Fast Thermal-Aware Floorplanning using White-Space Optimization

Sheldon Logan, Matthew R. Guthaus
Department of CE, University of California Santa Cruz, Santa Cruz, CA 95064
{slogan,mrg}@soe.ucsc.edu

***Abstract*— The power density of modern ICs continues to increase with each new process technology. Larger power density blocks result in higher temperatures which in turn decrease the reliability of chips and produce more leakage power. In this paper we present a method to help reduce the temperature of chips at the floorplan design level by adjusting block utilizations based on the available whitespace in a floorplan. We also briefly outline a method for fast and accurate thermal floorplanning. Our experimental results show that peak IC temperatures can be significantly reduced at the floorplan design stage by using the aforementioned methods without sacrificing significant increases in floorplanning run-time, or wirelength.**

I. Introduction

High on-chip temperatures have quickly become one of the major concerns for modern integrated circuit designers. Extreme power densities due to the aggressive scaling of transistor sizes have resulted in large peak temperatures and drastic temperature gradients. This chip temperature problem has caused most designers to now consider temperature, along with power, in the early parts of the design phase.

One way in which designers have tried to address the temperature problem is with thermal-aware floorplanning, which consists of adjusting the cost function of a floorplanner to include some temperature metric (maximum or average) in addition to the other typical design metrics such as area and wirelength. This method of thermal floorplanning has two major drawbacks. Firstly, computing thermal profiles is very slow in comparison to the other design metrics such as wirelength and area. Consequently, adding temperature to the cost function results in undesirably long floorplanning run times. The second drawback is that there is only one method of reducing temperatures, separating high temperature blocks. Since floorplanning is done at such an early design stage, other methods of reducing temperatures, specifically smart white space usage can be used to help decrease temperatures effectively.

Previous researchers have investigated algorithms for both 2-D [1]–[3] and 3-D [4]–[7] floorplans. The above papers, however, simply change the cost function without investigating how to best decrease the temperature. The 3-D papers do have heuristics for layer ordering, but temperature is simply left as part of the objective.

Of previous floorplanning research, most use grid-based thermal simulators [1], [4], [6]–[8]. Some [4], [8] perform simplifications in the z-axis to reduce thermal simulation complexity. Cong et al. [4] mention heuristics to run incremental thermal computations only after operations that "tend to have large impacts," but they do not provide details. They claim that swapping two blocks will result in a large change while modifying a single block will not. Since rotating a block can change the perimeter and thermal interaction with adjacent blocks, it is not clear that this is completely true. In addition, a sequence of seemingly small moves may have a cumulatively large impact. In general, most floorplanners trade off accuracy in temperature calculations for run-time. The temperature is typically modeled using detailed finite-element (FEM) simulation [9], [10], compact resistive network modeling [11], or other approximations [12].

In an orthogonal direction, there has been significant research in fast and accurate chip thermal simulators that are not grid-based. Semi-analytic approaches using Green's functions [13] and image filtering approaches [14] have all been proposed by researchers as faster methods of temperature estimation in ICs. However, even though these methods are much faster than the grid-based approaches used in previous floorplanners, they still require much longer run-times than calculating traditional design metrics such as area and wirelength.

Our contributions in this paper are two-fold. First, we show that in addition to moving blocks around in a floorplan, an effective way to lower maximum temperatures is by adjusting white space usage. Second, we create a power metric for thermal floorplanning that is faster than other methods and still results in excellent solutions.

Our work proceeds as follows: Section II introduces the theoretical and experimental motivation for our work. Section III then provides an outline of our novel thermal-aware floorplanner. Section IV provides the experimental setup and then the results obtained from our fast thermal-aware floorplanner are presented in Section V. Finally, Section VI concludes the results.

II. Integrated Circuit Thermal Analysis

Heat is created in ICs when the transistors which comprise the various logic blocks dissipate power. The rate at which this heat is removed determines the temperature of the block and is governed by Heat Conduction Equation:

$$-k\nabla^2 T = g \tag{1}$$

where g is volume power density(W/m^3), T is the temperature(K) and k is the thermal conductivity (W/mK). The boundary conditions for Equation 1 are usually assumed to be as follows: The vertical sides of the chip, and the side not attached to the heat sink are assumed to be adiabatic. The side of the chip attached to the heat sink is assumed to be convective. Equation 1 also assumes steady state temperature and that the thermal conductivity is not a function of temperature. These assumptions and approximations are used in most thermal simulators used in floorplanning.

A. Integrated Chip Cooling

There are several factors that affect chip cooling that can be controlled during the floorplanning stage of IC development such as block power densities, proximity of hot blocks to one another and proximity of hot blocks to the chip edges.

Analyzing Equation 1 it is not quite clear how power densities affect chip cooling. Consequently a simplified version of Equation 1 in which heat flows only in the z-direction (heat flowing upwards from the substrate to heat sink) will be utilized. While this assumption is not realistic, it allows one to gain a more intuitive understanding of heat flow in ICs. The implications of this assumption will be discussed later. Given the 1-D assumption, Equation 1 reduces to:

$$g = k\frac{\Delta T}{\Delta z} \tag{2}$$

where Δz is the distance between substrate and the heat sink, ΔT is the difference in temperature between the ambient air and substrate, k is the effective thermal conductivity from the substrate to the ambient air and g is the power density(W/m^2) of the block under consideration. Equation 2 can be rewritten as:

$$T_{substrate} = \frac{\Delta z}{k} g + T_{air} \tag{3}$$

where $T_{substrate}$ and T_{air} are the substrate and ambient air temperatures, respectively.

Analyzing Equation 3 it is quite clear that the block temperatures can be decreased by lowering block power densities. Block power densities can be lowered by two means: 1) Lowering the power used by the block and 2) Increasing the area of the block. Method 1) is applicable at the floorplanning stage since as shown in [2], [6], the leakage power of a block varies with different floorplans. Method 2 is one of the primary focuses of the paper; what is the best method of allocating available whitespace to high power density blocks in floorplans to increase their area and consequently reduced their power densities.

Analyzing Equation 3 it would appear that block temperatures are independent of other blocks or floorplan position. This fallacy is a result of the approximation (heat flow in one direction) used in deriving Equation 3 from Equation 1. When blocks dissipate power the primary direction of heat flow is in the z-direction however, heat flows in the other cardinal directions. The amount of heat flow is determined by the difference in temperature between the block under consideration and the neighbouring blocks. If the difference in temperature is small, there is less heat flow in the other cardinal directions. Consequently a hot block surrounded by other hot blocks will have a higher temperature than a hot block surrounded by cooler blocks. This phenomena is called thermal coupling. In a similar vein, since the edge of the chips are almost adiabatic, hot blocks placed close to the edge of a chip will have higher temperatures than if they were placed closer to the center. It should also be noted that blocks with the same power density and different areas will have different temperatures due to the fact that a larger block will have less thermal diffusion in its center than a smaller block. As a result the larger block will have a higher temperature than the smaller block.

To illustrate these effects we ran several experiments using the well know chip thermal simulator HotSpot [15]. For our first experiment we incrementally increased the distance between two high power-density blocks. The results shown in Figure 1 illustrates that block temperatures can be decreased by spreading high power density blocks away from each other.

The second experiment involved incrementally increasing the distance of a high power density block from the edge of the chip. The results shown in Figure 1 illustrate that block temperatures can also be decreased by spreading high power density blocks from the edge of the chip which is congruent with the results of [16].

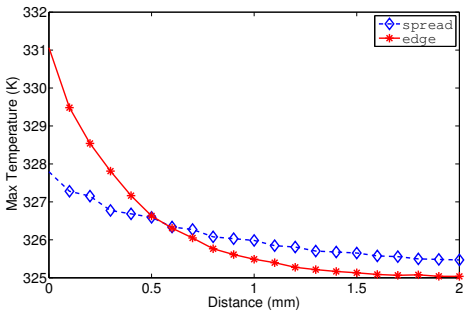


Fig. 1. Maximum Temperature vs Distance from Edge for a 100 W/cm^2 Block and Maximum Temperature vs Separation Distance for two 100 W/cm^2 Blocks.

For our final experiment we incrementally increased the area of a 200W/cm^2 block. The results shown in Figure 2 illustrate that block temperatures are in indeed dependent on area which is congruent with the results found in [17].

III. Thermal Floorplanning

In this section we will outline methods of incorporating the findings of the previous section into a fast thermal floorplanner. Specifically we will investigate the best method of allocating whitespace to increase the area of high power density blocks. We will also discuss our novel power metric that can be used during floorplanning to evenly space high power density blocks from each other and the edge of the chip.

A. Whitespace Allocation

There are two types of blocks used in floorplanning, soft blocks and hard blocks. Hard blocks are usually IP blocks

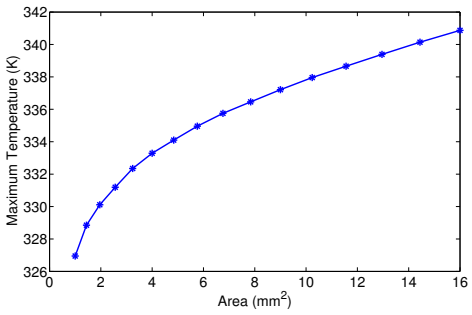


Fig. 2. Maximum Temperature vs Area for a 200 W/cm^2 Block

and consequently have a fixed area and fixed aspect ratio. Soft blocks are usually generated from synthesized logic and consequently have variable aspect ratios and areas. The area of soft blocks is usually decided by routing congestion; blocks need to have enough whitespace to be routed. We propose that temperature should also be considered when determining the final utilization of a block. Blocks that are expected to have high switching activity should be given slightly more area than what is required for routing to help reduce maximum temperatures. The amount of area to be allocated can be determined pre-floorplanning by using an algorithm to distribute the available whitespace to high power density blocks based on some metric. The area can also be allocated during floorplanning by adding additional moves to the floorplanner. For our experiments we tried both and the results are presented in the experiment section. The only bound for maximum area is the overall cost of die area. In a fixed-die scenario, the area has no cost as long as the floorplan fits.

Due to the fixed area constraint of hard blocks, increasing the area of these blocks during floorplanning is not a viable solution. An intuitive method of allocating whitespace for hard blocks is to place a small amount of whitespace around a high activity block in the form of a "halo" which would reduce thermal coupling and consequently reduce maximum temperatures.

B. Power Spreading

The main bottleneck of thermal floorplanning is calculating block temperatures. Solving Equation (1) accurately can require significant computation time, consequently, many researchers have developed alternate methods of computing block temperatures for chips [9], [13], [14]. Other researchers [5], [18], [19], have abandoned temperature simulations altogether and have used the approximated linear 1-D form of Equation (1) presented in Equation (3) to approximate block temperatures using power densities. We believe however, that the best method of doing fast thermal floorplanning is not to do temperature calculations but to develop a cost function that is minimal when high power density blocks are maximally spaced from each other and the edge of the chip. Such a cost function would avoid the need for time consuming simulations yet would be more accurate than estimating temperature from power densities alone.

A similar approach is presented in [18], however our work differs from theirs in several ways. Firstly, they do not take into consideration block areas or edge proximity which can significantly affect peak temperatures. Secondly, they use a diffusion cost that is based on the difference in power densities between two blocks whereas we use a force directed approach based on spreading high power density blocks. Thirdly, as shown in Figure 1, the thermal coupling of two blocks is quadratic with distance between them. Consequently, we use the squared distance between blocks as opposed to the linear formulation in [18]. And finally, they only consider adjacent blocks despite the fact that significant thermal coupling can exist if a small block is placed between two high power density blocks. Improvements to the work in [18] can be found in [19], but they also follow a similar thermal diffusion approach. The prior force directed approaches [6], [7] perform temperature (not power) spreading and consequently require long thermal simulations.

Our power spreading approach is based on statistical mechanics. We create a set of long and short range attraction and repulsion equations to govern the interactions of high power density blocks so as to evenly space them apart and also to space them away from the edge of the chip.

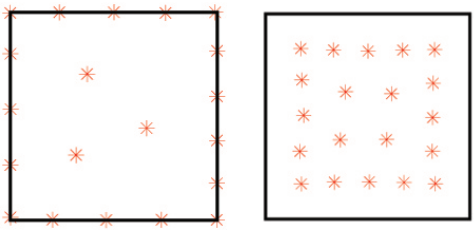
1) Block Repulsion Force: We first create a power spreading block repulsion force (P_B) that is used to maximally spread high power density blocks. The equation to calculate the average block cost is:

$$P_B = \frac{1}{n} \sum_{i,j}^{n} \frac{p_i + p_j}{d_{ij}^2} \tag{4}$$

where p_i is the power of block i, d_{ij} is the Euclidean distance between the edges of block i and block j, and n represents the number of blocks in the floorplan that have a power density greater than the average power density plus a standard deviation. Not all blocks were considered for two reasons. Firstly, by limiting the number of blocks we calculate a power spread cost for, we reduce the run-time of calculating the cost. Secondly, from our initial test results we noticed that the hottest block in a floorplan was always one with a comparatively large power density value with respect to the other blocks in the floorplan. Consequently, it is sufficient to just consider those blocks for power spreading since a block that has a low power density will never become the hottest block. It should also be noted that even though we use power density to select which blocks to consider for power spreading, the cost function uses the power of the blocks to calculate the cost. By doing this, the area of a block is taken into consideration. Otherwise, two small high power density blocks spaced a fixed distance apart would have the same cost as two large blocks with the same power density spaced at the same distance, even though there is significantly more thermal coupling between the two large high power density blocks.

2) Edge Repulsion Force: The problem with using only a block repulsion force is that it results in many blocks being pushed to the edge of the chip which can increase

peak temperature as shown in Section II-A. To illustrate this concept, we consider our cost function for 20 blocks (shown as asterisks) in Figure 3. If we only consider power spreading, many hot blocks will be forced to the chip's edge as shown in Figure 3(a).



(a) Edge Cost = 0 (b) Edge Cost = 10

Fig. 3. Power Spreading without and with Edge Costs

To combat the edge problem an edge repulsion term is incorporated into the power spreading cost. The edge repulsion term is:

$$P_E = \sum_{i}^{n} \left(\frac{p_i}{x_i^2} + \frac{p_i}{y_i^2} \right) \tag{5}$$

where x_i is the smallest distance in the x direction of block i from edge of the chip and y_i is the smallest distance in the y direction of block i from edge of the chip.

The results of the simulation with the combined edge and block repulsion cost is illustrated in Figure 3(b) and clearly show that the hot blocks are no longer congregated at the edge of the chip.

Our final power spreading cost is:

$$S_P = P_B + c \times P_E \tag{6}$$

where c determines the contribution of edge and weight repulsion forces. Changing c adjusts how close hot blocks will get to the edge of the chip. Large values results in too many blocks congregated in the center and small values of c, result in blocks being too close to the edge. For our final cost we used $c = 4$, which was found experimentally to give us the best results. The vale of c however, should be specific to the chip material parameters and the cooling solution. For example, we assumed a bulk silicon process in our experiments with a fairly large heat-sink (the default in HotSpot), however, if a SOI process was used a smaller value of c would enable better results since less heat diffusion occurs around blocks.

C. Proposed Thermal Floorplaning

Given the above insights, we wanted to formally investigate the effectiveness of area utilization and our power metric in thermal floorplanning. Our baseline floorplanner uses a simulated annealing algorithm that is is similar to other current floorplanners [4], [20]. Our simulated annealer has three moves: interchange two blocks by swapping both sequence pairs, displace a single block by swapping one pair in a single sequence pair, and the rotation of a single block. In addition to these normal moves, we add two additional perturbations to help allocate and manage whitespace in the floorplan solution: change the area of a soft block and change the virtual area of a hard block in order to increase the size of the halo around that block. We also changed the cost function to include our new power spreading metric in addition to a standard maximum temperature cost like prior works [4], [20]. The final cost function for our floorplanner is:

$$cost = \alpha \cdot \text{area} + \beta \cdot \text{HPWL} + \eta T_{Max} + \lambda S_P \tag{7}$$

where area represents the floorplan area, HPWL is the half perimeter wirelength, T_max is the maximum chip temperature, S_p is the spreading cost and α, β, η, λ, are the different weights associated with each value.

IV. Experimental Setup

We implemented our floorplanner in C++. It uses a simulated annealing algorithm with a sequence-pair (SP) representation [21]. For thermal analysis,we integrated with HotSpot 4.1 using the default parameters. We use the faster block mode for optimization and the slower, more accurate grid mode for final results. The temperature simulation is a steady-state analysis so improvements in transient simulators [9], [12]–[14] would not improve our run-times. Our results are run on a CentOS 5.1 Linux system with a 2.6GHz AMD Opteron processor and 8GB of memory.

We use the GSRC and MCNC benchmarks for our experiments. Both the GSRC and MCNC benchmarks do not have actual dimensions or power information and both of these parameters dramatically affect the performance of the heat sink and overall chip cooling. For benchmark dimensions, we scaled all the GSRC and MCNC benchmarks to be in range of medium to large area chips($0.5cm^2$ - $2cm^2$). To do this, we assume that the dimensions for the MCNC benchmarks are in microns and the GSRC benchmarks are in tenths of a micron. The aspect ratio of soft blocks is constrained to the limits specified in the respective benchmarks (0.3 to 3.0). Also, the area is limited so that blocks can not increase their area past 50% of their original area. These constraints are necessary since highly rectangular aspect ratios become increasing difficult to route and blocks that are very large will also increase the wirelengths within the blocks. Also, having too low of a block utilization will require significant buffering to drive signals across the block. In the case of hard blocks, large halos tend to increase wirelengths between blocks.

For block power information, we randomly generate power numbers using design power densities similar to the predicted 65nm node in [22]. The power densities used are $750\frac{W}{cm^2}$, $250\frac{W}{cm^2}$ and $25\frac{W}{cm^2}$ with corresponding frequencies of 15%, 45% and 40%. The mean is therefore $235\frac{W}{cm^2}$ and there is approximately a $3.2\times$ difference between the average and maximum power density as observed in [22].

During our initial experiments, we noticed that floorplans with more whitespace tend to have lower temperatures. This is due to two factors: a larger chip will decrease the chance that

TABLE II
PEAK TEMPERATURE OPTIMIZATION RESULTS FOR HARD BLOCKS

Experiment	HPWL	Max Temp (K)	Time (s)
HPWL and Power	0.20%	5.94	1.14x
HPWL and Temp	1.80%	6.12	158.4x
HPWL, Power and Halo	3.30%	9.16	1.14x

two hot blocks are close together and a larger chip area will improve the thermal conductivity to the heat sink which results in lower maximum and average temperatures. Consequently, for fair comparisons, we perform fixed-area floorplanning. The area cost during annealing is the area outside of the fixed area. We do not place constraints on the floorplan aspect ratio, however. For the experiments in the subsequent sections, we used a fixed area that is 10% larger than the total area of all blocks. All the results presented are mean values for 100 simulated annealing runs.

V. EXPERIMENTS

A. Fast Thermal Floorplanning

To verify our power density metric, we ran experiments for thermal floorplanning using both an actual temperature metric and the power metric in Equation 6. The results depicted in Table I clearly show that our power metric is effective for thermal-aware floorplanning. In the results, we perform HPWL only optimization, HPWL along with our power metric, and HPWL with integrated thermal simulation like prior works [1]–[7]. For the larger benchmarks, the integrated thermal simulation is too slow to finish in a reasonable amount of time. Our power metric, however, is very fast even when compared to HPWL-only optimization. Our maximum temperatures, however, are comparable or better than the direct temperature optimization results in all cases. The improved results are due to the global view of our cost function when compared to the other, more direct temperature optimization method. Our method may accept a move that improves the temperature in a region that is not the highest temperature, but soon becomes a hotspot whereas the temperature-direct method would reject the move. This allows our optimization to get more useful work out of the random SA moves and achieve improved results.

B. Whitespace Allocation

To determine the best method for using the available whitespace, we investigated the effectiveness of the different methods of allocating it: 1) halo allocation for hard blocks, 2) area allocation for soft blocks. The peak temperature results from the experiments are summarised in Table II and III, respectively. From Table II, we can see that almost a $2\times$ peak temperature decrease can be obtained by adjusting the halo around hard blocks. For soft blocks, Table III shows that dynamically allocating the whitespace for area utilization is effective at reducing temperatures but has only a small improvement over our power metric alone. Consequently, a more efficient method was investigated for whitespace allocation for soft blocks.

TABLE III
PEAK TEMPERATURE OPTIMIZATION RESULTS FOR SOFT BLOCKS

Experiment	HPWL	Max Temp (K)	Time (s)
HPWL and Power	1.80%	7.41	1.11
HPWL and Temp	3.00%	5.9	158.44
HPWL, Power, Static	6.40%	11.44	1.12
HPWL, Power, Dynamic	4.20%	7.60	1.12

C. Static or Dynamic Whitespace Allocation

Dynamic area allocation performs slightly better than temperature-aware placement alone, but an interesting question is whether static allocation of the whitespace before floorplanning is better than dynamically allocating during floorplanning. To study this, we compare with a greedy *a priori* algorithm that statically allocates whitespace to the most power dense blocks before floorplanning. Area is added to the highest power density block until either it is no longer the highest, there is no more available whitespace or the block is 50% larger than its original size. If it is no longer the highest power density, we start allocating area to the next highest power density block. The advantage of this method is that it is performed before annealing begins and consequently is simpler and slightly faster than the dynamic methods. Statically allocating the area might, however, result in an isolated block being inflated when the whitespace might be best used to inflate a cluster of relatively lower power density blocks that result in a high peak temperature.

We investigated both the static and dynamic floorplanning methods with 10% available whitespace. It should be noted that only 5% of the available whitespace was statically allocated for the *a priori* optimization. If all 10% of the whitespace available was used for the *a priori* optimization the floorplan would have to be perfect (i.e., contain no additional white space).

The results in Table III show that *a priori* static allocation does better than dynamic area allocation on average for all benchmarks. This occurs because increasing the area of a block will always decrease its maximum temperature. Consequently, dynamic allocation of the whitespace during floorplanning results in low power density blocks being inflated even though the whitespace might be best use for a high power density block. In addition, the usage of the SA random moves to adjust the whitespace usage detracts from more useful moves that reduce area, HPWL, and minimize the adjacency of high power blocks.

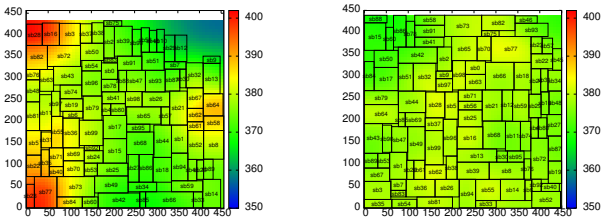
An example plot of n100 with and without power-aware optimization is shown in Figure 4 with identical fixed-areas. Figure 4(a) corresponds to HPWL-only optimization. The HPWL for this placement is 278431, the maximum temperature is 400.1K and it required 24.6s computing time. Figure 4(b) corresponds to HPWL, power spreading and static whitespace optimization. Its HPWL is 310124, the maximum temperature is 382.7K and required 28.6s computing time. These figures show that by utilizing static area allocation along with our power density metric significant temperature reductions can be obtained with a modest increase in HPWL

TABLE I

PEAK TEMPERATURE OPTIMIZATION RESULTS

	HPWL Only			HPWL and Power			HPWL and Temp		
Benchmark	HPWL	Max Temp (K)	Time (s)	HPWL	Max Temp (K)	Time (s)	HPWL	Max Temp (K)	Time (s)
n10	43653	448.40	2.87	44598	430.40	2.97	44378	440.67	108.35
n30	131170	401.33	8.40	132393	389.86	9.02	134340	389.10	1724.50
n50	172521	414.84	15.16	175237	404.88	15.59	179270	406.35	7082.00
n100	286352	394.63	28.62	290480	389.41	28.54	n/a	n/a	n/a
n200	545746	396.92	88.11	559340	393.07	114.50	n/a	n/a	n/a
n300	799656	397.83	191.05	815146	395.54	266.92	n/a	n/a	n/a
ami33	82287	358.54	10.68	84206	357.87	11.37	84627	358.26	2212.10
ami49	1190385	455.47	22.00	1192307	449.99	23.65	1262700	453.29	6495.20
apte	641408	453.27	4.89	648447	444.89	5.16	656604	446.95	92.73
hp	202489	444.36	5.23	213080	432.86	5.27	209961	439.17	133.90
xerox	540676	366.75	11.64	540177	362.03	11.91	546782	361.97	119.13
Mean Change	0	0.00	0.00	1.8%	7.41	1.12	3.0%	5.90	158.44

and a very small increase in run-time. The non-whitespace-aware placement tends to cluster unused area in the upper-right of the floorplan due to the sequence pair representation. The whitespace-aware placements however, distributes available whitespace to the internal floorplan blocks which results in fewer hotspots being created.



(a) HPWL Optimization Only

(b) HPWL, Power Spreading and Static Whitespace Optimization

Fig. 4. Example Results for n100

VI. CONCLUSIONS AND FUTURE WORK

High temperatures on chips have recently become a major concern of IC designers. In this paper we developed several methods for addressing the temperature problem in floorplanning for ICs. By strategically allocating pre-existing whitespace during floorplanning, we were able to significantly reduce the maximum temperature of the GSRC and MCNC benchmarks. We were also able to significantly decrease the runtime of thermal floorplanning by using a metric based on power density as opposed to direct thermal simulations.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grant 0720913; a Special Research Grant from the University of California, Santa Cruz; and the Sun OpenSPARC Center of Excellence at UCSC. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] W.-L. Hung *et al.*, "Thermal-aware floorplanning using genetic algorithms," in *ISQED*, 2005, pp. 634–639.

[2] A. Gupta *et al.*, "LEAF: A system level leakage-aware floorplanner for socs," in *ASP-DAC*, 2007, pp. 274–279.

[3] J.-L. Tsai *et al.*, "Temperature-aware placement for socs," *Proc. of the IEEE*, vol. 94, no. 8, pp. 1502–1518, 2006.

[4] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *ICCAD*, 2004, pp. 306–313.

[5] W.-L. Hung *et al.*, "Interconnect and thermal-aware floorplanning for 3D microprocessors," in *ISQED*, 2006, pp. 98–104.

[6] P. Zhou *et al.*, "3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits," in *ICCAD*, 2007, pp. 590–597.

[7] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *ICCAD*, 2003, p. 86.

[8] X. Li *et al.*, "Thermal-aware incremental floorplanning for 3D ICs," *ASICON*, pp. 1092–1095, 2007.

[9] T.-Y. Wang and C. C.-P. Chen, "Thermal-ADI: A linear-time chip-level dynamic thermal simulation algorithm based on alternating-direction-implicit (adi) method," in *ISPD*, 2001, pp. 238–243.

[10] Y.-K. Cheng *et al.*, "ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips," *TCAD*, vol. 17, pp. 668–681, 1998.

[11] M. Stan *et al.*, "Hotspot: A dynamic compact thermal model at the processor-architecture level." *Microelectronics Journal*, pp. 1153–1165, 2003.

[12] Y. Zhan and S. Sapatnekar, "Fast computation of the temperature distribu- tion in vlsi chips using the discrete cosine transform and table look-up," in *ASPDAC*, 2005, pp. 87–92.

[13] B. Wang and P. Mazumder, "Fast thermal analysis for vlsi circuits via semi-analytical green's function in multi-layer materials," *ISCAS*, vol. 2, pp. 409–412, 2004.

[14] J.-H. Park *et al.*, "Fast computation of temperature profiles of VLSI ICs with high spatial resolution," in *Semi-Therm*, 2008, pp. 50–54.

[15] W. Huang *et al.*, "Compact thermal modeling for temperature-aware design," in *DAC*, 2004.

[16] J. Lee, "General thermal force model with experimental studies," *Trans. on Packaging*, vol. 29, no. 1, pp. 20–29, 2006.

[17] R. J. Ribando and K. Skadron, "Many-core design from a thermal perspective," *DAC*, pp. 746–749, 2008.

[18] Y. Han *et al.*, "Temperature aware floorplanning," in *Temperature-Aware Computer Systems*, 2005.

[19] C.-T. Chu *et al.*, "Temperature aware microprocessor floorplanning considering application dependent power load," in *ICCAD*, 2007, pp. 586–589.

[20] K. Sankaranarayanan *et al.*, "A case for thermal-aware floorplanning at the microarchitectural level," *JILP*, 2005.

[21] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *TCAD*, vol. 15, no. 12, pp. 1518–1524, Dec 1996.

[22] G. M. Link and N. Vijaykrishnan, "Thermal trends in emerging technologies," *ISQED*, pp. 625–632, 2006.