since clock frequencies are limited today by the cooling capacity of a processor, if floorplanning reduces the peak temperature significantly, then similar to the temperature-tracking dynamic frequency scaling scheme of [22], the clock frequency could probably be increased to compensate for, or even enhance the performance.

The aspect ratios of the entire core and the data cache is also interesting. Right now, the aspect ratio of the core is not constrained by any upper bound while that of the data cache is limited by a bound of 1:3. The fact that the floorplanner chooses such aspect ratios as shown in Figure 3 is interesting and suggests future work, both to explore the pros and cons of such aspect ratios from an implementation and performance perspective, and to continue refinement of the floorplanner.

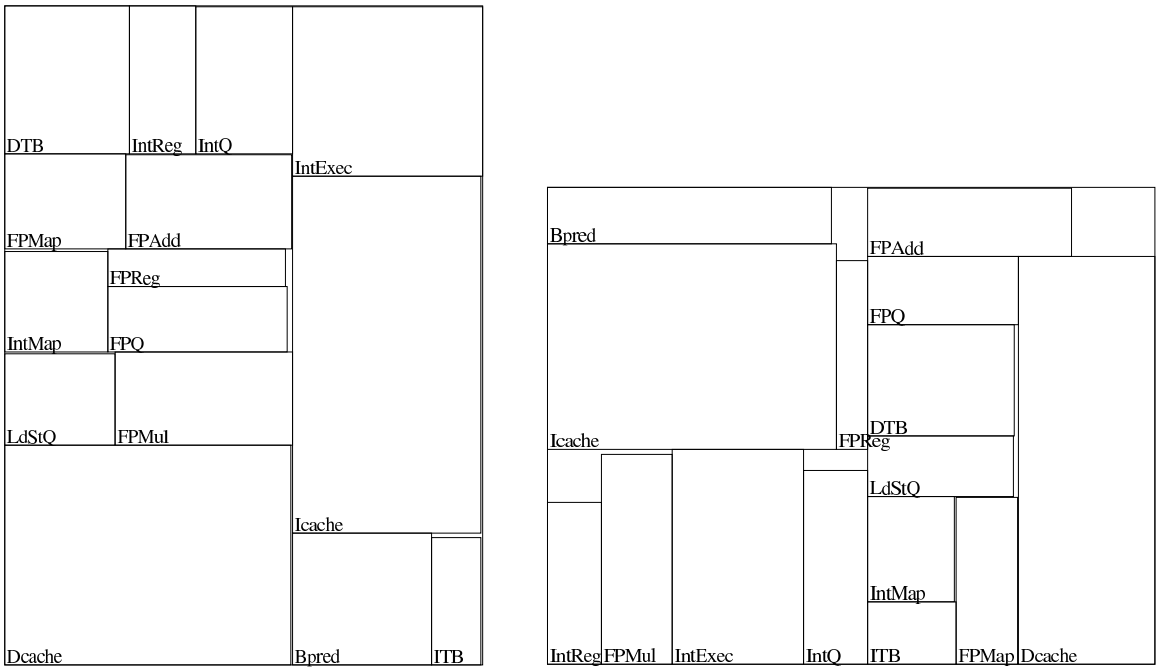

Figure 3: Floorplans generated by (a) *flp-basic* and (b) *flp-advanced* schemes.

These floorplans are then analyzed using the wire model in comparison with the base floorplan. For the *flp-basic* floorplan, its weighing all the 13 wires equally has resulted in most wires being shorter than the base floorplan. The only longer wires are Bpred-Icache, DTB-LdStQ and IntMap-IntQ. The first two are longer by 1 cycle while the last is longer by 2 cycles irrespective of the assumption about the metal level of the wires (global vs. intermediate). The total wire-length of this

floorplan is better than that of the base case by 12.7%. However, the longer wires are those critical to performance. In case of the *flp-advanced* scheme, five of the 13 wires are longer than the base floorplan. They are: IntQ-IntReg, Dcache-L2, FPMul-FPQ, Icache-L2 and FPMap-FPQ. All except the FPMul-FPQ interconnect are longer by 1 cycle, which is longer by 2 cycles. While the total wire-length of this floorplan is longer than the base floorplan by 13.7%, it can be seen from Figure 2 that these wires are less critical to performance than the wires longer in the *flp-basic* floorplan. This can be seen better when the performance results are shown.

Figure 4 shows the impact of our floorplan schemes on peak temperature. The leftmost bar in each group shows the base case. The middle bar shows the data for *flp-basic* and the rightmost one is for *flp-advanced*. The temperature reduction due to floorplanning occurs because of three main reasons. One is the lateral spreading of heat in the silicon. The second is the reduction of power density due to performance slowdown and the third is the reduction of leakage power due to the lowering of temperature. In order to decouple the effect of the later two from the first, each bar in the figure shows two portions stacked on top of each other. The bottom portions, called *basic* and *advanced* respectively, show the combined effect of all the three factors mentioned above. The top portions, called *basic-spread* and *advanced-spread*, show only the effect of spreading. This data is obtained by setting the power density of the new floorplans to be equal to that of the base case and observing the steady-state temperature for each benchmark. This does not involve the performance model and hence the effects of slowdown and reduced leakage. It is to be noted that in the *basic* and *advanced* portions of the graph, we assume zero power density for the white spaces generated by our floorplanner. However, the results do not vary much when the white spaces are assigned a power density equal to the minimum power density on the chip (which is usually in the L2 cache). In fact, in the *basic-spread* and *advanced-spread* portions of the graph shown, we actually do assign power densities in such a manner.

Figure 4: Impact of floorplanning on peak temperature.

It can be seen from the graph that with 115^o C emergency threshold, all thermal emergencies have been eliminated by floorplanning itself, even if not accounting for power reduction due to performance slowdown and leakage reduction. Also, between *flp-basic* and *flp-advanced*, the latter shows better temperature reduction. This is because of its increased area due to white spaces,

which absorb the heat from hotter units. Also, it can be observed that a significant portion of the temperature reduction comes from spreading. *flp-basic* shows a larger reduction in temperature due to performance and leakage effects when compared to *flp-advanced*. As it will be shown later, this is because the slowdown in performance itself is larger for that scheme. On the average, *flp-basic* and *flp-advanced* reduce peak temperature by 21.9 and 23.5 degrees respectively with 12.6 and 17.2 degrees respectively being just because of spreading. Since the peak temperatures with floorplanning are much lower than the emergency threshold, a careful increase in the processor clock frequency could compensate for the performance loss, still keeping the peak temperature within desired limits.

Figure 5 shows the performance impact of the schemes. The *flp-basic* and *flp-advanced* schemes are compared against *dvs* and *dvs-i*. The *advanced-g* and *advanced-i* bars correspond to the *flp-advanced* floorplan with global and intermediate metal layer assumptions respectively. The *basic* bar corresponds to the *flp-basic* scheme. There are no separate bars for different metal layer assumptions for *flp-basic* because the wire model's extra delay predictions fall into the same cycle boundary in both cases. The graph also shows a few other DVS schemes named in the format *'scheme-threshold'* where *'scheme'* is either *dvs* or *dvs-i* and the *'threshold'* is the thermal emergency threshold for the DTM scheme. While the normal emergency threshold is 115^o C for our experiments, we show these additional data as a sensitivity study with respect to the threshold.

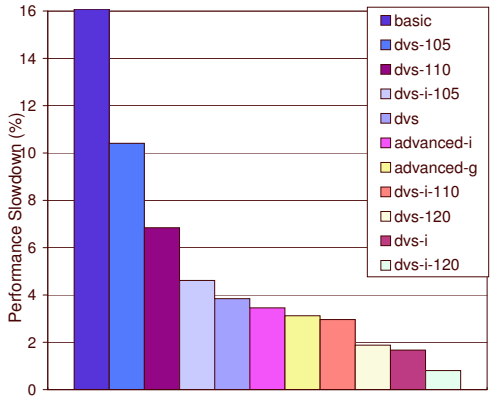

Figure 5: Performance slowdown of the various thermal management schemes.

It can be seen from the graph that *flp-advanced* performs better than *flp-basic*. It should be noted that its total wire-length is 30.3% *worse* than *flp-basic*. Still, its performance is better than flp-basic because of its high weightage for the critical wires. This also shows that a simple wire-length metric cannot be used as a proxy for performance when optimizing for temperature at the microarchitectural level. It could lead to an erroneous conclusion projecting a slower floorplan as the better one because of its shorter wire-length. *flp-advanced* is also quite competitive with the DTM schemes. It is slightly better than regular DVS and while worse than ideal DVS, is comparable to it. Even when the emergency threshold is reduced to 105^o C, the performance of the floorplan schemes does not change because the peak temperature with floorplanning is well below that and the fallback DTM is never engaged. However, changing the threshold affects the DTM schemes adversely. At a threshold of 105^o C, even ideal DVS is worse than *flp-advanced*. In real processors, the threshold temperature is set based on considerations like the capability of the cooling solution, leakage, lifetime and reliability. It is designed to be well above the average-case peak temperature. As technology scales

and as this average-case temperature itself increases, the gap between the threshold and the peak temperature gets smaller. The data shown in Figure 5 with threshold temperatures lower than 115^o C aim to model this narrowing gap.

6. Conclusions and Future Work

This paper presented a case for considering microarchitectural floorplanning for thermal management. It described HotFloorplan, a microarchitectural floorplanning tool that incorporates profile information to evaluate the temperature-performance trade-off early in the design stage. Results of this work show that there is a significant peak temperature reduction potential in floorplanning. In our experiments, all the thermal emergencies were removed by just floorplanning alone. A major part of this reduction comes from lateral spreading while a minor portion also comes from reduced leakage and slowed down execution. In comparison with a simple performance metric like the sum of the lengths of all wires, a profile-driven metric that takes into account the amount of communication and the relative importance of the wires reduces temperature better without losing much performance. In fact, the simple scheme results in a floorplan better in terms of total wire length (and temperature) but significantly worse in terms of performance. In order to optimize performance and temperature, the profile-driven scheme trades off a third variable—area. A tolerable area overhead is used in reducing temperature significantly without compromising performance. In comparison with DVS DTM scheme, the profile-based floorplanning scheme performed competitively. As the gap between the average-case peak temperature and the thermal envelope is narrowing down, the performance impact of DTM is on the rise. A combination of floorplanning and DTM could address this issue effectively. By reducing the peak temperature, floorplanning can reduce the amount of time DTM is engaged, thereby also reducing the undesirable clock and real time effects of DTM. Furthermore, since the peak temperature with floorplanning is significantly lesser than the emergency thresgold, the small performance impact of floorplanning could possibly be compensated by an increase in processor frequency, still staying within the desired thermal limits. While floorplanning reduces temperature, it does not eliminate the need for DTM. Even with floorplanning, DTM is necessary as a failsafe option. Moreover, both DTM and floorplanning address two orthogonal issues of power density and lateral spreading. Hence, they can complement each other in achieving the same objective.

In our immediate future work, we would like to investigate the effect of constraining the aspect ratio of the entire core area in our floorplanning schemes and its impact on the magnitude of white space. However, as a more general future direction of research, we would like to study the effects of thermal-aware floorplanning in multi-core architectures. This work has given an architectural framework to treat the area variable quantitatively. This opens up many interesting venues of future exploration. One could research efficient ways of trading off area and more precisely, white space, against the design constraints of temperature, power and performance. Combining such research with existing DTM schemes or coming up with new DTM schemes that work synergistically taking into account the area variable, could be further fruitful directions of research.

Acknowledgments

This work is supported in part by the National Science Foundation under grant nos. CCR-0103364 and CCF-0429765, an Army Research Office grant no. W911NF-04-1-0288, an Excellence Award from the Univ. of Virginia Fund for Excellence in Science and Technology, a grant from Intel MRL

and an IBM Research Faculty Partnership Award. We would also like to thank Wei Huang and Mircea Stan for their helpful and constructive feedback.

References

[1] D. F. Wong and D. L. Liu, "A new algorithm for floorplan design," in *Proccedings of the Design Automation Conference*, pp. 101–107, June 1986.

[2] W. Huang, M. R. Stan, K. Skadron, S. Ghosh, K. Sankaranarayanan, and S. Velusamy, "Compact thermal modeling for temperature-aware design," in *Proceedings of the ACM/IEEE 41st Design Automation Conference*, pp. 878–883, June 2004.

[3] V. Agarwal, S. W. Keckler, and D. Burger, "The effect of technology scaling on microarchitectural structures," Tech. Rep. TR-00-02, University of Texas at Austin Computer Sciences, May 2001.

[4] K. Banerjee and A. Mehrotra, "Global (interconnect) warming," *IEEE Circuits and Devices Magazine*, vol. 17, pp. 16–32, September 2001.

[5] R. H. J. M. Otten and R. K. Brayton, "Planning for performance," in *DAC '98: Proceedings of the 35th annual conference on Design automation*, pp. 122–127, 1998.

[6] S. H. Gerez, *Algorithms for VLSI Design Automation*. John Wiley & Sons, Inc., 1999.

[7] M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design*. McGraw-Hill Higher Education, 1996.

[8] L. Stockmeyer, "Optimal orientations of cells in slicing floorplan designs," *Information and Control*, vol. 57, no. 2-3, pp. 91–101, 1983.

[9] R. H. J. M. Otten, "Efficient floorplan optimization," in *Proceedings of the International Conference of Computer Design*, pp. 499–502, 1983.

[10] M. Ekpanyapong, J. R. Minz, T. Watewai, H. S. Lee, and S. K. Lim, "Profile-guided microarchitectural floorplanning for deep submicron processor design.," in *Proceedings of the 41th Design Automation Conference*, pp. 634–639, 2004.

[11] J. Cong, A. Jagannathan, G. Reinman, , and M. Romesis, "Microarchitecture evaluation with physical planning," in *Proceedings of the 40th Design Automation Conference*, June 2003.

[12] G. Chen and S. Sapatnekar, "Partition-driven standard cell thermal placement," in *ISPD '03: Proceedings of the 2003 international symposium on Physical design*, pp. 75–80, 2003.

[13] C. N. Chu and D. F. Wong, "A matrix synthesis approach to thermal placement," in *ISPD '97: Proceedings of the 1997 international symposium on Physical design*, pp. 163–168, 1997.

[14] W. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T.Theocharides, and M. J. Irwin, "Thermal-aware floorplanning using genetic algorithms," in *Sixth International Symposium on Quality of Electronic Design (ISQED'05)*, March 2005.

[15] M. Ekpanyapong, M. B. Healy, C. S. Ballapuram, S. K. Lim, H. S. Lee, and G. H. Loh, "Thermal-aware 3d microarchitectural floorplanning," Tech. Rep. GIT-CERCS-04-37, Georgia Institute of Technology Center for Experimental Research in Computer Systems, 2004.

[16] K. Sankaranarayanan, S. Velusamy, and K. Skadron, "Microarchitectural floorplanning for thermal management: A technical report," Tech. Rep. CS-2005-08, University of Virginia Department of Computer Science, May 2005.

[17] Y. Han, I. Koren, and C. A. Moritz, "Temperature aware floorplanning," in *Second Workshop on Temperature-Aware Computer Systems(TACS-2), held in conjunction with ISCA-32*, June 2005.

[18] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning : Enabling hierarchical design," *IEEE Transactions on VLSI*, vol. 11, pp. 1120–1135, December 2003.

[19] Standard Performance Evaluation Corporation, "SPEC CPU2000 Benchmarks." http://www.specbench.org/osg/cpu2000.

[20] D. C. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *Computer Architecture News*, vol. 25, pp. 13–25, June 1997.

[21] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proceedings of the 27th Annual ACM/IEEE International Symposium on Computer Architecture*, pp. 83–94, June 2000.

[22] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proceedings of the 30th Annual ACM/IEEE International Symposium on Computer Architecture*, pp. 2–13, Apr. 2003.

[23] SIA, *International Technology Roadmap for Semiconductors*, 2001.

[24] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 45–57, October 2002.